c2x-cmd

Benutzerhandbuch
für
c2x-cmd

(Für die Version 1.8.0)

Änderungsgeschichte

Das vorliegende Dokument wurde bislang den folgenden Änderungen unterworfen.

Version	Datum	Autor	Inhalt/Änderung
0.1	26.10.2011	J. Gödeke	Ersterstellung

Inhaltsverzeichnis

1 CSV2XML 4

1.1 Vorwort 4
1.2 Lizenz 4
1.3 Kontakt 4
1.4 FAQ 4

2 ÜBERSICHT 5

2.1 Aufruf über c2x-wizard 5
2.2 Aufruf über die Kommandozeile 5
2.3 Installation und Aufruf 6
2.3.1 Windows 6
2.3.2 Alle Betriebssysteme auf Unix-/Linuxbasis (inkl. MAC OS X) 7

3 C2X-CMD ÜBERSICHT 8

3.1 Generelle Anmerkungen zu den Optionen 8
3.2 Additive Optionen 8
3.2.1 Parametersequenz 8
3.2.2 Optionssequenz 8
3.3 Übersicht der Optionen 8
3.3.1 Optionen die von c2x-wizard unterstützt werden 8
3.3.2 Reine c2x-cmd Optionen: 9
3.3.3 Konfigurationsdateien 9
3.4 She-Bang Ausführung 10
3.5 c2x-cmd Optionen 10
3.5.1 Option : Source filename (m) / -s 11
3.5.2 Option : Target filename (m) / -t 11
3.5.3 Option : Separators by plain text / -sep 11
3.5.4 Option : Separator by ASCII code / -sep 11
3.5.5 Option : Encoding / -enc 12
3.5.6 Option : Convert command line parameters / -cc 12
3.5.7 Option : XML Mode / -m 13
3.5.7.1 Modus 1 : <eRecord><COLname>CELLvalue</ColName></eRecord> 13
3.5.7.2 Modus 2 : <eRecord><COLname aValue="CELLvalue"/></eRecord> 13
3.5.7.3 Modus 3 : <eRecord><eItem aName="COLname">CELLvalue</eItem></eRecord> 14
3.5.7.4 Modus 4 : <eRecord><eItem aName="COLname" aValue="CELLvalue"/></eRecord> 14
3.5.7.5 Modus 5 : <eRecord COLname="CELLvalue"/> 15
3.5.8 Option : suppress empty elements / -sup 15
3.5.9 Option : write „num“ attribute to record-element / -wn 15
3.5.10 Option : write xml:id=“id." attribute to record-element / -wid 16
3.5.11 Option : Stylesheet(s) / -styles 16
3.5.12 Option : Validation / -xsd & -dtd 17
3.5.13 Option : Use Namespace / -xmlns 17
3.5.14 Option : Alias / -alias 18
3.5.15 Option : Versionsausgabe / -V 18
3.5.16 Option : Hilfe / -? 19
3.6 Weitere Beispiele 19

4 ANHANG 20

4.1 Verweise 20

1 csv2xml

1.1 Vorwort

Ich bin immer wieder erstaunt, wie häufig doch CSV Dateien in das XML Format umgewandelt werden müssen. Dabei war die erste Version meines Konverters csv2xml aus dem Jahre 2002 lediglich als Demo für eine von mir erstellte CSV-Parserklasse in C++ gedacht. Ich veröffentlichte die Software auf meiner Homepage und war über dessen Beliebtheit überrascht. Relativ bald kamen auch die ersten Verbesserungsvorschläge und vor allem der Wunsch auf, csv2xml auf Non-Windows-Plattformen laufen zu lassen. Im Jahre 2003 folgte auf Aufteilung von csv2xml in den Kommandozeilen-Konverter „csv2xml“ und die grafische Generatoroberfläche „csv2xml_win“. Zudem veröffentlichte ich csv2xml für Linux. Im Jahr 2007 benannte ich das Programm „csv2xml“ in „c2x-cmd“ um, da es in der Zwischenzeit jede Menge gleichnamiger Tools im Internet gibt. Mit der Umbenennung kompilierte ich das Kommandozeilentool für weitere gängige Plattformen. Lediglich mit der Ablösung von „csv2xml_win“ ließ ich mir satte 4 Jahre Zeit. Dies lag u.a. daran, dass mir Job und Familie nicht sehr viel Freizeit für so etwas überließen und zum anderen, dass ich erst einmal eine passende Programmiersprache für eine entsprechende Oberfläche finden musste. Will man hier plattformunabhängig sein, kommt man nicht umhin auf irgendeinem sogenannten „Framework“ aufzubauen. Dieses sollte am besten bereits auf dem Zielsystem installiert sein, denn wer will schon erst riesige Pakete für eine kleine Konvertierungsanwendung installieren. Letzten Endes fiel meine Wahl auf Java. Bis 2006 erzeugte ich meine grafischen Javaprogramme mit dem Borland JBuilder in der Version 2005. Nachdem dieser auf die Eclipse-Plattform migriert wurde und darunter vor allem der Editor zum erstellen von grafischen Oberflächen litt, war es gar nicht mehr so einfach eine Alternative zu finden. Zuletzt entschloss ich mich Mitte 2011 meine ersten Gehversuche mit Oracles (ehemals SUNs) Netbeans zu machen. Dabei ist es dann auch geblieben. Ich hoffe dass sich das Resultat sehen lassen kann und die eventuell notwendige Installation einer Java Laufzeitumgebung nicht zu viel Aufwand macht.

