ESS
Enterprise Solution Server

Programmierbeispiele

Inhaltsangabe

1. Allgemeine Hinweise

1.1. Anmelden als Ess-Benutzer

Um ein ESS-Benutzer zu werden, muss man zuerst als Benutzer 'root' angemeldet sein.

```
<benutzer_aktuell>@<rechner>:~ su -
Password:
```

Danach meldet man sich als ESS-Benutzer an:
(mx001 ist ein Beispielname für einen Benutzer)

```
<rechner>:~ # su  -  mx001
```

Beachten Sie das Minuszeichen mit Leerzeichen!

1.2. Starten des Compilermechnismus

make:

make erlaubt es, ein in dem aktuellen Verzeichnis liegendes System neu zu generieren.Die dazu notwendigen Übersetzungen, Bibliotheksersetzungen und das Binden erfolgen dabei automatisch.

Um 'make' zu starten, muss man als <benutzer> auf dem <rechner> angemeldet sein und sich im Verzeichnis ~/src befinden.

z.B. als Benutzer 'mx001':

```
mx001@<rechner>:~/src > make
```

1.3. Starten und Stoppen der Datenbank- bzw. der ESS-Prozesse

Um ESS-Prozesse zu starten bzw. zu stoppen, muss man auf dem <rechner> als 'root' angemeldet sein.

Beispiel:

```
mx001@<rechner>:~ > su -
Password:
<rechner>:~ # sh /etc/pentaprise/ess/mxrc startmxsd
<rechner>:~ # sh /etc/pentaprise/ess/mxrc stopmxsd
```

<rechner>:~ # sh etc/pentaprise/ess/mxrc stop stoppt die ESS-Prozesse und die Datenbank.
<rechner>:~ # sh etc/pentaprise/ess/mxrc stopmxsd stoppt nur die ESS-Prozesse .
<rechner>:~ # sh etc/pentaprise/ess/mxrc startdb startet nur die Datenbank.
<rechner>:~ # sh etc/pentaprise/ess/mxrc start startet die ESS-Prozesse und die Datenbank.

Durch Anhängen des Benutzernamens werden nur die Prozesse des angegebenen Benutzers gestartet bzw. Gestoppt.

1.4. Aufruf des Datenbankänderungswerkzeuges 'reltool'

'reltool` ist ein Programm, das die Differenzen zwischen unterschiedlichen Masken-Beschreibungen (die Datei 'mxmask.msk') erkennt und die Datenbank-Struktur anpasst.
Das Programm kann auf der Datenbank neue Tabellen anlegen und die existierenden entfernen, Tabellen-Felder ändern oder neue Felder hinzufügen, neue Referenzen erstellen und die alten löschen.
Das Programm kann nicht Tabellen-Felder entfernen, was bei manchen Datenbanken auch nicht möglich ist.
Um das 'reltool'-Programm zu starten, muss man sich als <benutzer> auf dem <rechner> im Verzeichnis ~/src/programs befinden.

Beispiel:

```
mx001@<rechner>:~/src/programs >
        pg_dump -U pg_mx001 -s  MXDBS  |  reltool -n mxmask.msk -sfx -O -
```

Parameter für das Programm 'reltool':

-U <DB-Benutzer>	- für postgres -> pg_mx001 (bei Standardinstallation)
-f	- erzwingen von DROP Anweisungen;
-x	- erzeugt die SQL Dateien reltool.out und reltool.const im aktuellen Verzeichniss. In der Dateien reltool.out stehen die Anweisungen zur Änderung der Datenstruktur. In reltool.const die Vorschläge zur Änderung der Daten, wie z.B. Defaultwerte;
-s	- keine Checksummenprüfung auf der Datenbank;
-c	- erzwingt den Abgleich der Checksumme;
-O <alte_SQL_Datei>	- Alte Struktur der Datenbank erzeugt mit : mkdbs -S postgres -n

Hinweis:

Vor Aufruf des Programms müssen die mxsd-Prozesse des entsprechenden Mandanten gestoppt werden! (außer bei einem Aufruf mit dem Parameter '-x')

1.5. Erzeugen einer neuen Datenbank

Mit dem Programm 'mkdbs' wird ein neues Datenbank-Schema erzeugt. Dabei wird das bestehende Datenbankschema gelöscht und alle darin gespeicherten Daten sind somit unwiederbringlich verloren!

Um das 'mkdbs'-Programm zu starten, muss man sich als <benutzer> auf dem <rechner> im Verzeichnis ~/src/programs befinden.

Beispiel:

mx001@<rechner>:~/src/programs > mkdbs -S<Datenbank> -p<DB-Passwort> -u<DB-Benutzer> -s<Standardconfiguration>

Parameter für das Programm 'mkdbs':

-m<maskfile>	Name der Maskendatei
-d<Datenbank-Name>	Datenbank-Schema, wie z.B. nach der Standardinsallation 'MXDBS', kann aber auch anders sein;
-u<DB-Benutzer>	für postgres -> pg_mx001 (bei Standardinstallation);
-p<DB-Passwort>	für postgres -> postgress (bei Standardinstallation)
-s<Standard-Konfiguration>	nach der Standardinstallation 'mxstandard.dbl', aber andere sind auch möglich;
-S<DB-Typ>	Datenbank-Typ wie z.B. POSTGRES oder YARD;
-v	es erfolgt eine Ausgabe der Vorgänge des Programms
-n	ein SQL-File mit der aktuellen DB-Struktur wird erzeugt;
-x	es erfolgt ein Update auf die Tabellen die sämtliche Masken, Felder und Formulare beinhalten;
-r	erzeugt ein DB-Schema ohne Referenzen und Indices;
-i	für eine initiale Installation der Datenbank. Hierbei wird alles innerhalb der Datenbank (z.B. Postgres, etc.) gelöscht.

Hinweis:

Vor Aufruf des Programms müssen die mxsd-Prozesse des entsprechenden Mandanten gestoppt werden!

1.6. Ausführen von Datenbankskripten

Mit Hilfe von SQL-Skripten kann man gewünschte Änderungen an einer Datenbank durchführen oder Daten in die Datenbank einlesen.

Um SQL-Skripte zu starten, muss man zuerst als 'root' angemeldet sein. Danach muss man sich als <db_benutzer> anmelden.

Beispiel 1: als DB-Benutzer 'pg_mx001' das Skript '<scriptname>.sql' starten:

```
mx001@<rechner>:~ > su -
Password:
<rechner>:~ # su - pg_mx001
pg_mx001@<rechner>:~ > psql MXDBS < ~mx001/src/<scriptname>.sql
```

psql ist ein Programm, das den Zugriff auf eine Postgres-Datenbank ermöglicht.
Andere Datenbanken haben ihre eigenen Zugriffsprogramme.

1.7. Ausführen von Datenbankabfragen

Beispiel 2: als DB-Benutzer 'pg_mx001' den Inhalt der DB-Tabelle adr abfragen:

```
mx001@<rechner>:~ > su -
Password:
<rechner>:~ # su - pg_mx001
pg_mx001@<rechner>:~ >  psql MXDBS
MXDBS =#  select * from adr;
```

Mit dem Befehl <\q> wird die DB-Kommunikation wieder beendet.
Auf diese Art lassen sich mit den verschiedenen SQL-Statements Datenbankabfragen ausführen.

1.8. Erzeugen von Datenbank-Views

Um sich den Umgang mit den Tabellen-Namen zu erleichtern, gibt es ein Programm 'views.pl' im Verzeichnis /opt/ess/mx001/src/programs.

Mit dem Aufruf (als Benutzer mx001)

```
mx001@<rechner>:~ > perl views.pl
```

werden zwei Dateien erzeugt.

1. **masken.txt:** enthält eine Übersicht über alle Masken und Felder mit ihren Tabellen- und Spaltennamen sowie ihren Masken- und Felt-Titeln, -Typen und Referenzen;

/* Ausschnitt Datei masken.txt */

Maske: Adressen, Name: ADR, Tabelle: ADR

Titel	Spalte	Typ	Reference
	LFD	INTEGER	
Status	TSTAMP	INTEGER	
Hauptadr	HADR	CHAR(20)	ADR,ADNR
Konzern	AGR	CHAR(20)	ADGP,AGR

/* Ausschnitt Ende */

2. **views.sql:** mit Hilfe dieser Datei kann eine Sicht auf die Datenbank geändert werden. Hierzu muss die Datei in die Datenbank geschrieben werden

mx001@<rechner>:~ > **psql -u pg_mx001 MXDBS < views.sql**

anschließend erscheinen z.B. bei einem Select auf die Tabelle adr

mx001@<rechner>:~ > **psql -u pg_mx001 MXDBS**
MXDBS=# **select * from viewadressen;**

statt der Spaltennamen die Titel der Felder.

Hinweis:

Das oben beschriebene Programm ist momentan noch nicht Bestandteil des Releases, kann aber über unsere Web-Seite abgerufen werden.

