Technische Ergänzungen zum capella-2008-Benutzerhandbuch

Austausch von Partituren mit älteren capella-Versionen

Seit *capella* 2000 (Version 3.0) ist das *capella*-Format kompatibel geblieben. Sie können also alle Partituren aus Version 3.0 bis 5.3 problemlos öffnen. Auch umgekehrt können Anwender von *capella* ab Version 3.0 Ihre mit *capella* 2008 (Version 6.0) geschriebenen Partituren öffnen. Dabei gehen allerdings Formate verloren, die in den alten Versionen noch unbekannt waren.

Übernehmen von Partituren aus *capella* 2.x

Auch mit *capella* 2.x geschriebene Partituren können Sie in capella 2008 öffnen.

Im Prinzip werden *capella* 2.x-Partituren von *capella* 6.0 ohne Probleme akzeptiert. Wenn Sie allerdings in *capella* 2.x mit Tricks gearbeitet haben, sollten Sie die folgenden Anregungen zur Optimierung des Erscheinungsbildes und des logischen Aufbaus durchlesen:

Verankerung von Grafikobjekten: Wenn Sie in *capella* 2.x ein ausgedehntes Grafikobjekt (z. B. Voltenklammer) nur an der ersten Note verankert haben, kann seine Breite sich nicht an die Lage der Noten anpassen. Da die Notenabstände seit *capella* 3 etwas verändert wurden, wird dann das Ende des Grafikobjekts nicht mehr die gewünschte Position haben. In solchen Fällen müssen Sie nachträglich korrigieren. Bei dieser Gelegenheit sollten Sie auch gleich den Bezug des Grafikobjekts logisch richtig setzen: Markieren Sie das Grafikobjekt und ziehen Sie den rechten Verankerungspunkt an die Note, die dem Bereichsende entspricht.

Randausgleich. *capella* 6 begrenzt den Randausgleich in der Standardeinstellung so, dass hässliche Notenüberlappungen verhindert werden. In *capella* 2.x konnten beliebig breite Systeme durch den Randausgleich komprimiert werden. Wenn Sie nach dem Öffnen einer solchen Partitur zu breite Systeme sehen, können Sie entweder diese Systeme teilen oder die Komprimierung erzwingen mit Format – Partitur – Ausrichtung. Setzen Sie hier ein Häkchen bei *Kompression erlauben*.

> *capella* 6 ignoriert den Randausgleich auch dann, wenn ohne Randausgleich weniger als ein Viertel der Breite gefüllt ist.

Verschobene Noten: Wenn Sie in *capella* 2.x Noten horizontal verschoben haben, weil die Abstände nicht harmonisch aussahen (z. B. zu wenig Platz vor einem Versetzungszeichen), kann es sein, dass Sie diese Verschiebung in *capella* 6 besser wieder aufheben sollten, weil *capella* 6 eine verbesserte Abstandsberechnung verwendet. Unter Umstän-

den empfiehlt es sich, probeweise die gesamte Partitur zu markieren und (mit Format – Noten/Pausen – Allgemein) die Korrektur der horizontalen Lage auf Null zu setzen.

Mehrstimmige Notenzeilen. In *capella* 2.x waren maximal zwei Stimmen in einer Notenzeile möglich. Wenn Sie mehrere Notenzeilen mit Nullabständen übereinandergelegt haben, um diese Begrenzung zu umgehen, wird dies zwar in *capella* 6 unverändert angezeigt, führt aber in der Mustersystemansicht zu nur halb sichtbaren Notenzeilen. Wenn Sie solche Partituren in *capella* 6 weiter bearbeiten möchten, sollten Sie auf echte mehrstimmige Notenzeilen umsteigen.

Wenn Sie **Partituren mit sehr vielen Spezialtricks** in *capella* 2.x geschrieben haben, lohnt sich eine Anpassung an *capella* 6 nur dann, wenn Sie die Partitur weiter bearbeiten möchten. Sonst drucken Sie solche Partituren einfach weiter mit *capella* 2.x.