1.2 Lizenz

c2x-cmd & c2x-wizard unterliegen der FREEWARE Lizenz. Dies bedeutet, dass Sie die Programme zu kommerziellen oder nicht kommerziellen Zwecken kostenlos benutzen dürfen. Die Programme dürfen jedoch weder verändert noch verkauft werden.
Ich als Autor übernehme keinerlei Haftung für Schäden, die durch die Nutzung des Programmes entstehen.

1.3 Kontakt

Für weitere Fragen stehe ich Ihnen gerne zur Verfügung.
Jens Gödeke

Nutzen Sie bitte das Kontaktformular unter der folgenden Internetadresse für Anregungen und Fehlermeldungen.
http://www.jens-goedeke.de/kontakt

Die c2x-cmd Homepage:
http://www.jens-goedeke.de/tools/csv2xml

1.4 FAQ

Gibt es derzeit noch nicht.

2 Übersicht

Bei c2x-cmd handelt es sich um ein Kommandozeilenprogramm. Wie in Abbildung 1 zu sehen ist, gibt es vier verschiedene Arten den Konverter zu starten.

Hinweis : Unter Windows stehen nur die ersten drei zur Verfügung.

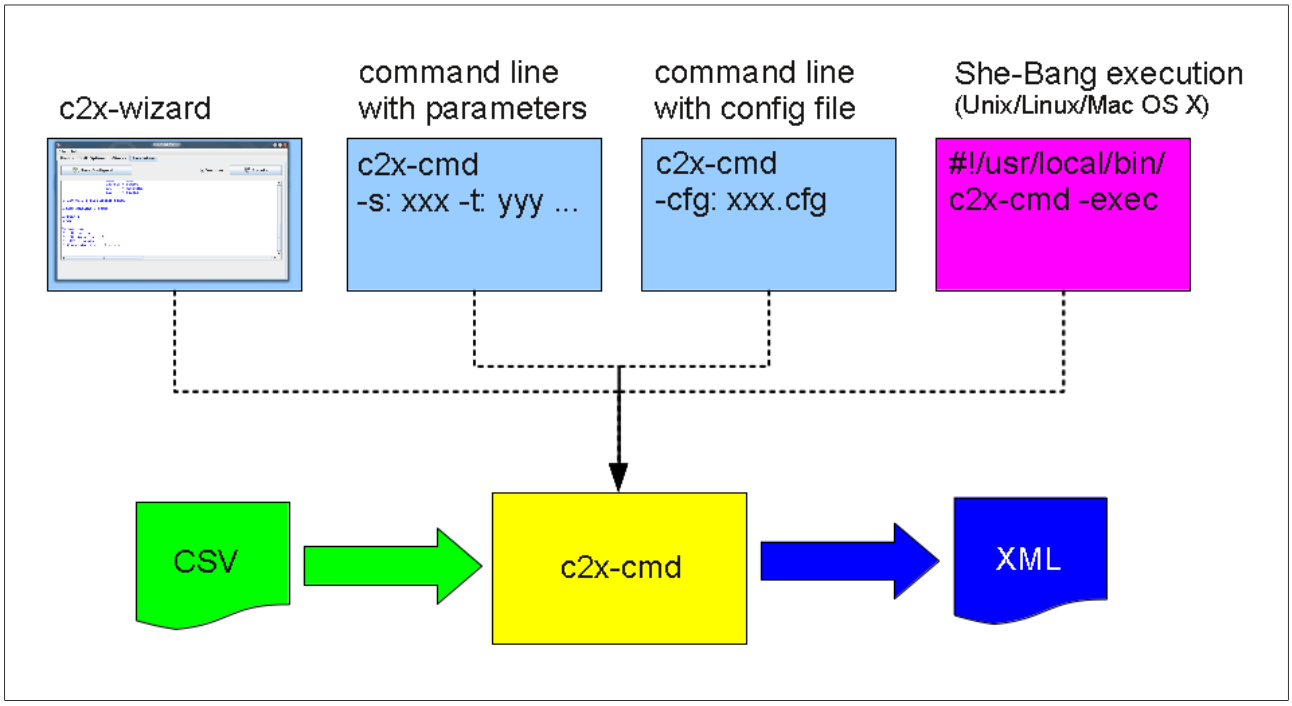

Abbildung 1: Aufruf von c2x-cmd

2.1 Aufruf über c2x-wizard

c2x-wizard soll dem Benutzer helfen, die verschiedene Optionen von c2x-cmd über eine grafische Oberfläche richtig zu setzen. c2x-wizard übernimmt dabei die folgenden Aufgaben:

- Unterstützung bei der Eingabe der für c2x-cmd relevanten Parameter
- Möglichkeit die vorgenommenen Einstellungen als Konfigurationsdatei abzuspeichern
- Möglichkeit die eingestellten Parameter direkt an c2x-cmd zu übergeben und damit die Konvertierung auszuführen.

Mehr dazu steht im c2x-wizard Handbuch.