2. Modul - Installation

2.1. Die Datei modules.in

Die Datei 'modules.in' enthält alle Modulnamen, die zum Übersetzen der Komponenten des ESS notwendig sind.
Diese Datei enthält zur Zeit standardmäßig folgende Einträge:

```
mod_base                1
mod_mail                1
mod_ess                 1
mod_adressen            1
mod_dtaus               1
mod_zm                  1
mod_afa                 1
mod_zahl                1
mod_zahlbeleg           1
mod_anzahlung           1
mod_provision           1
mod_bestellvorschlag    1
mod_ticket              1
mod_wiedervorlage       1
mod_zeit                1
mod_ups                 0
mod_verband             1
mod_versand             1
mod_projekt             1
mod_dokumente           1
mod_crm                 1
mod_crmprojekt          1
mod_intrastat           1
mod_xml                 1
mod_liefermahnung       1
mod_bankbeleg           1
mod_korrespondenz       1
mod_gdpdu               1
mod_essmawi             1
```

Hierbei ist die Reihenfolge der Module zu beachten.
Eigen entwickelte Module müssen am Ende der Datei angehängt werden. Bei einem Releasewechsel werden alle uns unbekannten Module nach den ESS-Modulen wieder angehängt.

2.2. .ppi-Dateien

In *.ppi-Dateien müssen alle Dateien eingetragen, die vom Compiler berücksichtigt werden sollen. Sie müssen in jedem Verzeichnis vorhanden sein, in dem Dateien übersetzt werden sollen, und werden teilweise schon bei der Installation angelegt.

fomular_p.ppi: sie wird nicht automatisch generiert und muss vom Programmierer im Verzeichnis 'formular' angelegt werden. In der Datei müssen alle zu übersetzenden Formulare mit der Endung .sr\ stehen (siehe Kapitel 3.3.4).

maskdef_p.ppi: sie wird automatisch mit einem Eintrag

FORMULARS = \

im Verzeichnis 'maskdef' generiert. Alle Maskenquellcode-Dateien müssen im Anschluß eingetragen werden (siehe Kapitel 3.3.2).

help_p.ppi: sie wird nicht automatisch generiert und muss vom Programmierer im Verzeichnis 'help' mit dem Eintrag

HELPS = \

in der ersten Zeile angelegt werden.
Anschließend müssen alle Dateien eingetragen werden, in denen sich die Hilfetexte der Masken und Felder des Moduls befinden (siehe Kapitel 4.2.5).
Eine leere Datei wird vom Compiler nicht akzeptiert, deshalb muss gegebenenfalls die Datei gelöscht werden.

images_p.ppi: sie wird nicht automatisch generiert und muss vom Programmierer im Verzeichnis 'images' angelegt werden.
Hier müssen alle Bilder, die verwendet werden, eingetragen sein. Im Gegensatz zu den anderen _p.ppi Dateien muss hier noch

<modulname>/images

den Dateinamen vorangestellt werden (siehe Kapitel3.5).

2.3. Vorgehensweise der Modul-Installation

Vor der Modul-Einbindung muss das Standardsystem lauffähig sein.
Ins Verzeichnis '<benutzer>@<rechner>:~/src/' wechseln.

1. Das neue Modul muss in die Datei

   <benutzer>:~/src/modules.in

   eingetragen werden. Im folgenden Beispiel soll das Modul 'mod_beispiel' heißen. Es muss eine neue Zeile mit dem Namen 'mod_beispiel' angelegt werden.

2. Damit das Modul vom Compiler berücksichtigt wird, muss hinter dem Modulnamen eine '1' eingegeben werden. **Als Whitespaces ist nur der Tabulator zulässig!**

3. Weiter wird mit dem Aufruf:

   <benutzer>@<rechner>:~/src > make modulesdir

   eine leere Verzeichnisstruktur erstellt und folgende Verzeichnisse angelegt:

docs: dieses Verzeichnis ist für Hilfedateien, wie z.B. Dokumentationen vorgesehen, wird aber derzeit nicht bei der Modulentwicklung verwendet.

formscripts: wird nicht mehr verwendet

formular: hier liegen die Quelldateien mit Anwendungsfunktionalität.
.sr = kompilierte Dateien
.ssh = include-Dateien
.ssr = Quellcode-Dateien
.ssd = diese Dateien werden ausschließlich vom "make" verwendet und definieren Abhängigkeiten von Formularen und Include-Dateien.

help: hier liegen Dateien mit Hilfetexten, für alle Masken und Felder eines Modules.

images: hier liegen alle Bilder, wie Logos oder Hintergrundbilder als eps- oder jpg.-Dateien, oder Button-Bilder als .gif -Dateien.

maskdef: beinhaltet alle Quelldateien für Masken- und Felddefinitionen.
.def -files sind die Quellcode-Dateien der Maskendefinitionen.
.ded-files definieren Abhängigkeiten von Masken und include-files; diese Dateien werden generiert und ausschließlich vom "make" verwendet.
.deh-files sind include-Dateien und enthalten Definitionen.

4. Jetzt können die benötigten Dateien in den jeweiligen Verzeichnissen erstellt oder auch eine Archiv-Datei (*.tar, *.tgz) mit den Beispiel-Dateien ausgepackt werden.

5. Danach wird der Compiler mit dem Befehl 'make' aufgerufen (im Verzeichnis ~/src).

6. Die vom 'make' angezeigten Fehler müssen korrigiert werden.

Achtung!

zum nächsten Schritt darf man nur dann übergehen, wenn das 'make' fehlerfrei durchläuft! Enthält das Modul keine neuen oder geänderten Dateien, wird ein 'make'-Aufruf fehlschlagen, weil mit dem Einbeziehen eines neuen Moduls Änderungen erwartet werden. Die Datenbank-Struktur wird aktualisiert. Die Änderungen, die das neue Modul eventuell beinhaltet, werden dabei berücksichtigt.
Das Programm dafür lautet 'reltool' (siehe Kapitel 1.4) und muss im Verzeichnis
<benutzer>@<rechner>:~/src/programs
aufgerufen werden.

WICHTIG!!!

Vorher müssen die ESS-Prozesse heruntergefahren werden.
Dafür muss vom Benutzer-'root' der folgende Befehl ausgeführt werden:

<root>:~ # sh etc/pentaprise/ess/mxrc stopmxsd

Die Datenbank-Prozesse laufen mit diesem Befehl weiterhin.

Der Befehl für eine Postgres-Datenbank kann zum Beispiel wie folgt aussehen:

pg_dump -U pg_mx001 -s MXDBS | reltool -n mxmask.msk -sfxD -O -

Liegen nach Analyse der Datei reltool.out keine Fehler vor, kann man den folgenden Befehl

aufrufen:

psql MXDBS -U pg_mx001 < reltool.out

Damit werden die Datenbankänderungen aktiv!
Die Protokollierung der Aktion wird in die Datei load.prot geschrieben.

Hinweis:
Werden durch Module Änderungen an Grundfunktionalitäten des ESS erstellt, so hat hierarchisch immer das Formular des am letzten genannten Modules die höchste Priorität!

3. Dokumentausdruck ändern

Achtung!
In diesem Beispiel werden keine neuen Tabellen erstellt und keine neuen Felder zu existierenden Tabellen hinzugefügt. Das bedeutet, dass die Datenbank-Struktur nicht verändert wird!

3.1. Allgemeine Beschreibung des Druckaufbaus

Der Aufbau der Druckformulare ist **zellenorientiert** (vergleichbar mit Excel-Tabellen). Zellen-überschneidungen können nicht gedruckt werden! Die Zellendefinitionen erfolgen von links oben nach rechts unten.

Eine Seite ist in drei Bereiche aufgeteilt:

Kopfbereich:	(definiert in der Funktion header()) Hier sind für die erste Seite die Bereiche für Adresse und Absender festgelegt. Sie wird immer nach einem Seitenwechsel ausgeführt.
Druckbereich:	Hier werden Text und die einzelnen Positionen gedruckt. In diesem Bereich dürfen nur Zelleneigenschaften gesetzt werden und Zellenzuweisungen erfolgen
Fußbereich:	(definiert in der Funktion footer()) Hier werden Seitenumbrüche und Überträge berechnet. Sie wird immer vor einem Seitenwechsel ausgeführt.

Beispiel für ein Setzen von Zelleigenschaften:

cellprops (A_CENTER, F_BOLD, SLANT, C_FOR, C_BACK, C_LINE, L_WIDTH, 0, PREC);

hier gelten folgende Zellenattribute für alle nachfolgenden Zellenzuweisungen solange bis neue gesetzt werden:

A_CENTER:	Text in der Zelle wird zentriert dargestellt;
F_BOLD:	Text wird in der Schriftart (Font) "Bold" dargestellt;
SLANT:	beschreibt den Neigungswinkel der Schrift; hier keine Neigung;
C_FOR:	Vordergrundfarbe der Zelle;
C_BACK:	Hintergrundfarbe der Zelle;
C_LINE:	Rahmenfarbe der Zelle
L_WIDTH:	Liniendicke;
0:	Standardeinstellung: keine Linie um die Zelle;
PREC:	Darstellung der Nachkommastellen von float-Zahlen; hier keine;

Die Funktion nextcell() generiert eine Zelle und schließt immer an die vorherige Zelle an:

nextcell (cell_ypos, cell_xpos, print_width * 100, S_BIG, VSPACE);

print_width:	Zellenbreite; hier über die ganze Seitenbreite; (print_width*50 generiert eine Zelle über die halbe Seitenbreite)

S_BIG:	Schriftgröße; hier groß;
VSPACE:	definiert einen Raum innerhalb der Zelle für die Schrift (innerer Zell-Rahmen)