Übernehmen von Partituren aus *capella* 1.x

Dateien, die mit *capella* 1.x geschrieben wurden (Erweiterung `.all`), können mit *capella* 6 nicht gelesen werden. Diese Dateien müssen zunächst mit *capella* 2.x (oder *capella* 1000) ins *capella*-2-Format konvertiert werden.

Eigene Schriftarten für *capella* herstellen

Um eine eigene Schriftart herzustellen, benötigen Sie ein Programm, das Fonts bearbeiten kann. Zum Beispiel können Sie das weit verbreitete Programm *CorelDraw* (in beliebiger Version) verwenden. Die folgende Kurzanleitung wendet sich an Experten:

1. Machen Sie eine Kopie der Datei `capella3.ttf`, z. B. unter dem Namen `meinfont.ttf`. Die Kopie hat nun zwar einen anderen Dateinamen, der Schriftartname (er ist innerhalb der Datei vermerkt) ist aber immer noch *capella3*. Er muss geändert werden, damit die neue Schriftart gleichzeitig mit der alten verwendet werden kann. Das werden Sie im zweiten Schritt erledigen.

2. In *CorelDraw* gehen Sie für jedes Zeichen, das Sie ändern wollen (z. B. Notenköpfe, Schlüssel) so vor: Markieren Sie das Textwerkzeug, bewegen Sie das Mausfadenkreuz genau in den Nullpunkt (Lineale beachten!), klicken Sie und schreiben Sie das Zeichen (z. B. ein A für den G-Schlüssel). Markieren Sie das Zeichen und wählen Sie die Schriftart *capella3* und die Schriftgröße 720. Wandeln Sie nun das Zeichen in Kurven um und ändern Sie es nach Wunsch. Am besten richten Sie sich noch passende Gitterlinien ein: Bei *CorelDraw* 11 wählen Sie Extras → Optionen → Dokument → Gitter und geben für die Rasterweite ein: horizontal 0, vertikal 0,8 pro Zoll. Nun sehen Sie waagerechte Hilfslinien im Abstand von 1,25 Zoll. Das entspricht genau dem halben Notenlinienabstand in *capella*. Ein normaler Notenkopf sollte also symmetrisch auf der waagerechten Nulllinie liegen und die beiden Nachbarlinien berühren.
   Wenn Ihr Symbol fertig ist, markieren Sie es und exportieren es in die neue Schriftart (Datei → Exportieren → Dateityp: *TrueType*). Mit OK kommen Sie in einen zweiten Dialog, in dem Sie den richtigen Zeichencode (gleicher Buchstabe wie

vorher) und die Designgröße 720 angeben müssen. *Wichtig*: Beim ersten Zeichen, das Sie exportieren, klicken sie auch auf Optionen und geben unter Familienname z. B. „Meine Schriftart“ ein.

Wegen eines Programmfehlers in CorelDraw hat Ihr neuer Font höchstwahrscheinlich eine falsche Prüfsumme. *capella* stört sich nicht daran, wohl aber manche Schriftverwaltungsprogramme. Sie können die Prüfsumme z. B. mit dem Sharewareprogramm „Softy“ (`http://users.breathe.com/l-emmett/`) korrigieren.

Einen neuen Stil einrichten

Auf der Registerkarte Extras → Optionen → Allgemein finden Sie das Aufklappfeld *Stil der Musiksymbole*. Welche Einträge im Aufklappfeld angeboten werden, hängt von den Einträgen im Abschnitt `[Styles]` der Datei `capella.dat` ab.

Enthält `capella.dat` z. B. den folgenden Abschnitt

```
[Styles]
N=2
1=capella-Standard
2=Barock
```

So werden die beiden Einträge „*capella*-Standard“ und „Barock“ in der Aufklappbox angeboten.