2.2 Aufruf über die Kommandozeile

Der direkte Aufruf über die Kommandozeile hat den Vorteil der automatisierten Verarbeitung. So kann c2x-cmd direkt in Batch- oder Shell-Skripte eingebunden werden, ohne das jedes mal die grafische Oberfläche dafür aufgerufen werden muss.
Der Aufruf erfolgt dabei über die Kommandozeile:

- Mittels einzelner Parameter
- Mittels Übergabe einer (eventuell mit c2x-wizard) erstellten Konfigurationsdatei

oder (leider nur für Betriebsysteme auf Linux- oder Unixbasis):

- Mittels She-Bang[1]-Aufruf in der (eventuell mit c2x-wizard) erstellten Konfigurationsdatei

Mehr dazu in Abschnitt 3

[1] siehe http://de.wikipedia.org/wiki/Shebang

2.3 Installation und Aufruf

2.3.1 Windows

Laden Sie sich die passende ZIP Datei von meiner Homepage und entpacken Sie diese in ein beliebiges Verzeichnis. Falls Sie c2x-cmd aus Batch- oder anderen Skripten heraus starten wollen, müssen Sie das Ablageverzeichnis dem Suchpfad hinzufügen.

Dies geschieht bei Windows 7 & Vista über:
Start → Systemsteuerung → System → Erweiterte Systemeinstellungen → Reiter „Erweitert“ → Knopf „Umgebungsvariablen“

Bei Windows 2K oder XP:
Start → Einstellungen → Systemsteuerung → System → Reiter „Erweitert“ → Knopf „Umgebungsvariablen“

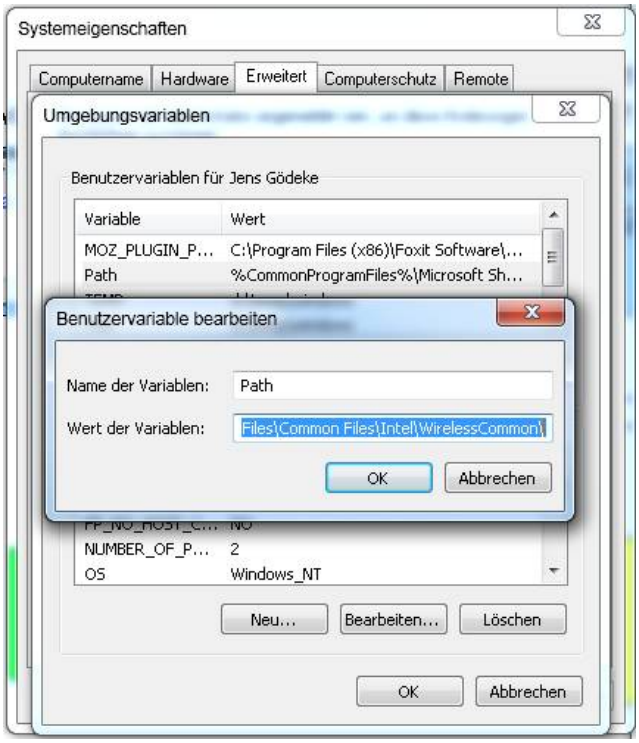

Abbildung 2: Suchpfad unter Windows erweitern

Fügen Sie an den Wert der Variable Path ein Semikolon und anschließend den absoluten Pfad des von Ihnen gewählten Ablageverzeichnisses ein.

Im Erfolgsfall, können Sie anschließend c2x-cmd testen. Die Eingabe von:

```
c2x-cmd -V
```

sollte eine Versionsinformation ausgeben:

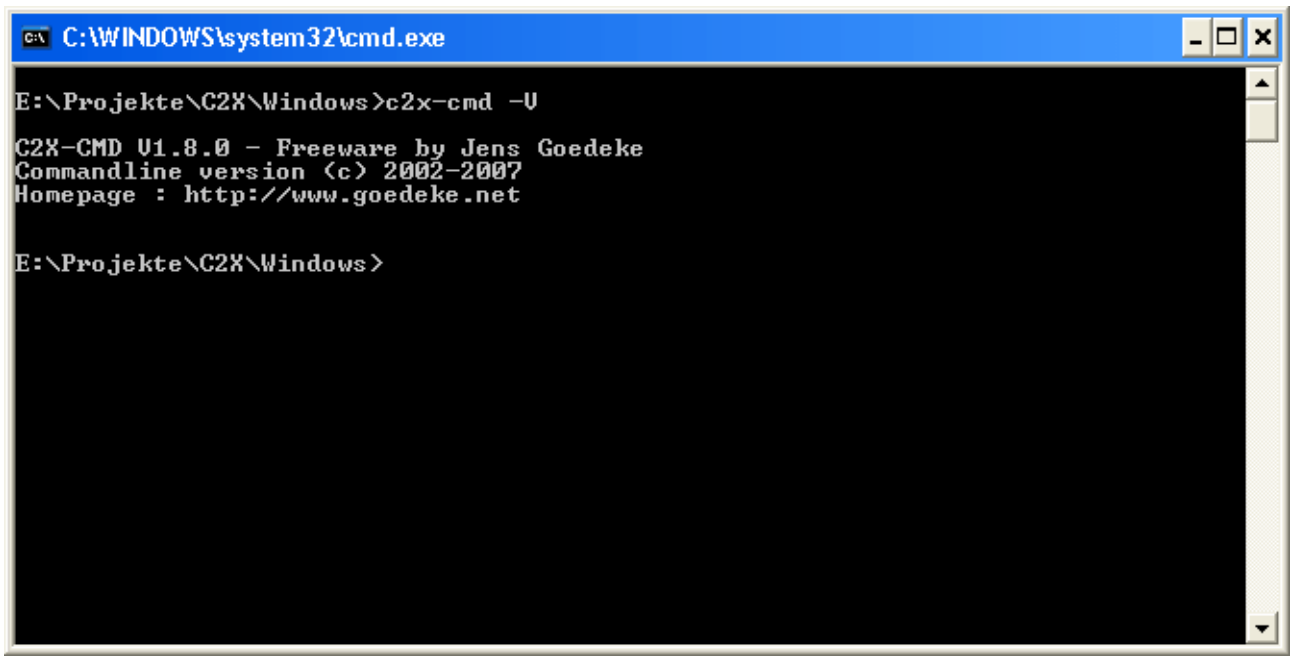

Abbildung 3: Versionsabfrage unter DOS

2.3.2 Alle Betriebssysteme auf Unix-/Linuxbasis (inkl. MAC OS X)

Hier wird das entsprechende ZIP oder tar.gz Archiv in eine beliebiges Verzeichnis entpackt.
Falls Ihre Plattform dort nicht aufgelistet ist, so lassen Sie mir eine Mail zukommen.

Entpacken Sie den heruntergeladenen Konverter in das gleiche Verzeichnis, oder mit entsprechenden Rechten in ein Verzeichnis des Suchpfades (z.B. „/usr/local/bin“ etc.). Das hat den ungemeinen Vorteil, dass Sie den Konverter jederzeit in Ihre Skripte einbetten können.

Aufrufen können Sie c2x-cmd direkt über eine Unix-/Linux-Shell (SH, KSH, BASH etc.).

```
./c2x-cmd -V
```

Dabei sollte im Erfolgsfall eine Versionsinformation ausgegeben werden.

Abbildung 4: Versionsabfrage unter Linux

3 c2x-cmd Übersicht

c2x-cmd ist ein Kommandozeilen Konverter und hat keine grafische Oberfläche. Daher ist c2x-cmd hervorragend zur automatisierten Konvertierung von Quelldateien geeignet.

3.1 Generelle Anmerkungen zu den Optionen

Die Optionen werden immer mit einem führenden Minuszeichen eingeleitet. Sind die Optionen nicht nur einfache Schalter, so folgt der entsprechende Wert durch ein Doppelpunkt (ohne Leerzeichen) getrennt.

3.2 Additive Optionen

Die Optionen -alias und -styles sind additive Optionen. Hierbei können auf zwei verschiedene Arten mehrere Einträge angegeben werden.

3.2.1 Parametersequenz

Hierbei werden die einzelnen Parameter einer Option durch Kommata getrennt angegeben.
Beispiel:

```
-alias:eRecord=Datensatz,aValue=Mein_Wert
```

3.2.2 Optionssequenz

Hierbei wird die gleiche Option mehrfach hintereinander verwendet. Diese Art der Notation bietet sich für Konfigurationsdateien an, da Sie dort zur Übersichtlichkeit beiträgt:

```
-alias:eRecord=Datensatz -alias:aValue=Mein_Wert
```

3.3 Übersicht der Optionen

3.3.1 Optionen die von c2x-wizard unterstützt werden

Die folgenden Optionen (in alphabetischer Reihenfolge) werden von c2x-wizard unterstützt:

Option	Bedeutung	Pflicht	Aufrufbeispiel
-alias:	Umbenennung von Elementen & Attributen	nein	-alias:eItem=Test,eRecord=xx
-cc:i2u	Kommandozeilenparameter von ISO-8859-1 nach UTF-8 konvertieren	nein	
-cc:u2i	Kommandozeilenparameter von UTF-8 nach ISO-8859-1 konvertieren	nein	
-dtd:	DTD Validierungsinformationen einbetten	nein	-dtd:dtd-datei
-enc:	Encoding festlegen	nein	-enc:ISO-8859-1
-m:	XML Modus	nein	-m:2
-s:	Quelldaten aus Datei übernehmen	ja	-s:dateiname
-sep:	Textseparatoren festlegen	nein	-sep:,_-
-sep:::	Asciiseparator festlegen	nein	-sep::9:
-styles:	Stylesheets einbinden	nein	-styles:css-datei,xsl-datei
-sup	Leere Elemente und Attribute nicht in die Zieldatei schreiben	nein	
-t	Zieldateiname aus Quelldateiname ermitteln und in Datei schreiben	ja	
-t:	Zieldaten in Datei schreiben	ja	-t:zieldatei
-v	Informationen zum Ablauf anzeigen (Verbose)	nein	
-wid	XML:ID in die Zieldatensätze schreiben	nein	
-wn	Datensatznummer in die Zieldatei schreiben	nein	
-xmlns:	XML Namespace ausgeben	nein	-xmlns:ns=uri

Option	Bedeutung	Pflicht	Aufrufbeispiel
-xsd:	XSD Validierungsinformationen einbetten	nein	-xsd:xsd-datei

3.3.2 Reine c2x-cmd Optionen:

Die folgenden Optionen werden von c2x-wizard **nicht** unterstützt:

Option	Bedeutung	Pflicht	Aufrufbeispiel
-V	Versionsinfo ausgeben	nein	
-?	Hilfe & Benutzungsinformationen abrufen	nein	
-cfg:	Einstellungen aus Konfigurationsdatei übernehmen.	nein	-cfg:cfg-datei
-exec	She-Bang Ausführung	nein	
-s:-	Quelldaten von STDIN übernehmen	ja	
-t:-	Zieldaten nach STDOUT schreiben	ja	