Die Zuweisung eines Inhalts in diese Zelle erfolgt mit der Zeile:

```
curcell = <Inhalt>;
```

wobei der Inhalt ein Text sein kann, der in der Datei mxform.<modulname> verwaltet wird und einer Zahl zugeordnet wurde (z.B. flangstring (1029, " ")), oder ein Feldinhalt (z.B. atrk.ATNR_ATRK_ATRK.m).
(siehe auch Kapitel 3.6 Definieren von Maskenvariablen)

3.2. Verwalten von Text für Druckdokumente

Um Texte an zentralen Stellen verwalten zu können empfiehlt es sich, Text der in Druckformularen verwendet werden soll, in der Datei mxform.<modulname> zu hinterlegen.

```
/* Code-Ausschnitt: mxform.mod_ess */

   ...
1029@D@Auftragsbestätigung@@
1029@GB@Order Confirmation@Confirmation of an order @
1029@NL@orderbevestiging@Orderbevestiging@
   ...
}
/* Ausschnitt-ENDE */
```

Die Nummer, die als 'langstring' für den Text zugewiesen wird, muss aus dem Nummernkreis, der dem Modul zugewiesen ist, verwendet werden.

```
Nummer@Sprachkennung@Text@Hilfetext@
```

Der Hilfetext muss nicht angegeben werden, kann aber als Anregung für den Übersetzer mitgegeben werden.
Der Aufruf des Textes im Druckformular erfolgt mit

```
flangstring (<Nummer>, " ");
```

3.3. Firmenadresse ausblenden

Es soll eine Auftragsbestätigung angepasst werden, die aus der Maske AUFTRÄGE gedruckt wird. Standardmäßig werden im Dokument-Kopf Kunden- und Firmen-Adressen gedruckt. In dem Beispiel soll die Firmen Adresse nicht mehr ausgegeben werden, weil z.B. das Hintergrund-Bild diese Information enthält.

3.3.1. Einbinden eines neuen Moduls mod_beispiel

Um beispielhaft Änderungen erklären zu können, soll ein Modul mod_beispiel in das System eingebunden werden. Hierzu wird folgendermaßen vorgegangen:

1. Eintrag des neuen Moduls mod_beispiel in die Datei modules.in (siehe 2.1)
2. Aufruf: < make modulesdir>
3. Eine Datei, in der alle Änderungen der Maskendefinitionen am Standard-System gemacht werden, wird im Verzeichnis

   ~/src/mod_beispiel/maskdef

   angelegt und hat den Namen mod_beispiel.def.
   Der Dateiname sollte mit 'mod' beginnen. Danach kommt der Modul-Name 'beispiel' und die für alle Masken-Dateien übliche Erweiterung '.def'. Die Datei muss folgende Zeilen beinhalten:

```
/* Code-Ausschnitt: mod_beispiel.def */

#include "../../include/maskdef/layout_base.deh"
OS_MODULE mod_beispiel 0 0 {
    ...
}
/* Ausschnitt-ENDE */
```

   Alle Änderungen werden innerhalb der geschweiften Klammern implementiert.Falls noch keine Feldänderungen gemacht wurden, muss zuerst mindestens eine Dummy-Änderung vorgenommen werden. Sonst kann kein Formular übersetzt werden.

Hinweis:

Für größere Projekte empfiehlt es sich, einer besseren Übersichtlichkeit wegen, Änderungen der Maskendefinitionen in Dateien zu schreiben, die den Namen der zu ändernden Maske haben (also Änderungen an der Maske adr werden in einer Datei adr.def definiert, etc.)
Alle diese Dateien müssen natürlich in der maskdef_p.ppi eingetragen werden!

3.3.2. Eintrag der Maskendefinitionsdatei in maskdef_p.ppi

Vor dem Compilieren muss die Maskendefinitionsdatei 'atrk.def' in die Datei 'maskdef_p.ppi' im gleichen Verzeichnis eingetragen werden. Solange eine Maskendatei in der Datei 'maskdef_p.ppi' nicht eingetragen ist, wird sie beim Compilieren nicht berücksichtigt. Die Eintragung sieht wie folgt aus:

```
/* Code-Ausschnitt: mod_beispiel/maskdef/maskdef_p.ppi */

MASKDEFS=\
                atrk.def\

/* Ausschnitt-ENDE */
```

3.3.3. Erzeugen eines Beispiel-Druckformulars

Das Original-Druckformular, das geändert werden soll, kann ins Verzeichnis formular des Beispiel-Moduls kopiert werden.
Die Formular-Datei soll hier druck_mbaratrk_beispiel.ssr genannt werden.

```
<benutzer>@<rechner /src/ mod_beispiel/formular>
cp ../../mod_ess/formular/druck_mbaratrk.ssr   druck_mbaratrk_beispiel.ssr
```

3.3.4. Eintrag des Beispiel-Druckformulars in formular_p.ppi

Diese muss in die Datei 'formular_p.ppi' , die im gleichen Verzeichnis erzeugt werden muss, eingetragen werden. Dabei wird die Datei mit der Erweiterung *.sr statt *.ssr eingegeben. die Eintragung für das Druckformular 'druck_mbaratrk_beispiel.ssr' sieht z.B. wie folgt aus:

```
/* Code-Ausschnitt: formular_p.ppi */

                druck_mbaratrk_beispiel.sr\

/* Ausschnitt-ENDE */
```

3.3.5. Definieren des Formular-Textes

Im Pull-Down-Menü "Formularauswahl" soll das neue Formular unter dem Namen 'NEU-Auftrag drucken' hinzugefügt werden. Der String "NEU-Auftrag drucken" muss deshalb in der Datei mxlang.mod_beispiel definiert werden.

```
/* Code-Ausschnitt: mxlang.mod_beispiel */

100001@D@NEU-Auftrag drucken@@

/* Ausschnitt-ENDE */
```

Beim Compilieren wird er dann automatisch in die Datei mxlang gelinkt, in der alle Strings definiert sind.

3.3.6. Zufügen des neuen Formulars in Formularauswahl der Maske

Jetzt muss noch ein Eintrag in den Formscripten in der Maskendefinition im Verzeichnis maskdef vorgenommen werden, damit unser neues Formular auch ausgeführt werden kann.

Beispiel: vi mod_beispiel/maskdef/atrk.def

```
/* Code-Ausschnitt: mod_beispiel/atrk.def */

OS_CREATEFORMSCRIPT atrk 100001 OS_AFTER  51100 ;
OS_UPDATEFORMSCRIPT atrk OS_FORMPULLDOWN 100001 OS_PRINT
        druck_mbaratrk_beispiel OS_AFTER druck_mbaratrk;

/* Ausschnitt-ENDE */
```

Mit diesem Eintrag wird unser neues Formular als neue Aktion in der Formularauswahl der Maske AUFTRÄGE zugefügt.

3.3.7. Ersetzen eines bestehenden Formulars

Um ein bestehendes Formular durch unser neues zu ersetzen muss in der selben Datei folgender Eintrag gemacht werden:

```
/* Code-Ausschnitt: mod_beispiel/atrk.def */

OS_UPDATEFORMSCRIPT atrk OS_FORMPULLDOWN 51100 OS_PRINT
    druck_mbaratrk_beispiel OS_REPLACE druck_mbaratrk;

/* Ausschnitt-ENDE */
```

Hierbei erfolgt keine sichtbare Änderung in der Maskenoberfläche, beim Ausführen der Aktion wird jedoch das neue Formular zum Einsatz kommen.

3.3.8. Änderungen im Druckformular vornehmen

Um Änderungen vorzunehmen muss die Datei druck_mbaratrk_beispiel.ssr geöffnet werden.
Für das Drucken der Adressen im Dokument ist die Funktion druckadress() verantwortlich. Die Funktion befindet sich in der Include-Datei 'druckkopf.ssh'.
(Für vi-Benutzer: setzen Sie den Cursor mitten in die Zeile und drücken Sie die Tasten 'g' und 'f'. So gelangen sie am schnellsten in die include-Datei. Mit ':n#' gehen sie wieder zurück)

Die Ausgabe der Mandanten-Adresse ist von der Definition von DOC_MAN_ADR abhängig. Die Definition findet in der Include-Datei 'layout.ssh' statt und sieht wie folgt aus:

```
/* Code-Ausschnitt: layout.ssh */

#define DOC_MAN_ADR

/* Ausschnitt-ENDE */
```

Damit die Änderungen sich nur auf die gewünschten Dokumente auswirken, muss eine neue Include-Datei erstellt werden.
Hierfür muss die Original-Include-Datei 'layout.ssh' ins Verzeichnis '~/src/mod_beispiel/formular' kopiert und in 'layout_beispiel.ssh' umbenannt werden:

<benutzer>@<rechner /src/ mod_beispiel/formular>
cp ../../mod_ess/formular/layout.ssh layout_beispiel.ssh

(alle abgeänderten Original-Dateien sollten nach folgendem Muster benannt werden <dateiname>_<modulname>.<dateierweiterung>).

Um die Ausgabe der Mandanten-Adresse zu verhindern, muss die Definition von DOC_MAN_ADR rückgängig gemacht werden. Das Auskommentieren der Zeile kann wie folgt aussehen:

```
/* Code-Ausschnitt: layout_beispiel.ssh */

//#define DOC_MAN_ADR

/* Ausschnitt-ENDE */
```

Im Druckformular druck_mbaratrk_beispiel.ssr muss die Eingabe:

#include "../../include/formular/layout.ssh"
durch:
#include "../../include/formular/layout_beispiel.ssh"
ersetzt werden.