Um einen neuen Stil einzufügen, gehen Sie so vor:

1. Schreiben Sie ans Ende des Abschnitts einen neuen Stilnamen Ihrer Wahl.
2. Ändern Sie die Zahl hinter N= auf die neue Zahl der Einträge.
   Beispiel:
   ```
   [Styles]
   N=3
   1=capella-Standard
   2=Barock
   3=Mein Spezialstil
   ```
3. Zu jedem Stilnamen in der Liste muss ein eigener Abschnitt des gleichen Namens vorhanden sein.
   Beispiel:
   ```
   [Mein Spezialstil]
   FontFile=capella3
   FontName=capella3
   NoteSpaces=64,30,43
   HeadWidth=512,384,307,307
   LineWidth=24,21,21
   xStemDown=0,0
   yStemDown=-43,-43
   xStemUp=307,307
   yStemUp=43,43
   Bracket=-16,-16,8,1,-24,-16,8,-16,30,-32,40,-48,30,-32,8,-4,-24,-4
   ```

Stilabschnitte

Die einzelnen Einträge eines Stilabschnitts haben folgende Bedeutung:

FontFile, FontName. Hier können Sie den Dateinamen und den Schriftartnamen eines alternativen Zeichensatzes angeben, der sich im *capella*-Programmordner befinden muss. Der Dateiname muss ohne die Erweiterung `.ttf` angegeben werden. Den Schriftartnamen erfahren Sie, wenn Sie z. B. im Windows-Explorer auf den Dateinamen doppelklicken. Dann öffnet sich ein Fenster mit Angaben über die Schriftart. Bei `capella3.ttf` lesen Sie dann z. B. „Schriftart: capella3“. Der Schriftartname kann sich aber auch vom Dateinamen unterscheiden!
Wenn Sie viele alternative Schriftarten verwenden, brauchen Sie diese nicht alle permanent zu installieren. Nicht installierte Schriften installiert *capella* temporär und gibt sie bei Beendigung des Programms wieder frei.

NoteSpaces. Hier geben Sie drei Zahlen (Maßeinheit: 1/256 Zwischenraum) ein für (1) den Abstand zwischen Schlüssel (Tonart, Takt) am Anfang des Systems und der ersten Note, (2) den Freiraum am Anfang und (3) am Ende eines Taktes.

HeadWidth. Breite der Notenköpfe (1) Brevis, (2) Ganze, (3) Halbe und (4) Viertel (Einheit: 1/256 Zw.).

LegerLineJutOut. Gibt an, wie weit die Hilfslinien aus den Notenköpfen herausragen (Einheit: 1/32 Zw.). Wenn dieser Eintrag fehlt, wird der Standardwert 10 verwendet.

LineWidth. Stärke der (1) Hälse, (2) Notenlinien und (3) Taktstriche (Einheit: 1/256 Zw.).

xStemDown. Horizontale Lage eines nach unten gerichteten Notenhalses vom linken Anfang des Kopfes aus gemessen für (1) Halbe- und (2) Viertelköpfe (Einheit: 1/256 Zw.). Das Maß bezieht sich auf die linke Kante des Halses.

yStemDown. Vertikale Lage des Halsansatzes eines nach unten gerichteten Notenhalses für (1) Halbe- und (2) Viertelköpfe (Einheit: 1/256 Zw.). Beispiel: Bei `capella3.ttf` muss wegen der schrägen Ovalform der Köpfe der Hals etwas unterhalb der Mitte des Notenkopfes beginnen. Daher ist der Wert –43.

xStemUp. Horizontale Lage eines nach oben gerichteten Notenhalses vom linken Anfang des Kopfes aus gemessen für (1) Halbe- und (2) Viertelköpfe (Einheit: 1/256 Zw.). Das Maß bezieht sich auf die rechte Kante des Halses.

yStemUp Vertikale Lage des Halsansatzes eines nach oben gerichteten Notenhalses (vgl. yStemDown).

Bracket. Die eckigen Klammern (Balkenklammern), mit denen mehrere Notenzeilen eines Systems zusammengefasst werden, bestehen normalerweise aus einem senkrechten Balken, der an beiden Enden sichelförmig abgeknickt ist.