3.3.3 Konfigurationsdateien

Da die Anzahl der Optionen durchaus so manche Kommdozeilengrenze sprengen kann, besteht die Möglichkeit die Optionen in eine Textdatei zu schreiben und diese anstelle der einzelnen Optionen zu übergeben. Diese Konfigurationsdatei hat das folgende reine Textformat:

```
[# Bemerkung]|[Leerzeile]
Option 1|[Leerzeile]
[# Bemerkung]|[Leerzeile]
[Option 2]|[Leerzeile]
...
```

Sie können sich die Konfigurationsdatei bequem durch den c2x-wizard erstellen lassen. Andernfalls stellen Sie sicher, dass Sie einen Quelltexteditor (kein Textverarbeitungsprogramm wie Word, OpenOffice etc.) verwenden. Quelltexteditoren sind z.B. ...
für Windows:

```
Notepad, Notepad++, Ultraedit, ...
```

für Unix/Linux:

```
vi, emacs, gedit, kate, ...
```

Aufruf:

```
-cfg:Konfigurationsdatei
```

Beispiel für eine Konfigurationsdatei die durch c2x-wizard erstellt wurde.
Die Erklärung der einzelnen Optionen erfolgt in den nächsten Abschnitten.

```
#!c2x-cmd.exe -exec

# CSV source filename
-s:E:\Projekte\C2X\Windows\examples\source\example1.csv

# XML target filename must be optained from the csv source filename
-t

# Plain text separator(s)
-sep:;
```

```
# XML mode : <eRecord><COLname>CELLvalue</ColName></eRecord>
-m:1

# Encoding
-enc:ISO-8859-1

# Validation:
-xsd:E:\Projekte\C2X\Windows\examples\extend\mode3-schema.xsd

# Namespace
-xmlns:mode3-schema.xsd

# Aliases
-alias:eRoot=Wurzeldokument
-alias:eRecord=Datensatz

# Suppress empty elements:
-sup

# Write "num" attribute to record-element
-wn

# Write xml:id="id." attribute to record-element
-wid

# Verbose:
-v
```

3.4 She-Bang Ausführung

Diese Funktion wird nur unter Unix-/Linux-Betriebsystemen unterstützt. Dabei wird der Interpreter mit dem die betreffende Datei ausgeführt werden soll in den Kopf der Datei geschrieben. Man kennt dies insbesondere durch Shell-Skripte:

```
#!/bin/bash
```

Genau das gleiche kann jedoch auch mit beliebigen Interpretern gemacht werden. Hierfür dient bei c2x-cmd die -exec Option:

Aufruf:

```
#![Pfad zu c2x-cmd]/c2x-cmd -exec
```

Wird die entsprechende Konfigurationsdatei mittels c2x-wizard erstellt, füllt dieser die entsprechende Information (inkl. Pfad) anhand des gefundenen c2x-cmd Programmes selbst aus. Nun muss die entsprechende Konfigurationsdatei nur noch mit dem Execute-Flag („x“) versehen werden, so dass diese Datei ausführbar wird. Ein anschließender Test sieht dann z.B. so aus:

```
> chmod u+x ./myconfig.cfg
> ./myconfig.cfg
```

3.5 c2x-cmd Optionen

In dem folgenden Abschnitt werden die einzelnen Optionen von c2x-cmd erklärt.

3.5.1 Option : Source filename (m) / -s

Hiermit wird der Pfad zur CSV Quelle festgelegt.

Aufruf:

Dies kann auf zwei Arten erfolgen:

a) Als Eingabequelle wird eine Datei definiert:

```
-s:Dateiname.csv
```

b) Falls die Daten direkt aus der Standardeingabestrom (STDIN) gelesen werden sollen:

```
-s:-
```

Beispiel:

```
more eine-datenquelle.csv | c2x-cmd -s:- -t:-
```

Daten aus „eine-datenquelle.csv“ an c2x-cmd weiterleiten. Dort werden die Daten übernommen und in den Standardausgabestrom geschrieben.

3.5.2 Option : Target filename (m) / -t

Hiermit wird der Pfad zur XML Zieldatei festgelegt.

Aufruf:

Dies kann auf drei Arten erfolgen:

a) Als Eingabequelle wird eine Datei definiert:

```
-t:Dateiname.xml
```

b) Falls die Daten direkt in den Standardausgabestrom (STDOUT) geschrieben werden sollen:

```
-t:-
```

c) Falls der Zieldateiname dem Quelldateinamen (mit Ausnahme der Dateinamenserweiterung) entsprechen soll.

```
-t
```

Beispiel:

```
c2x-cmd -s:meine-datei.csv -t
```

Erzeugt eine Zieldatei namens „meine-datei.xml“

3.5.3 Option : Separators by plain text / -sep

Hier kann ein einzelner oder auch mehrere Separatoren angegeben werden. Serienmäßig ist hier das Semikolon voreingestellt.

Aufruf:

```
-sep:ZEICHEN-1[ZEICHEN-2]...[ZEICHEN-N]
```

Beispiel 1:

```
-sep:,
```

Hiermit wird als Trennzeichen das Komma festgelegt.

Beispiel 2:

```
-sep:,=
```

Hiermit werden als Trennzeichen das Komma und das Gleichheitszeichen festgelegt

3.5.4 Option : Separator by ASCII code / -sep

Möchte man einen einzelnen Separator einstellen, welcher nicht ohne weiteres als Zeichen eingegeben werden kann, so kann man dies auch per ASCII Code tun. Verwendet die zu konvertierende CSV Datei z.B. einen Tabulator als Separator, so wäre dies der ASCII Code „9“. Ein Leerzeichen entspricht dem ASCII Wert „32“. Die Unterscheidung zwischen Text und Ascii Separator erfolgt durch die Einbettung in zwei Doppelpunkte.