3.3.9. Compilieren

Um den Compiler zu starten muss man sich als <benutzer> im Verzeichnis ~/src befinden und den Aufruf

mx001@<rechner>:~/src> make

eingeben.

Nach dem erfolgreichen Compilieren werden aus der Maske AUFTRÄGE die Dokumente ohne Firmenadresse ausgedruckt.

3.4. Ändern von Randeinstellungen

Abhängig von der Größe des Hintergrund-Bildes, könnte es nötig sein, die Rändergrößen für das Papier einzustellen.
Dies geschieht in der Include-Datei 'layout.ssh'.

Die Rändereinstellungen sehen wie folgt aus:

```
/* Code-Ausschnitt: layout_beispiel.ssh */

/* Randeinstellungen */
#ifndef B_LEFT
#define B_LEFT   200                    /* 200 Randabstand von Links in 1/10 mm */
#endif
#ifndef B_RIGHT
#define B_RIGHT  150                    /* 150 Randabstand von Rechts in 1/10 mm */
#endif
#ifndef B_TOP
#define B_TOP    100                    /* 100 Randabstand von Oben in 1/10 mm */
#endif
#ifndef B_BOTTOM
#define B_BOTTOM 270                    /* 270 Randabstand von Unten in 1/10 mm */
#endif

/* Ausschnitt-ENDE */
```

Durch das Ändern der Zahlen nach B_LEFT, B_RIGHT, B_TOP und B_BOTTOM in der includierten layout_Datei wird der Druckbereich des Dokumentes für alle Druckdokumente neu definiert.
Um nur für ein einzelnes Druckdokument Randeinstellungen zu Ändern muss im Druckdokument selbst eine Änderung vorgenommen werden, die der includierten layout_Datei vorangestellt sein muss.

Beispiel: Im Druckformular druck_mbaratrk_beispiel.ssr soll die Randeinstellung von Links geändert werden:

```
/* Code-Ausschnitt: layout_beispiel.ssh */

#define DOC_HEADER
#define DOC_FOOTER
#define DOC_KOPF
#define DOC_PRINTLINE
#define B_LEFT   300          /* 300 Randabstand von Links in 1/10 mm */

#include "../../include/formular/define.ssh"
#include "../../include/formular/pagesize.ssh"
#include "../../include/formular/color.ssh"
#include "../../include/formular/psfonts.ssh"
#include "../../include/formular/layout_beispiel.ssh"

/* Ausschnitt-ENDE */
```

3.5. Verwenden eines Hintergrund-Bildes

Es können mehrere Hintergrund-Bilder verwendet werden.
Ein Bild vom Typ "PostScript" (z.B. muster_logo.eps) muss in der richtigen Größe (z.B. für die ganze Seite mit Kopf- und Fusstext) in dem Verzeichnis

~/src/mod_beispiel/images

als eps.- oder gif-file abgelegt werden.
Damit die Grafik-Datei gefunden wird, muss im gleichen Verzeichnis eine Datei images_p.ppi' mit folgendem Eintrag erstellt werden:

```
/* Code-Ausschnitt: images_p.ppi */

mod_beispiel/images/muster_logo.eps\

/* Ausschnitt-ENDE */
```

In der Include-Datei 'druckkopf.ssh' kann zusätzlich die Anzeigegröße des Bildes bestimmt werden.
Damit die Änderungen sich nur auf bestimmte Dokumente auswirken, muss eine Kopie dieser Datei ins Modul kopiert und umbenannt werden (wie bei der Datei layout.ssh).
Die Datei wird ins Verzeichnis 'formular' kopiert und danach in 'druckkopf_beispiel.ssh' umbenannt.
Jetzt muss in der Funktion 'druck_logo' der Variablen 'lheight' ein Wert zugewiesen werden:

```
/* Code-Ausschnitt: druckkopf_beispiel.ssh */

lheight = S_NORM * 60;

/* Ausschnitt-ENDE */
```

Dadurch wird die Größe des Beispiel-Bildes auf die ganze Seite eingestellt.(S_NORM entspricht in etwa einer Zeilenhöhe)
Es kann auch für die Größe eine Zahlenangabe gemacht werden:

```
/* Code-Ausschnitt: druckkopf_beispiel.ssh */

lheight = 200;

/* Ausschnitt-ENDE */
```

Dadurch wird die Größe des Beispiel-Bildes 20 cm eingestellt.

Außerdem muss noch die Include-Datei in dem lokal kopierten Druckformular (im Beispiel: druck_mbaratrk_beispiel.ssr)) ersetzt werden.

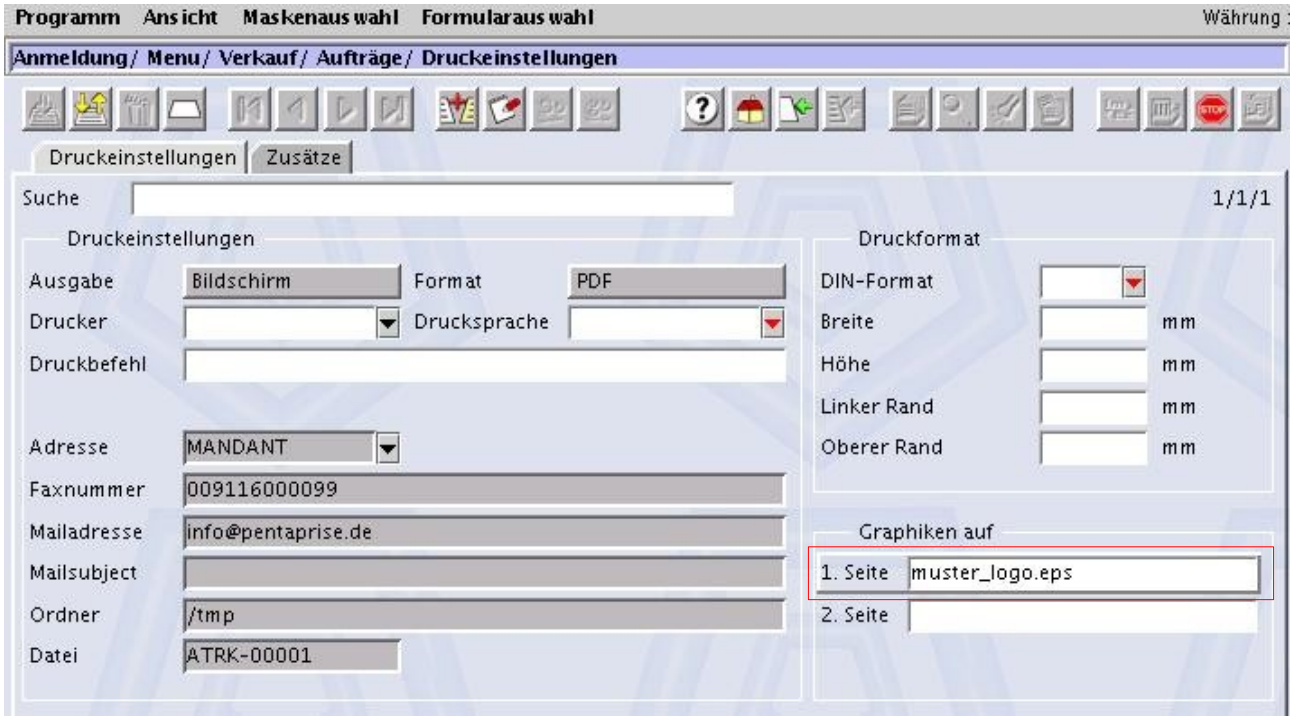
```
/* Code-Ausschnitt: druck_mbaratrk_beispiel.ssr */

// #include "../../include/formular/druckkopf.ssh"
#include "../../include/formular/druckkopf_beispiel.ssh"

/* Ausschnitt-ENDE */
```

Anschließend kann das Compilieren gestartet werden.

Um die Ausgabe eines Logos auf dem Druckdokument zu erhalten muss noch ein Eintrag in den Druckeinstellungen der ESS-Anwendung gemacht werden:



Im Bereich *Graphiken* muss das gewünschte Logo für die gewünschte Seite eingetragen werden.

Nach Aktivieren des Druckformulars wird das Logo (im Beispiel auf der 1.Seite) mittig abgebildet.

3.5.1. Ändern der Position des Hintergrund-Bildes

Um die Position des Logos auf dem Druckformular anstatt zentriert linksbündig zu drucken, muss noch ein Eintrag in der Datei druckkopf_beispiel.ssh geändert werden:

```
/* Code-Ausschnitt: druckkopf_beispiel.ssh */

        /*
         *  firmenlogo ausdrucken
         */
        cellprops (A_LEFT, F_NORM, SLANT, C_FOR, C_BACK, C_LINE, L_WIDTH,
                GRAPHIC, PREC);

/* Ausschnitt-ENDE */
```

3.5.2. Ändern eines Hintergrund-Bildes

Um ein schon bestehendes Hintergrund-Bild durch ein neues zu ersetzen, gehen wir wie folgt vor:
(Voraussetzung: Die Bild-Datei ist im Verzeichnis images und in der Datei images_p.ppi hinterlegt)

a) In der Maske AUFTRÄGE aus dem Pull-Down-Menü *Maskenauswahl* die Maske DRUCKEINSTELLUNGEN wählen.

b) Im Feld *Logo 1.Seite* den Bildnamen ohne den Pfad (z.B :<bildname>.eps) eingeben.
Soll das gleiche Hintergrund-Bild an allen anderen Seiten erscheinen, muss der Bildname auch ins Feld *Logo 2.Seite* eingegeben werden. Ansonsten kann für weitere Seiten kein, oder ein anderes Hintergrundbild verwendet werden. (siehe Kapitel 3.5)

3.6. Definieren von Maskenvariablen

Ein wichtiges Element in der Modulentwicklung ist die Möglichkeit Maskenvariablen zu definieren.