Balkenklammern

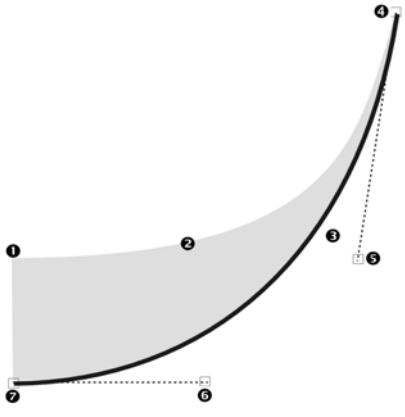
Um genau die bekannte Balkenklammer mit zwei Sicheln zu bekommen, können Sie die folgende Zeile in eine Stilbeschreibung einfügen:

Bracket=-16,-16,8,1,-24,-16,8,-16,30,-32,40,-48,30,-32,8,-4,-24,-4

Da die Standarddaten automatisch gesetzt werden, brauchen Sie diese Zeile aber nur, wenn Sie die Werte ändern möchten. Die Werte bestimmen das Rechteck für den senkrechten Balken und die genaue Form der Sicheln. (Bei nach oben weisender y-Achse wird durch die Daten die untere Sichel, bei nach unten weisender y-Achse die untere Sichel beschrieben, y=0 entspricht jeweils der äußersten Notenlinie). Die beiden Bögen des Sichelrandes werden (wie Bindebögen) durch Anfangs- und Endpunkt sowie zwei Stützpunkte für die Krümmung bestimmt:

Wert	**Standard**	**Bedeutung (alle Maße in 1/32 Zwischenräumen)**
1./2.	-16, -16	Rechte obere Ecke des vertikalen Klammerbalkens (x- und y-Koordinate)
3.	8	Breite des Klammerbalkens
4.	1	1 = mit, 0 = ohne Sicheln (in diesem Fall können 5. bis 18. entfallen)
5./6.	-24, -16	Obere Ecke der oberen Sichel (x- und y-Koordinate, 1)
7./8	8, -16	Erster Stützpunkt des oberen Bogens der oberen Sichel (2)
9./10.	30, -32	Zweiter Stützpunkt des oberen Bogens der oberen Sichel (3)
11./12.	40, -48	Spitze der oberen Sichel (4)
13./14.	30, -32	Erster Stützpunkt des unteren Bogens der oberen Sichel (5)
15./16.	8, -4	Zweiter Stützpunkt des unteren Bogens der oberen Sichel (6)
17./18.	-24, -4	Untere Ecke der oberen Sichel (7)

Die nebenstehende Abbildung zeigt eine obere Sichel (mit hervorgehobenem unteren Bogen). Die Nummern der Ecken und Stützpunkte sind in der Tabelle in Klammern angegeben.



Musiksymbol-Paletten anpassen

Für Experten: Paletten anpassen. Die Paletten, die Sie im Symbol-Dialog sehen, sind in der Datei `data\symbols.dat` im capella-Programmordner definiert. Die Datei enthält eine Tabelle, in deren linker Spalte die Überschriften der einzelnen Registerkarten stehen. In den 20 weiteren Spalten stehen die Zeichencodes der auf der jeweiligen Karte angezeigten *capella*-Zeichen.
Wenn Sie wollen, können Sie mit einem ASCII-Editor die Paletten für Ihre Bedürfnisse maßschneidern. Die Zeichencodes finden Sie im Anhang dieses Handbuchs. Speichern Sie eine Kopie der Datei im Ordner `config\data\symbols.dat` Ihres persönlichen *capella*-Ordners und passen Sie diese an.

PostScript-Export

PostScript ist eine weit verbreitete Seitenbeschreibungssprache. Um mit PostScript beschriebene Seiten zu drucken, wird ein PostScript-Interpreter benötigt.

Sie können Ihre Partituren

- direkt auf einen PostScript-Drucker ausgeben,
- einen PostScript-Drucker wählen, auf Datei ausdrucken und die Datei an ein Satzstudio schicken oder
- einen PostScript-Drucker wählen, auf Datei ausdrucken und die Datei mit Hilfe eines PostScript-Emulators auf Ihrem nicht PostScript-fähigen Drucker ausgeben.

Im zweiten Fall sollten Sie sich mit dem Satzstudio absprechen, welchen Druckertreiber Sie verwenden. Im dritten Fall können Sie einen beliebigen PostScript-Druckertreiber wählen. PostScript ist ein komplexes Thema. Deshalb kann dieses Handbuch eine PostScript-Einführung nicht ersetzen. Die folgende exemplarische Anleitung kann Ihnen aber vielleicht eine Starthilfe geben.