Aufruf:

```
-sep::ASCII-WERT:
```

Eine Liste der ASCII Zeichen können Sie hier entnehmen:
http://de.wikipedia.org/wiki/ASCII-Zeichensatz#ASCII-Tabelle

Beispiel 1:

```
-sep::9:
```

Hiermit wird als Trennzeichen das Tabulatorzeichen festgelegt.

Beispiel 2:

```
-sep::32:
```

Hiermit wird als Trennzeichen das Leerzeichen festgelegt.

3.5.5 Option : Encoding / -enc

Das Encoding legt die Zeichenkodierung der Zieldatei fest.
Beim Encoding können zwei Werte eingestellt werden:

- ISO-8859-1
  Dies entspricht nahezu dem von Windows verwendeten „1252“er Zeichensatz. Hierbei entspricht jedes Byte genau einem Zeichen.
- UTF-8
  Dies entspricht dem serienmäßig unter Linux verwendeten Zeichensatz. Hierbei werden gewisse Sonderzeichen (ab ASCII 128) durch eine Folge aus zwei Bytes ausgedrückt.

Die meisten Programme, welche UTF-8 Dateien lesen oder schreiben, fügen als erste Bytes der Zieldatei den sogenannten BOM (Byte-Order-Marker) hinzu. c2x-cmd kann diesen BOM erkennen und würde im Falle einer abweichenden „Encoding“-Einstellung eine Konvertierung von der Eingangs- in die Ausgangszeichenkodierung vornehmen.

Aufruf:

```
-enc:ISO-8859-1|UTF-8
```

Beispiel 1:

```
-enc:ISO-8859-1
```

Kodierung wird auf ISO-8859-1 eingestellt.

Beispiel 2:

```
-enc:UTF-8
```

Kodierung wird auf UTF-8 eingestellt.

3.5.6 Option : Convert command line parameters / -cc

Sollen die Parameter der Kommandozeile (bzw. Konfigurationsdatei) in eine andere Zeichenkodierung konvertiert werden, so muss dieser Schalter aktiviert werden. Wenn Sie z.B. unter Windows ein Encoding UTF-8 ausgewählt haben und die Alias-Parameter Umlaute o.ä. enthalten, sollten Sie diesen Schalter aktivieren.

Aufruf:

Es werden zwei Richtungen unterstützt:

```
-cc:u2i
```

Entspricht der Konvertierung von UTF-8 zu ISO-8859-1 und mit

```
-cc:i2u
```

wird eine Konvertierung von ISO-8859-1 zu UTF-8 vorgenommen.

3.5.7 Option : XML Mode / -m

Es gibt verschiedene Möglichkeiten eine CSV Datei in eine XML zu konvertieren. 5 Arten unterstützt c2x-cmd serienmäßig. Um die verschiedenen Arten besser beschreiben zu können, wird hier von folgenden CSV Beispieldaten ausgegangen:

```
Name;Type;Year-1987;Year-1995;Year-2001;Year-2006
Berlin;Cnty;BE;3.260.000;3.471.418;3.388.434;3.404.037
```

Die Konvertierung wurde jeweils mit den Standardeinstellungen vorgenommen.

Hinweis:
Die Bezeichner mit einem führenden kleinen Buchstaben (wie *eRecord*) weisen darauf hin, dass dieser Bezeichner umbenannt werden kann. Deren Standardwerte stehen dahinter in eckigen Klammern. Beispiel: *eRecord [record]*. Mehr dazu im Abschnitt 3.5.14.

3.5.7.1 Modus 1 : <*eRecord*><COLname>CELLvalue</ColName></*eRecord*>

Hinweis :
Dieser Modus wurde bisher als „Elements with Element Content (default)“ bezeichnet.

Aufruf:

```
-m:1
```

Erklärung:
Innerhalb eines Datensatzes („record“) existiert pro Parametername ein Element mit dem Parameterbezeichner in dessen Inhalt der Parameterwert eingebettet ist.

```
    <record>
        <Name>Berlin</Name>
        <Type>Cnty</Type>
        <Year-1987>BE</Year-1987>
        <Year-1995>3.260.000</Year-1995>
        <Year-2001>3.471.418</Year-2001>
        <Year-2006>3.388.434</Year-2006>
    </record>
```

3.5.7.2 Modus 2 : <*eRecord*><COLname *aValue*="CELLvalue"/></*eRecord*>

Hinweis :
Dieser Modus wurde bisher als „Elements with Attributes Values“ bezeichnet.

Aufruf:

```
-m:2
```

Erklärung:
Innerhalb eines Datensatzes („record“) existiert pro Parametername ein Element mit dem Parameterbezeichner und ein Attribut namens *aValue [value]*. Der Wert des Attributes ist der Parameterwert.

```
    <record>
        <Name value="Berlin" />
```

```
        <Type value="Cnty" />
        <Year-1987 value="BE" />
        <Year-1995 value="3.260.000" />
        <Year-2001 value="3.471.418" />
        <Year-2006 value="3.388.434" />
    </record>
```

3.5.7.3 Modus 3 : <*eRecord*><*eItem aName*="COLname">CELLvalue</*eItem*></*eRecord*>

Hinweis :
Dieser Modus wurde bisher als „Attributes with Element Content“ bezeichnet.