Über die Anweisung <mask ATRP atrp> wird eine Variable atrp vom Typ Maske-ATRP definiert.
Die Definition der Variablen muss innerhalb einer Funktion stehen.
Diese Variable kann über Anweisungen wie

```
< i = dbfetchcursor (atrp);>          // (i = Rückgabewert der Funktion)
```

mit Werten gefüllt werden.

Eine Variable vom Typ Maske kann über den Variablen- oder Aliasnamen, den Feldnamen laut der Maskendefinition und einen Typ angesprochen werden.
Die Syntax lautet in diesem Fall: <**alias.Feldname.Typ**>

Beispiel: artp**.**PORA_ATRP_ATRP**.**m gibt den aktuellen Wert des Feldes *Positionsrabatt* der Maske AUFTRAGSPOSITION aus.

es sind vier Typen verfügbar:

- s Selektionskriterium (suchend in Datenbank zugreifen);
- m aktueller Maskeninhalt oder der entsprechende Datenbankinhalt;
- o Ordnungskriterium (+ = aufsteigende Reihenfolge; - = absteigende Reihenfolge);
- i Indikatorfeld (relevant für die Überprüfung auf 0 oder NULL Inhalt);

Wenn Variablen global definiert sind, dürfen sie nicht noch einmal in einer anderen Datei definiert sein.

Über diese Variablen kann z.B. auf Datenbankinhalte zugegeriffen werden, um sie in einem Druckformular auszugeben.

3.7. Zusammenfassung

Um die Tatsache, dass Änderungen nur für das neue Druckformular wirksam sind zu bestätigen, kann man die Druckausgaben der beiden Formulare 'Auftrag drucken' und 'NEU Auftrag drucken' aus der Maske AUFTRAG vergleichen.
Im Standard-Dokument erscheint z.B. das Logo oben mittig in der Druckausgabe.
Alle vorgenommenen Änderungen im Druckformular erscheinen nur beim Ausführen des neu erzeugten Druckformulars 'NEU Auftrag drucken'.

Wünscht man die gleichen Änderungen für anderen Dokumente, müssen wie oben beschrieben

- entsprechende Druckformulare ins Modul kopiert
- und in die Datei 'formular_p.ppi' eingetragen werden;
- die Include-Zeilen müssen in den entsprechenden Dateien geändert werden

Auf die gleiche Art können andere Änderungen in jedem kopierten Druckformular vorgenommen werden.
Es ist auch möglich Funktionen in den Include-Dateien zu ändern oder neue Funktionen zu schreiben. Hierbei ist zu beachten:

- Funktionen müssen vor ihrem Aufruf definiert sein;
- Nach einer Zuweisung in einer Funktion sind keine Variablendefinitionen in dieser Funktion möglich;
- es darf keine Funktionsdeklaration existieren, die nicht aufgerufen wird

Die neuen oder geänderten Include-Dateien müssen in die entsprechenden Druckformulare einbezogen werden.

4. Masken oder Formulare ändern

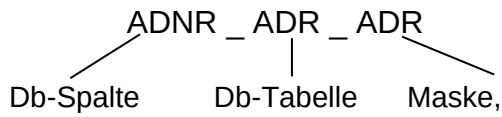
Achtung:

Viele Maskenänderungen können auch im Edit_Modus der betreffenden Maske vorgenommen werden. (siehe Benutzerhandbuch)
Alle im Edit-Modus vorgenommenen Feld- oder Frame-Änderungen überschreiben grundsätzlich die Definitionen der Maskendatei!!!!

Tip für die Praxis:

sollte eine Änderung, die in die Maskendatei geschrieben wurde, aus unerfindlichen Gründen nicht wirksam werden, empfiehlt es sich zu überprüfen, ob das entsprechende Feld bzw. der Frame über den Edit-Modus schon geändert wurde.
Diese Änderungen sind in der Tabelle FELDPOSITIONEN (SYSTEM -> FELDPOSITIONEN) nachzulesen.
Nach Löschen des jeweiligen Eintrags ist die Änderung der Maskendatei wirksam!

Beziehung Feldname - Datenbank:



Das Feld ADNR_ADR_DEB steht also für das selbe Feld in der Datenbank, das in der Maske Debitor dargestellt wird, d.h. beide Felder referenzieren den selben Wert auf der Datenbank.

4.1. Maskenänderung anlegen

Alle Masken- oder Formular-Änderungen am Standard-System müssen in der Datei mod_beispiel.def (oder für größere Projekte jeweils in einer Datei mit dem Namen der zu ändernden Maske) eingetragen werden.
Sie wird im Verzeichnis

~/src/mod_beispiel/maskdef

angelegt.

In die Datei müssen folgende Zeilen eingetragen werden:

```
/* Code-Ausschnitt: mod_beispiel.def */

OS_MODULE mod_beispiel 0 0 {
...
...
}

/* Ausschnitt-ENDE */
```

OS_MODULE <Modulname> Developernummer Modulnummer { }

Die Developernummer wird von Pentaprise an Partner bzw. Kunden vergeben und ist eindeutig; die Kombination von Developernummer und Modulnummer muss ebenfalls eindeutig sein!

Beispiel: Developernummer: 1, Modulnummer: 100, Modulname: mod_beispiel
Der Eintrag sieht dann wie folgt aus:

OS_MODULE mod_beispiel 1 100 { }

Alle Änderungen werden innerhalb der geschweiften Klammern implementiert.

4.2. Ändern bestehender Felder

4.2.1. Submaske verkleinern

Die Änderung einer Feldeigenschaft kann wie folgt aussehen:

```
OS_UPDATEFIELD <feldname> {
   <FELDEIGENSCHAFT> <neuwert>;
}
```

Beispiel 1: die Submaske *Artikel* in der Maske ARTIKEL soll schmäler gemacht werden:

/* Code-Ausschnitt: mod_beispiel.def */

```
OS_MODULE mod_beispiel  1 100 {
...
...
OS_UPDATEFIELD SUBMASK1_ARTI {
   OS_VISLEN 63 * FONTWIDTH;
   OS_HEIGHT 11 * LINEHEIGHT - LINEGAP;
}
}
```

/* Ausschnitt-ENDE */

Hierbei ist zu beachten, dass die Angabe einer Höhe für die Submaske auch dann erwartet wird, wenn sie nicht geändert werden soll.

Vor dem Compilieren muss die Maskendefinitionsdatei mod_beispiel.def in der Datei maskdef_p.ppi im gleichen Verzeichnis eintragen werden.
Solange ein Maskenname in der Datei nicht eingetragen ist, wird die Maske beim Compilieren nicht berücksichtigt.
Der Eintrag sieht wie folgt aus:

```
/* Code-Ausschnitt: maskdef_p.ppi */

MASKDEFS=\
    mod_beispiel.def\

/* Ausschnitt-ENDE */
```

Nach dem erfolgreichen Compilieren muss der mxsd-Prozess gestoppt und neu gestartet werden, damit die Änderungen wirksam werden.
Dazu muss als root (!) folgender Befehl eingegeben werden:

```
etc/pentaprise/ess/mxrc stopmxsd <Benutzername>
etc/pentaprise/ess/mxrc startmxsd <Benutzername>
```

Im ESS erscheint nun die Submaske ARTIKELPOSTEN in der Maske ARTIKEL nur noch in halber Maskenbreite.

4.2.2. Feldgröße ändern

Beispiel 2: das Feld *Abmessung* (GRO_ARTI_ARTI) im Register Bestand soll verkürzt werden:

```
/* Code-Ausschnitt: mod_beispiel.def */

OS_MODULE mod_beispiel  1 100 {
...
...
OS_UPDATEFIELD GRO_ARTI_ARTI {
      OS_VISLEN 12 * FONTWIDTH;
}
}

/* Ausschnitt-ENDE */
```

Nach dem erfolgreichen Compilieren muss der mxsd-Prozess gestoppt und neu gestartet werden, damit die Änderungen wirksam werden.
Dazu muss als root (!) folgender Befehl eingegeben werden:

```
etc/pentaprise/ess/mxrc stopmxsd <Benutzername>
etc/pentaprise/ess/mxrc startmxsd <Benutzername>
```

Im ESS erscheint nun das Feld *Abmessung* in der Maske ARTIKEL nur noch in halber Länge.

4.2.3. Feld ausblenden

Beispiel 3: das Feld Limit (LIM_DEB_DEB) in der Maske DEBITOREN soll ausgeblendet (für den Anwender unsichtbar) werden;

Hierfür muss die Feldeigenschaft OS_ACCESS auf OS_NOACCESS gesetzt werden.