Beispiel für die PostScript-Ausgabe auf Ihrem Drucker

1. Installieren Sie *GhostScript* und die Programmoberfläche *GSview*. Unter `http://www.cs.wisc.edu/~ghost/` können Sie diese Programme kostenlos herunterladen.
2. Installieren Sie den PostScript-Druckertreiber *Linotronic 630* (Systemsteuerung: Drucker → Neuer Drucker). Dieser Treiber hat sich in der Praxis gut bewährt. Wählen Sie bei Anschluss „FILE“.
3. Klicken Sie mit der rechten Maustaste auf das neue Druckersymbol, und wählen Sie im Kontextmenü Eigenschaften und im erscheinenden Dialog die Registerkarte PostScript. Wählen Sie dort unter PostScript-Ausgabeformat „PostScript (Optimale Portierung - ADSC)“.
4. Wählen Sie in *capella* unter Datei → Druckereinrichtung den Drucker *Linotronic 630*.

5. Starten Sie den „Druck“ wie gewohnt. Sie werden nach dem Namen der Druckdatei gefragt. Ändern Sie die vorgeschlagene Dateierweiterung von „`.prn`“ in „`.ps`“ um.
6. Klicken Sie nun (z. B. im Windows-Explorer) auf den Namen der soeben entstandenen PostScript-Datei. GSview öffnet sich und zeigt eine Druckvorschau.
7. Wählen Sie File → Print und machen Sie die nötigen Angaben (ziehen Sie ggf. die Dokumentation von GhostScript und GSview zu Rate).

Verkleinerte Ausgabe. Auf der Registerkarte Grafik der Druckereigenschaften des PostScript-Druckers ist bei Skalierung 100% voreingestellt. Wenn Sie z. B. von A4 auf A5 verkleinern möchten, wählen Sie dort 71%.

Informationen für Skript-Entwickler

Hinweis zur Python-Installation

Hinweis zur Python-Installation: Mit *capella* wird Python nicht komplett installiert. Wenn Sie ein Python-Skript unabhängig von *capella* aufrufen möchten, müssen Sie die offizielle Python-Distribution installieren. Sie haben dabei zwei Möglichkeiten:
(a) Sie installieren die aktuellste Python-Distribution. Diese darf nicht in den *capella*-Programmordner installiert werden! (b) Sie installieren Python-2.3.5. Wenn Sie Platz auf der Platte sparen wollen, können sie als Installationsziel den `Python23`-Ordner im *capella*-Programmordner wählen.

Hinweis für Entwickler

Wenn Sie in einem Skript, das unabhängig von *capella* aufgerufen wird, die *caplib* verwenden wollen, fügen Sie folgende Zeilen ein:

```
import os, sys, _winreg
r = _winreg.OpenKey(_winreg.HKEY_LOCAL_MACHINE,
                    r'SOFTWARE\capella-software\capella\5.0')
capPath = os.path.split(_winreg.QueryValueEx(r, 'ExePath')[0])[0]
sys.path.append(os.path.join(capPath, 'py-ext'))
import caplib
```

Plugins

Skripte, die besonders häufig benötigt werden, können in das Plugin-Menü und in die Plugin-Symbolleiste integriert werden. Wenn Sie das vorgegebene Plugin-Menü oder die Belegung der Plugin-Symbolleiste ändern wollen, müssen Sie die Datei `data\plugins.dat` im *capella*-Programmordner anpassen. Sie könnte z. B: so aussehen:

```
<group>
  <item   name="Stimmenliste"            file= "Demos\voiceList.py" />
  <item   name="Spiel_dauer"  symbol="7" file= "Spieldauer.py" />
  <separator/>
  <group  name="_Taktnummerierung">
    <item name="_Einfügen"    symbol="1" file= "Taktnummern.py" />
    <item name="_Löschen"     symbol="2" file= "Taktnummern_entfernen.py" />
  </group>
</group>
```

Diese Datei entspricht den XML-Formatkonventionen. Das Beispiel zeigt alles, was Sie zur Bearbeitung wissen müssen:

- Zeile 1, 9: Die erste und die letzte Zeile müssen immer so aussehen. Sie umschließen das Hauptelement `group`.
- Zeile 2: Jeder Menübefehl wird durch eine Element `item` definiert. Der Wert des Attributs `name` (hier „Stimmenliste“) gibt an, wie der Befehl im Menü heißt. Mit dem Attribute `file` wird die Python-Datei des Skripts relativ zum Skript-Ordner angegeben.
- Zeile 3: Optional kann zusätzlich mit dem Attribut `symbol` eine Nummer (1 bis 8) eines Symbols für dieses Skript angeben werden.
  Das Skript kann dann durch Mausklick auf das entsprechende Symbol gestartet werden oder mit der Tastatur, indem bei gedrückten Tasten **[Shift]**+**[Alt]** zusätzlich die entsprechende Zifferntaste gedrückt wird.
- Zeile 3: Mit einem Unterstreichungsstrich (_) im Attribut `name` wird festgelegt, dass der dahinter stehende Buchstabe im Menü unterstrichen wird.
- Zeile 4: Mit `separator`-Elementen können Trennstriche ins Menü eingefügt werden.
- Zeile 5, 8: Um mehrere Befehle in ein Untermenü zu gruppieren, werden diese (hier die beiden in Zeile 6 und 7) in ein `group`-Element eingeschlossen.
  Auf diese Weise können Untermenüs auch beliebig verschachtelt werden!
- Die Einrückung der Zeilen und die Lücken zwischen den einzelnen Elementen und Attributen haben (XML-gemäß) für *capella* keine Bedeutung. Sie dienen nur zur besseren Lesbarkeit der Datei

Wenn Sie die Symbole in der Plugin-Symbolleiste ändern möchten, öffnen Sie die Datei `data\toolbars\plugins-1.png` mit einem Malprogramm und bearbeiten den entsprechenden Bereich: Für jedes der acht Symbole ist jeweils ein Rechteck der Breite 16 Pixels zuständig.

Skript-Internationalisierung

Da *capella* in mehreren Sprachen erscheint, mussten bisher auch Skripte an verschiedene Sprachen angepasst werden. Wenn dann Änderungen anfielen, konnte es leicht geschehen, dass sich die verschiedenen Sprachversionen auseinanderentwickelten.

capella 2008 bringt einen neuen Standard für die Lokalisierung von Skripten, der es erlaubt, die unterschiedlichen Sprachinformationen unabhängig von der eigentlichen Python-Datei zu pflegen.

Ein typisches Beispiel

Die Umstellung soll hier an einem Minimal-Beispiel gezeigt werden:

`demo.py`

```
# -*- coding: ISO-8859-1 -*-
""" capellaScript -- Copyright (c) 2004-2008 Hartmut Ring
>>> Test des Dateiauswahldialogs
```

```
    Dies ist eine Demo für die neuen
    Lokalisierungsmöglichkeiten in capella 2008.
<<<
"""

dlg = FileDialog()
dlg.setTitle("Test des Dateiauswahldialogs")
dlg.addFilter("capella-Dateien", "*.cap;*.capx")
if dlg.run():
    messageBox ("gewählte Datei", dlg.filePath())
```

Dieses Skript läuft natürlich unverändert auch unter *capella* 2008, aber eben nur auf deutsch. Um es sowohl für die deutsche als auch die englische *capella*-Version anzupassen, genügt es, neben dem Skript folgendes XML-Datei bereitzustellen:

`demo.info`

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<info>
  <lang id="en">
    <title>Test of the file dialog</title>
    <descr>
      <p>This is a demo of the new
        localization possibilities in capella 2008.</p>
    </descr>
  </lang>
  <lang id="de">
    <title>Test des Dateiauswahldialogs</title>
    <descr>
      <p>Dies ist eine Demo für die neuen
        Lokalisierungsmöglichkeiten in capella 2008.</p>
    </descr>
  </lang>
</info>
```

Diese Datei muss den Namen [skriptName]`.info` tragen.
Beispiel: Skript: `demo.py`, Info-Datei: `demo.info`.

Nun zeigt der Skript-Browser nicht mehr den Dateinamen an (der jetzt international einheitlich sein kann), sondern einen der in den XML-Informationen unter `<title>` angegebenen Titel in der aktuellen Landessprache. Im Beschreibungsfeld des Skript-Browsers wird der entsprechende Text aus dem Element `<descr>` gezeigt. Dieser Text muss je Absatz ein `<p>`-Element besitzen. Sie können auch leere Absätze (`<p/>`) für Leerzeilen einstreuen.

> Da frühere *capella*-Versionen die XML-Information nicht interpretieren können, zeigen sie im Browser keine Hinweise zum Skript an. Um es allen Versionen recht zu machen, können Sie die alte Information (Text von >>> bis <<<) auch stehen lassen. *capella* 2008 sucht in den ersten 4 KBytes des Skripts nach Informationen im alten Stil.

Was noch fehlt, ist die Lokalisierung der sprachabhängigen Texte, die das Skript selbst ausgibt. Dazu definiert man zunächst eine Liste, die für jede Sprache ein Paar aus Sprachkennung und einem Python-Dictionary enthält, das alle Sprach-Texte als Werte zu frei wählbaren Schlüssels enthält. Die Schlüssel müssen für alle bereitgestellten Sprachen übereinstimmen.

```
translations = [
("en", {
    "title"       : "Test of the file dialog",
    "capFiles"    : "capella files",
    "chosenFile"  : "chosen file"}),
("de", {
    "title"       : "Test des Dateiauswahldialogs",
    "capFiles"    : "capella-Dateien",
    "chosenFile"  : "gewählte Datei"})
]
```

In jedem Paar in der Liste Dabei `translations` muss links ein Sprachkürzel stehen, das gleich erklärt wird, und rechts ein Übersetzungs-Dictionary.

Diese Sprachtabellen übergibt man nun der Funktion `setLanguages`, damit das auch für ältere *capella*-Versionen klappt, am besten in einem `try`-Block:

```
try:
    setLanguages(translations)
except:
    def tr(s):
        return german[1].get(s, "???")
```

capella sucht sich nun automatisch die passendste der angebotenen Sprachen aus. Die Funktion `tr(key)` sucht zum `key` den Wert in der passenden Sprache. Es müssen also im Skript alle Sprachtexte durch die entsprechenden `tr`-Aufrufe ersetzt werden:

```
dlg = FileDialog()
dlg.setTitle(tr("title"))
dlg.addFilter(tr("capFiles"), "*.cap;*.capx")
if dlg.run():
    messageBox (tr("chosenFile"), dlg.filePath())
```

Noch eleganter ist es, nur die Originalsprache im Skript zu definieren und weitere Sprachen in ein separates Modul auszulagern. Da Python-Module die Endung .py tragen müssen und der Skript-Browser diese Module nicht anzeigen soll, muss dabei die Konvention beachtet werden, dass das Modul den Namen [SkriptName]`_tr.py` trägt. Am Ende finden Sie das Beispielskript, komplett nach diesen Richtlinien zerlegt.

Sprach-Kennungen

Die internationale Norm ISO 639-1 definiert Kürzel aus zwei Kleinbuchstaben für alle möglichen Sprachen der Welt, z. B. „`de`“ für Deutsch, „`en`“ für Englisch. Die Norm ISO 3166 definiert Kürzel aus zwei Großbuchstaben für geografische Einheiten, z. B. „DE“ für Deutschland, „US“ für die USA.

In den `<lang>`-Elementen der Skript-Header kann für das Attribut `id` entweder eine Sprachkennung (z. B. `de`) oder eine Kombination aus Sprachkennung und geografischer Einheit, getrennt durch ein Minuszeichen verwendet werden (z. B. `en-GB`):

Für die aktuellen Sprachversionen von *capella* kommen folgende Kennungen in Frage:

Code	Bedeutung

de	Deutsch
en	(Amerikanisches) Englisch Wenn Sie zwei englische Versionen anbieten, sollten Sie eine davon einfach mit „en“ bezeichnen. Bei „en“ für amerikanisches Englisch und „en-GB“ für britisches Englisch wird das amerikanische Englisch als Standard gewählt und die britische Version nur für das Vereinigte Königreich. Nennen Sie dagegen die amerikanische Version „en-US“ und die britische „en“, ist Britisch der Standard und Amerikanisch wird nur für USA gewählt.
en-GB	Britisches Englisch „GB“ steht paradoxerweise nicht für Großbritannien sondern für das Vereinigte Königreich (von Großbritannien und Nordirland). Es ist das einzige Kürzel nach ISO 3166, das nicht mit dem Länder-Domain-Code (hier: uk) übereinstimmt.
nl	Niederländisch
cs	Tschechisch
pl	Polnisch
fi	Finnisch
sv	Schwedisch

Oft genügt eine einzige englische Version (en). Nur wenn im Skript Unterschiede wie color/colour vorkommen, kann zusätzlich eine britische Version (en-GB) bereitgestellt werden.

Wie *capella* die optimale Sprache findet

Jede lokalisierte *capella*-Version kennt (aus ihren Ressourcen) ein vollständiges Sprachkürzel (z. B. de-DE).

capella sucht zunächst nach einer Lokalisierung mit dem vollständigen Sprachkürzel. So wird die deutsche Version zunächst nach de-DE suchen. Wird dies nicht gefunden (was in der Regel der Fall ist), schraubt *capella* seine Ansprüche zurück und sucht nach einer Version mit dem Sprachkürzel (de). Hier würde z. B. auch de-AT akzeptiert. Wenn auch dies erfolglos ist, sucht *capella* nach einem Kürzel, das mit en beginnt. Gibt es auch das nicht, wird die als erste angegebene Sprache genommen.

Lokalisierte Hilfedateien für Skripte

capella 5 hat zu einem Skript nach einer .chm- oder .html-Datei gleichen Namens gesucht. Wenn eine dieser beiden Dateien vorhanden war wurde der Button „Dokumentation“ im Skriptbrowser aktiv.

capella 6 sucht zunächst nach .chm- oder .html-Dateien mit dem um ein Minuszeichen und eine Sprachkennung verlängerten Namen.
Beispiel: Wenn neben dem Skript demo.py auch die Dateien demo-en.chm, demo-en-GB.chm und demo-de.chm existieren, wählt *capella* (wie oben angegeben) die

passendste Version. Nur wenn gar keine lokalisierte Version vorhanden ist, greift *capella* auf die Hilfedateien mit neutralem Namen (`demo.chm`) zurück.

Ordner-Informationen

Da in *capella* 2008 der gesamte Skriptordner sprachunabhängig ist, werden auch die Ordnernamen lokalisiert. Wenn sich in einem Skriptordner eine Datei `dirinfo.xml` befindet, zeigt der Skript-Browser nicht den Ordnernamen an, sondern den in `dirinfo.xml` angegebenen Titel in der passenden Sprache. Der Aufbau der `dirinfo.xml` ist identisch mit den XML-Informationen in einem Skript.

Das komplett internationalisierte Beispielskript

`demo.py`

```
# -*- coding: ISO-8859-1 -*-
""" capellaScript -- Copyright (c) 2004-2008 Hartmut Ring
>>> Test des Dateiauswahldialogs
    Dies ist eine Demo für die neuen
    Lokalisierungsmöglichkeiten in capella 2008.
<<<
"""

german = ("de", {
    "title"       : "Test des Dateiauswahldialogs",
    "capFiles"    : "capella-Dateien",
    "chosenFile"  : "gewählte Datei"})

try:
    from chordSymbols_tr import translations
    translations.append(german)
    setLanguages(translations)
except:
    def tr(s):
        return german[1].get(s, "???")
#---------------------------------------------------------------------

dlg = FileDialog()
dlg.setTitle(tr("title"))
dlg.addFilter(tr("capFiles"), "*.cap;*.capx")
if dlg.run():
    messageBox (tr("chosenFile"), dlg.filePath())
```

`demo_tr.py`

```
# -*- coding: ISO-8859-1 -*-
translations = [
("en", {
    "title"       : "Test of the file dialog",
    "capFiles"    : "capella files",
    "chosenFile"  : "chosen file"})
]
```

`demo.info` (Siehe oben)