Aufruf:

```
-m:3
```

Erklärung:
Innerhalb eines Datensatzes („record“) existiert pro Parametername ein Element mit dem Parameterbezeichner *eItem [item]* und ein Attribut namens *aName [name]*. Der Inhalt des Elements ist der Parameterwert.

```
<record>
      <item name="Name">Berlin</item>
      <item name="Type">Cnty</item>
      <item name="Year-1987">BE</item>
      <item name="Year-1995">3.260.000</item>
      <item name="Year-2001">3.471.418</item>
      <item name="Year-2006">3.388.434</item>
 </record>
```

3.5.7.4 Modus 4 : <*eRecord*><*eItem aName*="COLname" *aValue*="CELLvalue"/></*eRecord*>

Hinweis :
Dieser Modus wurde bisher als „Attributes with Attribute Values“ bezeichnet.

Aufruf:

```
-m:4
```

Erklärung:
Innerhalb eines Datensatzes („record“) existiert pro Parametername ein Element mit dem Parameterbezeichner *eItem [item]* und ein Attribut namens *aName [name]*. Der Wert des Parameters ist in einem weiteren Attribut *aValue [value]* gespeichert.

```
<record>
      <item name="Name" value="Berlin" />
      <item name="Type" value="Cnty" />
      <item name="Year-1987" value="BE" />
      <item name="Year-1995" value="3.260.000" />
      <item name="Year-2001" value="3.471.418" />
      <item name="Year-2006" value="3.388.434" />
</record>
```

3.5.7.5 Modus 5 : <*eRecord* COLname="CELLvalue"/>

Hinweis :
Dieser Modus wurde bisher als „Attributes with Values“ bezeichnet.

Aufruf:

```
-m:5
```

Erklärung:
Dies ist die kompakteste Darstellung aller Modus. Innerhalb eines Datensatzes („record“) existiert pro Parametername ein Attribut. Der Wert des Parameters ist in dem Attributswert gespeichert.

```
    <record Name="Berlin" Type="Cnty" Year-1987="BE" Year-1995="3.260.000"
Year-2001="3.471.418" Year-2006="3.388.434" />
```

3.5.8 Option : suppress empty elements / -sup

Ist ein Parameterwert leer, so wird bei gesetzter Option die Ausgabe des entsprechenden XML Elementes (oder Attributes) unterdrückt.
Quelldaten (die Werte von „Type“ & „Year-1987“ sind leer):

```
Name;Type;Year-1987;Year-1995;Year-2001;Year-2006
Berlin;;;3.260.000;3.471.418;3.388.434;3.404.037
```

Aufruf:

```
-sup
```

Resultat : Ausschnitt der Zieldatei im Modus 5 mit deaktiviertem Schalter:

```
<record Name="Berlin" Type="" Year-1987="" Year-1995="3.260.000" Year-
2001="3.471.418" Year-2006="3.388.434" />
```

Resultat : Ausschnitt der Zieldatei im Modus 5 mit aktiviertem Schalter:

```
<record Name="Berlin" Year-1995="3.260.000" Year-2001="3.471.418" Year-
2006="3.388.434" />
```

3.5.9 Option : write „num“ attribute to record-element / -wn

Ist dieser Schalter gesetzt, so erhält jeder Datensatz dessen Nummer als Attribut. Die Nummerierung beginnt mit „1“.

Aufruf:

```
-wn
```

Resultat : Ausschnitt bei nicht gesetztem Schalter:

```
<record>
...
</record>
```

Resultat : Ausschnitt bei gesetztem Schalter:

```
<record num="1">
```

```
...
</record>
```

3.5.10 Option : write xml:id="id." attribute to record-element / -wid

Ist dieser Schalter gesetzt, so erhält jeder Datensatz dessen Nummer als XML-ID-Attribut. Die Nummerierung beginnt mit „1“.

Aufruf:

```
-wid
```

Resultat : Ausschnitt bei nicht gesetztem Schalter:

```
<record>
...
</record>
```

Resultat : Ausschnitt bei gesetztem Schalter:

```
<record xml:id="id.1">
...
</record>
```

3.5.11 Option : Stylesheet(s) / -styles

Mit der Stylesheet(s) Option ist es möglich mehrere Stylesheet-Anweisungen in den Kopf der XML Datei einzubetten. Es handelt sich hierbei also um eine reine Einbettung – es wird keine Transformation oder ähnliches durchgeführt.

Es können zwei Arten von Stylesheetklassen eingebunden werden:

- CSS - Cascading Stylesheets
- XSL – XML Stylesheets

Während CSS Dateien zur reinen optischen Aufbereitung genutzt werden, ist man mit XSL(T) Dateien in der Lage die einbindende XML Datei durch die Transformation in ein anderes Format umzuwandeln. Ein Stylesheet muss man als Standard einbinden und weitere als alternative Stylesheets kennzeichnen.

Mehr zu CSS Dateien unter: http://de.wikipedia.org/wiki/Cascading_Style_Sheets
Mehr zu XSL Dateien unter: http://de.wikipedia.org/wiki/Extensible_Stylesheet_Language

Aufruf:

```
-styles:STYLESHEET
-styles:STYLESHEET,STYLESHEET,...
-styles:STYLESHEET,[STYLESHEET],...
```

Bei der -styles Option handelt es sich um eine additive Option (siehe 3.2ff). Die verschiedenen Stylesheets können also durch Kommata getrennt oder durch weitere Aufrufe angegeben werden. Befindet sich der Name des Stylesheets in eckigen Klammern, so wird hier durch der Wert von „Alternative“ auf „yes“ gesetzt.