Die Feldänderung wird durch OS_UPDATEFIELD gefolgt vom Feldnamen eingeleitet.
Änderungen von einzelnen Eigenschaften werden innerhalb der geschweiften Klammer eingegeben.

```
/* Code-Ausschnitt: mod_beispiel.def */

    OS_UPDATEFIELD LIM_DEB_DEB {
                        OS_ACCESS OS_NOACCESS;
    }

/* Ausschnitt-ENDE */
```

Nach dem erfolgreichen Compilieren muss der mxsd-Prozess neu gestartet werden, damit die Änderungen wirksam werden.

4.2.4. Feldposition ändern

Beispiel 4: die Position des Feldes Summenkonto-Kennzeichen (SUMK_KTO_KTO) in der Maske KONTENRAHMEN soll geändert werden.

Die Feldposition wird wie folgt definiert

```
OS_POSITION ( <spalte>, <zeile> );

Parameter:
      spalte - Ganzzahl
      zeile   - Ganzzahl
```

Die Standardposition für das Feld SUMK_KTO_KTO ist:

```
/* Code-Ausschnitt: kto.def - im Standard-Modul */

OS_FRAME KTO_SONST [FWIDTH(56), FHEIGHT(7)] {
      ....
      OS_ADD SUMK_KTO_KTO  [FXPOS(0),  FYPOS(4)];
      .....
    }

/* Ausschnitt-ENDE */
```

Jetzt soll das Feld 10 Stellen nach rechts verschoben werden :

```
/* Code-Ausschnitt: mod_beispiel.def */

OS_UPDATEFRAME KTO_SONST [0, 0] {
   OS_MFIELDLIST {
      OS_MOVE SUMK_KTO_KTO [FXPOS(10), FYPOS(4)];
   }
```

```
}

OS_UPDATEMASK KTO {
   OS_MFIELDLIST {
      OS_MOVE OS_SCREEN 1 OS_TITLE 16200{
         OS_UPDATE KTO_SONST;
      }
   }
}

/* Ausschnitt-ENDE */
```

Beim Ändern der Feldposition ist zu beachten, dass der Platz, an den das Feld verschoben werden soll, frei ist (d.h. von keinem anderen Feld bereits belegt ist).
In diesem Fall würde der Compiler eine Fehlermeldung ausgeben.

Nach dem erfolgreichen Compilieren muss der mxsd-Prozess neu gestartet werden, damit die Änderungen wirksam werden.

4.2.5. Feldhilfe ändern

Die Hilfe für jedes Feld ist in der Hilfedatei im Modul-Verzeichnis help folgendermaßen definiert:

```
/* Code-Beispiel */                                    // immer gleiche Form bewahren!

@@<sprachkennzeichen>@<feldname>@<hilfstext>@
...

/* Ausschnitt-ENDE* /
```

Um den Hilfetext für ein Feld zu ändern, wird in der Hilfe-Datei /mod_beispiel/help/mod_beispiel eine neue Hilfe-Definition implementiert. Dadurch wird die Standard-Hilfe für das Feld überschrieben.
z.B. der neue Hilfetext für das Feld SUMK_KTO_KTO:

```
/* Code-Ausschnitt: mod_beispiel */

@@D@SUMK_KTO_KTO@Neuer Hilfetext für das Feld "Summenkonto-Kennzeichen"@

/* Ausschnitt-ENDE */
```

Anschließend muss die Hilfe-Datei noch in eine help_p.ppi-Datei eingetragen werden, damit sie beim Compilieren berücksichtigt wird:

```
/* Code-Ausschnitt: help_p.ppi */

HELPS=\
        mod_beispiel\

/* Ausschnitt-ENDE */
```

Nach dem erfolgreichen Compilieren kann die Seite <Hilfe zur Maske KONTENRAHMEN> durch das Anklicken des Hilfe-Buttons oder durch Strg+F1 aus der Maske KONTENRAHMEN aufgerufen werden.

Unter der Feldbeschreibung *Summenkonto-Kennzeichen* steht nun der neue Text.

4.2.6. Feldbeschriftung ändern

Die Beschriftung eines Feldes wird in der Felddefinition innerhalb der Zeile durch Eingabe einer Nummer festgelegt. Der Nummer ist ein Text (in einer Stringtabellendatei) zugeordnet.
Mit OS_PLEFT wird die Feldbeschriftung in dem Beispiel 'links' neben dem Feld angezeigt.

```
/* Code-Ausschnitt: mod_ess/maskdef/kto.def */

OS_BOOL SUMK_KTO_KTO [1 * FONTWIDTH, 1] OS_READ|OS_WRITE {
   OS_DBINFO KTO, SUMK;
   OS_TITLE 16216 OS_TITLEPOS OS_PLEFT;
   .....

/* Ausschnitt-ENDE */
```

Um die Beschriftung eines Feldes zu ändern muss die bestehende Titel-Nummer mit dem neuen Text überschriebenwerden.
Dafür wird der Datei 'mxlang.mod_beispiel' die folgende Zeile hinzugefügt:

```
/* Code-Ausschnitt: mxlang.mod_beispiel */

16216@D@Summenk.KZ@@

/* Ausschnitt-ENDE */
```

Nach dem erfolgreichen Compilieren erscheint auf der Maske der neue Feldtitel.

4.3. Ändern eines bestehenden Maskenformulars

Beispiel 5: das Maskenformular 'deb.ssr' in der Maske DEBITOREN soll durch das geänderte Formular 'deb_beispiel.ssr' ersetzt werden.

Ein Maskenformular ist für die Funktionalität einer Maske verantwortlich.
Besteht der Bedarf die Funktionalität zu ändern, müssen folgende Schritte durchgeführt werden:
Das schon bestehende Maskenformular 'deb.ssr' muss in das Verzeichnis

~/src/mod_beispiel/formular

kopiert und in 'deb_beispiel.ssr` umbenannt werden.

```
<benutzer>@<rechner /src/ mod_beispiel/formular>
cp ../../mod_ess/formular/deb.ssr   deb_beispiel.ssr
```

Jetzt können die gewünschten Änderungen an dem Formular vorgenommen werden.
Beispielhaft soll über das Maskenformular das Feld *Kennzeichen 1* zum Force-Feld werden:

```
/* Code-Ausschnitt: mod_beispiel/formular/deb_beispiel.ssr */

  if (!ret) {
     setcurfieldforce ("K1_DEB_DEB, 1);
  }

/* Ausschnitt-ENDE */
```

Damit der Compiler das Maskenformular finden kann, muss die umbenannte Datei im Verzeichnis ~/src/mod_beispiel/formular_p.ppi eingetragen sein:

```
/* Code-Ausschnitt: mod_beispiel/formular_p.ppi */

FORMULARS=\
        deb_beispiel.sr

/* Ausschnitt-ENDE */
```

In der Datei mod_beispiel.def muss nun noch die Änderung des Maskenformulars eingetragen werden.
Die Syntax hierfür lautet:

```
OS_UPDATEFORMSCRIPT
    <Name des bestehenden Formulars> <Bereich>
    <Name des neuenFormulars><Aktion><Name des bestehenden Formulars>
```

Der Bereich kann sei: OS_FORMPULLDOWN
OS_MASK
OS_INSERT
OS_UPDATE

Eine Aktion kann sein:OS_REPLACE
OS_BEFORE //Ausführen vor dem bestehenden Formular
OS_AFTER //Ausführen nach dem bestehenden Formular

Der Eintrag in der mod_beispiel.def muss also lauten:

```
/* Code-Ausschnitt: mod_beispiel.def */

 OS_UPDATEFORMSCRIPT deb OS_MASK deb_beispiel OS_REPLACE deb;

/* Ausschnitt-ENDE */
```

Nach dem Abspeichern muss nur der Compiler aufgerufen werden.
Das geänderte Formular ist nun wirksam.

Zusammenfassung:

- kopieren des Original-Formulars ins Modul und Umbenennung desselben;
- neues Formular in die Datei formular_p.ppi im eintragen;
- Änderung in der Datei mod_beispiel.def im Verzeichnis maskdef eintragen;
- Änderungen in dem Formular vornehmen;
- Compilieren;

4.4. Einfügen einer neuen Submaske

Beispiel 6: eine zweite Submaske, die mit der Lieferantenmaske verknüpft ist, soll neben der ersten, die in Kapitel 4.2.1 verkleinert wurde, angelegt werden;

In der Datei mod_beispiel.def müssen die Änderungen wieder eingetragen werden:

```
/* Code-Ausschnitt: mod_beispiel.def */

OS_SUBMASK SUBMASK3_ARTI [55 * FONTWIDTH, 60] OS_READ|OS_WRITE {
      OS_LINEHEIGHT FONTHEIGHT;
      OS_HEIGHT 11 * LINEHEIGHT - LINEGAP;
      OS_NEXTMASK LFART;
      OS_GIVEFIELDLIST{
         ART_ARTI_ARTI, ART_LFART_LFART;
         ART_ARTI_ARTI, ART_LFART_LFART, OS_TGSEL;
      }
}

OS_UPDATEFRAME ARTI_SUB1 [0, 0]{        // nötig bei neuem Feld bzw. Submaske
      OS_MFIELDLIST {
         OS_ADD SUBMASK3_ARTI [FXPOS(57), FYPOS(0)];
      }
}

/* Ausschnitt-ENDE */
```

Nach dem erfolgreichen Compilieren muss der mxsd-Prozess gestoppt und neu gestartet werden, damit die Änderungen wirksam werden.

4.5. Einfügen neuer Felder in eine bestehende Maske

4.5.1. Feld ohne Datenbankbezug

Beispiel 7: ein berechnetes Feld soll in der Maske ARTIKEL hinzugefügt werden.
Das Feld hat für die Maske keine datenbanktechnischen Auswirkungen und beinhaltet den doppelten Einzelpreis. Der berechnete Wert wird nur in der Maske angezeigt und in keine Tabelle gespeichert.

Zuerst kann, wie in dem Kapitel "4.2.5 Feldhilfe ändern" beschrieben, der Hilfetext zum neuen Feld definiert werden.
Das Feld soll DEZPR_ARTI genannt werden, und in der Datei 'mod_beispiel' im Verzeichnis help muss der folgende Eintrag gemacht werden:

```
/* Code-Ausschnitt: mod_beispiel */

@@D@DEZPR_ARTI
berechneter doppelter Einzelpreis
@

/* Ausschnitt-ENDE */
```

Danach muss eine Nummer mit einer Zeichenkette definiert werden, die als Beschriftung für das neue Feld dienen soll.
String-Definitionen erfolgen immer im Verzeichnis maskedef in der Datei mxlang.mod_beispiel:

```
/* Code-Ausschnitt: mxlang.mod_beispiel */

 100002@D@2xEZ-Preis@doppelter Einzel-Preis@

/* Ausschnitt-ENDE */
```

Feldtitel — Tooltip-Text

Die Definition des Feldes findet in der Datei 'mod_beispiel.def' statt und muss von einem Update des zugehörigen Frames, in dem die Position des Feldes festgelegt wird, gefolgt sein:

```
/* Code-Ausschnitt: mod_beispiel.def */

 OS_CURRENCY DEZPR_ARTI [10 * FONTWIDTH, 10 ] OS_READ {
      OS_TITLE 100002 OS_TITLEPOS OS_PLEFT;
      OS_TITLELEN 14 * FONTWIDTH;
      OS_DEFAULT "0.00";
  }

OS_UPDATEFRAME ARTI_BUCH [0, 0] {
      OS_MFIELDLIST {
          OS_ADD DEZPR_ARTI [FXPOS(35), FYPOS(2)];
       }
}

/* Ausschnitt-ENDE */
```

OS_CURRENCY	Datentyp des Feldes;
DEZPR_ARTI	Name des Feldes,
DEZPR	der Namensteil ist frei definierbar (hier für Doppelter Einzelpreis)
ARTI	der Namensteil muss wie die Maske heißen;
OS_READ	die Eigenschaft sagt aus, dass das Feld schreibgeschützt ist;
OS_TITLE <zahl>	mit der Definition der <Zahl> wird die Beschriftung des Feldes bestimmt (String, der in mxlang_mod_beispiel für die Zahl definiert ist);
OS_TITLEPOS OS_PLEFT	die Beschriftung wird links vom Feld angezeigt;
OS_DEFAULT	als Standard-Vorgabe wird der Wert "0,00" angezeigt.

Mehr Information zur Felddefinition kann man im Programmierhandbuch finden.
Nach dem Compilieren und dem Neustarten des mxsd-Prozesses wird das neue Feld in der Maske ARTIKEL angezeigt.

Um den Wert des Feldes auszurechnen, muss das Maskenformular erweitert werden.
Wie schon in Kapitel 5 beschrieben muss die Datei 'arti.ssr' in das Verzeichnis 'formular' kopiert und in 'arti_beispiel.ssr' umbenannt werden.
Die Datei wird mit dem folgendem Code erweitert:

```
/* Code-Ausschnitt: arti_beispiel.ssr */

if (!ret) {
      if (arti.EZPR_ARTI_ARTI.m != 0.00) {
            arti.DEZPR_ARTI.m = arti.EZPR_ARTI_ARTI.m * 2;
      } else {
            arti.DEZPR_ARTI.m = 0.00;
      }
      setcurfieldcont (arti.DEZPR_ARTI);  // aktualisieren des Feldes !
}

/* Ausschnitt-ENDE */
```

Das neue Maskenformular muss in die Datei formular_p.ppi eingetragen werden.
In der Datei mod_beispiel.def wird das Maskenformular für die Maske neu deklariert:

```
/* Code-Ausschnitt: mod_beispiel.def */

OS_UPDATEFORMSCRIPT arti OS_MASK arti_beispiel OS_REPLACE arti

/* Ausschnitt-ENDE */
```

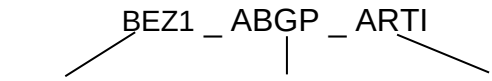
Nach dem Compilieren und dem Neustarten der mxsd-Prozesse sind die Änderungen in der Maske zu sehen.

4.5.2. Feld mit DB-Bezug

Beispiel 8:

Zur Maske ARTIKEL soll noch ein Feld hinzugefügt werden, das eine Verknüpfung beinhaltet.
Das neue Feld soll BEZ1_ABGP_ARTI heißen und die entsprechende Bezeichnung zur Buchungsgruppe, die in der Tabelle BUCHUNGSGRUPPE hinterlegt ist, anzeigen.
Der Wert des Feldes soll aber **nicht** in der Tabelle ARTIKEL (ARTI) **gespeichert, sondern** über die Verknüpfung mit dem bestehenden Feld *Buchungsgruppe* (ABNR_ARTI_ARTI) aus der Maske BUCHUNGSGRUPPE (ABGP) **geholt**, **und** in der Maske ARTIKEL lediglich **angezeigt** werden.

Beziehung Feldname - Datenbank:



Der Wert des Feldes *Buchungsgruppe* wird ebenfalls aus dieser Maske übernommen und, im Unterschied zu unserem neuen Feld, in der Tabelle ARTIKEL (ARTI) abgespeichert.

1. Die Definition und das Hinzufügen des Feldes 'BEZ1_ABGP_ARTI' ist ähnlich wie im Beisp. 7:

```
/* Code-Ausschnitt: mod_beispiel.def */

OS_TEXT  BEZ1_ABGP_ARTI [20 * FONTWIDTH, 230]  OS_READ {
OS_DBINFO ABGP,  BEZ1;
OS_TITLE 100003  OS_TITLEPOS  OS_PLEFT;
OS_TITLELEN  10 * FONTWIDTH;
OS_JOINFIELD ABNR_ARTI_ARTI;
}

OS_UPDATEFRAME ARTI_BUCH [0, 0] {
    OS_MFIELDLIST {
        OS_ADD BEZ1_ABGP_ARTI [FXPOS(35), FYPOS(1)];
    }
}

/* Ausschnitt-ENDE */
```

der Feldname besteht aus drei Teilen

BEZ1	der Namensteil ist frei definierbar;
ABGP	die Tabelle, zu der das Feld gehört;
ARTI	die Maske, in der das Feld angezeigt wird.

OS_TEXT ..... [,]	die erste Zahl definiert die Größe des Feldes auf der Datenbank, die zweite Zahl die Feldgröße auf der Maske .
OS_DBINFO <tabellename>, <feldname>;	der Datenbank wird das Feld 'BEZ1' zur Tabelle 'ABGP' gehörend, bekannt gemacht.
OS_JOINFIELD <feldname>	gibt an, über welches Feld die Verknüpfung zur Tabelle ABGP hergestellt wird.

2. Wie in Beispiel 7 muss der Hilfetext und die Feldbeschriftung noch definiert werden!
3. Der Wert des verknüpften Feldes 'BEZ1_ABGP_ARTI' wird nicht gleich nach dem Ausfüllen des Verknüpfungsfeldes 'ABNR_ARTI_ARTI' angezeigt.

Der aktuelle Datensatz muss zuerst abgespeichert werden.

Um das Feld gleich anzuzeigen, muss eine Aktualisierung (OS_UPDATEFIELD) für das Feld ABNR_ARTI_ARTI' gemacht werden.

Diese sieht wie folgt aus:

```
/* Code-Ausschnitt: mod_beispiel.def */

 OS_UPDATEFIELD ABNR_ARTI_ARTI {
     OS_TAKEFIELDLIST {
          BEZ1_ABGP_ABGP,  BEZ1_ABGP_ARTI;
     }
}

/* Ausschnitt-ENDE */
```

OS_TAKEFIELDLIST{ } bewirkt, dass die Wertübernahme des Feldes BEZ1_ABGP_ABGP ins Feld BEZ1_ABGP_ARTI parallel zur Wertübernahme des Feldes ABNR_ABGP_ABGP ins Feld ABNR_ARTI_ARTI passiert.

Mehr zu den Definitionen im Programmierhandbuch des ESS.

4. Vor dem Compilieren muss nun eine Sicherung der mxmask.msk erfolgen!
Nach dem erfolgreichen Compilieren müssen die Datenbankprozesse gestoppt werden.
Anschließend wird wie unter 1.4 beschrieben, die Datenbankstruktur mit dem Befehl 'reltool' aktualisiert und die Datenbankprozesse werden wieder gestartet.

Die Änderungen in der Maske sind nun vollzogen.