Beispiel 1:

```
-styles:.\extend\mode5-stylesheet.css
```

Resultat in der XML Datei:

```
<?xml-stylesheet type="text/css" href="./extend/mode5-stylesheet.css"
title="mode5-stylesheet" ?>
```

Beispiel 2:

```
-styles:.\extend\mode5-stylesheet.css,[.\extend\mode1-stylesheet.xsl]
```

Resultat in der XML Datei:

```
<?xml-stylesheet type="text/css" href=".\extend\mode5-stylesheet.css"
title="mode5-stylesheet" ?>
<?xml-stylesheet type="text/xsl" href=".\extend\mode1-stylesheet.xsl"
alternate="yes"  title="mode1-stylesheet" ?>
```

3.5.12 Option : Validation / -xsd & -dtd

Hier kann wahlweise eine Referenz zu einer Document-Type-Definition (DTD) oder zu einer XML-Schema-Definition (XSD) eingebunden werden. Ferner kann bei einer XSD Validierung ein Namensraum (targetNamespace) angegeben werden.

Aufruf:

Einbettung einer XSD Datei:

```
-xsd:xsd-dateiname.xsd [-xmlns:targetNamespace]
```

Einbettung einer DTD Datei:

```
-dtd:dtd-dateiname.dtd
```

Beispiel:

```
-xsd:.\extend\mode3-schema.xsd -xmlns:mode3-schema.xsd
```

Resultat in der XML Datei:

```
...
<csv_data_records xsi:schemaLocation="mode3-schema.xsd .\extend\mode3-
schema.xsd"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xmlns="mode3-schema.xsd"
...>
```

3.5.13 Option : Use Namespace / -xmlns

Unter dieser Option kann ein Namespace Präfix und ein Namespace URI eingegeben werden. Der Präfix ist dabei optional.

Aufruf:

```
-xmlns:[PREFIX=]URI
```

Beispiel 1:

```
-xmlns:MyURI
```

Resultat in der XML Datei:

```
...
<csv_data_records xmlns="MyURI"
...
```

Beispiel 2:

```
-xmlns:MyPrefix=MyURI
```

Resultat in der XML Datei:

```
...
<MyPrefix:csv_data_records xmlns:MyPrefix="MyURI">
     <MyPrefix:record>
          <MyPrefix:Name>Aachen</MyPrefix:Name>
...
```

3.5.14 Option : Alias / -alias

Möchten Sie die Elementnamen oder Attributnamen in der Ausgabedatei umbenennen, so können Sie dies mit der Alias-Option tun. Sollte dieser Name nicht XML-konform sein, so wird dieser entsprechend umgeformt.

Aufruf:

```
-alias:Name=Alias[,Name2=Alias2,...]
```

In der folgenden Tabelle sind alle derzeit änderbaren Parameter (Name) und deren Standardwerte (Default) aufgelistet.

To change the label of a parameter enter your prefered name in the "Alias" field.

Name	Type	Default	Alias
eRoot	element	csv_data_records	WurzelElement
eRecord	element	record	Datensatz
eItem	element	item	
aName	attribute	name	
aValue	attribute	value	
aSource	attribute	source	Quelle
aDT	attribute	datetime	Zeitstempel
aID	attribute	xml:id	
aNum	attribute	num	

Beispiel:

```
-alias:eRoot=WurzelElement,eRecord=Datensatz,aSource=Quelle,aDT=Zeitstempel
```

Resultat in der XML Datei:

```
...
<WurzelElement Quelle="example1.csv" Zeitstempel="2007-12-16T19:54:22+01:00" >
     <Datensatz>
      ...
```

3.5.15 Option : Versionsausgabe / -V

Die aktuelle Version von c2x-cmd kann man mit dieser Option ausgeben.

Aufruf:

```
-V
```

Ausgabe:

```
C2X-CMD V1.8.0 - Freeware by Jens Goedeke
Commandline version (c) 2002-2007
Homepage : http://www.goedeke.net
```

3.5.16 Option : Hilfe / -?

Möchte man sich eine Hilfeseite zum Aufruf ausgeben lassen, so kann man dies durch diese Option.

Aufruf:

```
-?
```

Ausgabe:

```
Usage:
c2x-cmd    -s:FILE [-t[:FILE]] [-v] [-wn] [-wid] [-sep:SEPARATORS|-sep::#:]
           [-m:#] [-sup] [-e:ENCODING] [-cc:i2u|-cc:u2i] [-xmlns:[NSPFX=]NSURI]
           [-dtd:FILE|-xsd:FILE] [-styles:FILE[,...]] [-alias:TOKEN=NAME[,...]]
           [-cfg:FILE] [-V] [-?]

...
```

3.6 Weitere Beispiele

Im entpackten c2x-cmd Paket befindet sich ein Unterververzeichnis mit diversen Beispielen. Diese können durch Aufruf der Datei „example.bat“ bzw. „example.sh“ ausgeführt werden. Als Quelle dient dabei das Verzeichnis „source“, zusätzliche Dateien werden aus „extend“ mit eingebunden und die Zieldateien werden im Verzeichnis „results“ abgelegt.

4 Anhang

4.1 Verweise

Verweis	URL
c2x-wizard Handbuch	http://www.jens-goedeke.de/c2x-cmd/c2x-wizard-handbuch.pdf
c2x-cmd Handbuch	http://www.jens-goedeke.de/c2x-cmd/c2x-cmd-handbuch.pdf