5. Neues Element in Maskenauswahl

(mit bestehender Maske)

Beispiel 9:

In diesem Beispiel soll das Pull-Down-Menü *Maskenauswahl* (bzw. Der Maskenbaum) in der Maske TÄTIGKEITSGRUPPEN ('RGRP') um eine Verknüpfung zur Maske TÄTIGKEITEN ('RES') erweitert werden. (unter STAMMDATEN -> LAGER)

Dafür müssen in der Datei 'mod_beispiel.def' folgende Zeilen eingegeben werden.

```
/* Code-Ausschnitt: mod_beispiel.def */

OS_UPDATEMASK RGRP {              //Existierende Maske ändern.
   OS_NMGIVELIST OS_BEFORE OS_FIRST {
      RES {                       /*Innerhalb der Klammer können Quell- und Zielfelder  stehen, zwischen
        ;                          *denen die Werte übergeben werden können*/
   }
}

/* Ausschnitt-ENDE */
```

Die neue Auswahlmöglichkeit in dem Maskenauswahl-Menü soll den gleichen Namen wie die Maske TÄTIGKEITEN haben.

Da keine Werte aus der Maske 'RGRP' in die Maske 'RES' übergeben werden, muss innerhalb der inneren Klammer mindestens ein Semikolon vorhanden sein.
Bei Bedarf könnten Feldwerte aus der Maske 'RGRP' in die Maske 'RES' übergeben werden.

Nach dem erfolgreichen Compilieren muss der der mxsd-Prozess gestoppt und neu gestartet werden, damit die Änderungen wirksam werden.

6. Neues Element in Formularauswahl

(mit bestehendem Formular)

Beispiel 10:

Zur *Formularauswahl* in der Maske 'RGRP' (TÄTIGKEITSGRUPPEN) soll noch ein Formular hinzugefügt werden.

Als Beispiel wird das gleiche Druckformular 'survey_print.ssr' verwendet, das über die existierende Auswahlmöglichkeit "Listendruck" aufgerufen wird.
Da keine UpDate-Möglichkeit für das Pull-Down-Menü existiert, muss die Definition des Menüs komplett wiederholt werden.

Danach kommt die neue Auswahlmöglichkeit.

Die existierende Definition findet man unter ~/src/mod_ess/maskdef/<maskenname>
(im Beispiel: ~/src/mod_ess/maskdef/rgrp)

```
/* Code-Ausschnitt: mod_beispiel.def */

FORMSCRIPT rgrp
FORMPULLDOWN {
   SCRIPT 60005 {
      FORMULAR  SURVPRINT survey_print;          //exist. Menüauswahl
   }
   SEPARATOR;                                    //neue Trennlinie

   SCRIPT 100004 {
      FORMULAR SURVPRINT survey_print;           //neue Menüauswahl
   }
}
END

/* Ausschnitt-ENDE */
```

Die Beschriftung im Menü wird mit der Definition bestimmt:

```
/* Code-Ausschnitt: mxlang.mod_beispiel */

   100004@D@Beispiel10-Druckformular@@

/* Ausschnitt-ENDE */
```

Nach diesen Eintragungen muss nur kompiliert werden, um die Änderungen auf der Maske wirksam zu machen.

Auf diese Weise können sowohl Druckformulare als auch Formulare, die bestimmte Funktionalitäten beinhalten, hinzugefügt werden.

7. Neue Maske erstellen

Eine neue Maske soll erstellt werden, in der zusätzliche Informationen zu einem Artikel angezeigt werden.
Die Maske soll 'ARTIZU' heißen und der Maskentitel soll ARTIKEL -ZUSATZINFO sein.

Beim Erstellen einer komplett neuen Maske empfiehlt es sich hierfür eine eigene Maskendefinitions-Datei anzulegen, die dem Namen der Maske entsprechen sollte.
Für unser Beispiel wird also eine Datei 'artizu.def ' im Verzeichnis

<benutzer>@<rechner>:~/src/<modulname>/maskdef

angelegt.

7.1. Sichern der Datei mxmask.msk

Bei Modulen, die über die Maskendefinitionen eine Datenbankänderung definieren, wird die Datei mxmask.msk neu erstellt, und somit die aktuelle mxmask.msk überschrieben.
Die Definitionen der Datenbankstruktur werden über die mxmask.msk in die Datenbank eingearbeitet.
Die Checksumme der Datenbank muss mit der in der Datei mxmask.msk hinterlegten Checksumme übereinstimmen, sonst starten die Prozesse nicht.
Deshalb muss unbedingt vor Programmiertätigkeiten, die in die Datenbankstruktur eingreifen, die mxmask.msk gesichert werden! (Siehe auch Kapitel 2.3.)

7.2. Eintrag in maskdef_p.ppi

Da die Maske in ein existierendes Menüsystem eingebunden wird, muss im gleichen Verzeichnis noch eine Masken-Datei 'mod_beispiel.def' erstellt sein, in der Änderungen des Standardsystems vorgenommen werden können.

Die beiden Masken-Dateien müssen in die Datei 'maskdef_p.ppi' im gleichen Verzeichnis eingetragen werden :

```
/* Code-Ausschnitt: maskdef_p.ppi */

MASKDEFS=\
        artizu.def\
        mod_beispiel.def\

/* Ausschnitt-ENDE */
```

Durch diese Eintragung werden neue Masken beim Compilieren zusätzlich zu Standard-Modulen übersetzt.

7.3. Definieren der neuen Maske

In der Datei 'artizu.def' wird die neue Maske vollständig definiert.
Die Maskendefinitions-Datei enthält alle gewünschten Felder, Feldbeschriftungen, Hilfebeschreibungen zu jedem Feld.
Die einzelnen Inhalte sind in der Masken-Datei kommentiert.

Hilfe zu Sprachelementen findet man Online unter

"Startseite/Dokumentation/Programmierhandbuch (programmers.pdf)".

Eine gute Möglichkeit ist auch eine ähnliche Maske zu kopieren und ihr den gewünschten Dateinamen zu geben. Dann müssen nur noch die gewünschten Änderungen in der Masken-Datei vorgenommen werden:

~/src/<modulname>/maskdef> cp ../../mod_ess/maskdef/art1info.def artizu.def

Nach dem Editieren der Datei sollte zuerst der Modulname geändert werden.
Anschließend müssen alle Bezeichnungen 'ART1INFO ' in 'ARTIZU' umbenannt werden.

Dies geht im vi am schnellsten und sichersten mit folgendem Befehl:

:%s/ART1INFO/ARTIZU/g

(Andere Editoren arbeiten hier z.B. mit 'suchen und ersetzen')
Außerdem müssen natürlich die Strings geändert werden.
Sie können, wie in Kapitel 4.2.6 beschrieben, in der Datei mxlang.<modulname> definiert werden.

Hinweis:
Vor der Definition OS_SORTLIST...(für Tabellensortierung zuständig) kann für Auswertungen noch ein zusätzlicher Datenbank-Indexeintrag erzeugt werden:

```
/* Code-Ausschnitt: artizu.def */

 OS_INDEXLIST {
      BEZ1_ARTIZU_ARTIZU, 0;       //Feldname, Sortierung aufsteigend
 }

/* Ausschnitt-ENDE */
```

Auf diese Weise wurde eine neue Maske erstellt.
Im nächsten Schritt muss nun noch ein Button generiert werden, über den die Maske erreicht werden kann.

7.4. Erstellen eines Menu-Buttons

In der Datei 'mod_beispiel.def' werden alle Änderungen zum Standardsystem eingetragen.
Hier muss auch der neue Button definiert werden.

Beispiel 11:

Ein Button ARTIKELZUSÄTZE soll im Hauptmenü ('HOMEPAGE') zugefügt werden, über den die neue Maske ARTIKELZUSÄTZE erreicht werden kann.

```
/* Code-Ausschnitt: mod_beispiel.def */

#include '../../include/maskdef/layout_base.deh'

OS_BUTTON  BUTTONARTIZU [MenuButtonLen] OS_READ | OS_WRITE {
            OS_POSITION (MenuSpalte 1, 3);
            OS_TITLE 100005 OS_TITLEPOS OS_PLEFT;
            OS_TITLELEN MenuButtonTitleLen;
            OS_NEXTMASK ADRZU;
}

OS_UPDATEMASK HOMEPAGE {                  //der Button soll in der Maske
      OS_MFIELDLIST {                     Hauptmenue(  HOMEPAGE)
            BUTTONADRZU;                  abgebildet werden.
       }
}

/* Ausschnitt-ENDE */
```

7.5. Einbinden der neuen Maske ins System

Nach dem erfolgreichen Compilieren müssen die Datenbankprozesse gestoppt werden. Anschließend wird, wie in Kapitel 1.5 beschrieben, die Datenbankstruktur mit dem Befehl 'reltool' aktualisiert, und die Datenbankprozesse werden wieder gestartet.

Der neue Button ist nun auf dem Hauptmenü angelegt und über ihn gelangt man in die neue Maske ARTIKELZUSÄTZE.

Hinweis:

Beim Compilieren einer **neuen** Maske, die noch **keine Formulare** beinhaltet, erzeugt der Compiler eine Fehlermeldung

' nothing to merge '

nach dem ersten Durchlauf.
Ein erneuter Aufruf des 'make' erlaubt eine Übersetzung trotzdem.

8. Deinstallation

Beim Einbinden des Modules 'mod_beispiel' wurde eine neue Tabelle ARTIZU (ARTIKEL-ZUSATZINFO) in der Datenbank angelegt und der Tabelle ARTI 'Adressen' ein neues Felde BEZ1 hinzugefügt.

Mit Hilfe des Programms 'reltool' können Tabellen aus der Datenbank gelöscht werden, jedoch keine einzelnen Spalten (Felder). Diese müssen gegebenenfalls manuell per SQL-Anweisung aus der Datenbank gelöscht, oder einfach auf der Oberfläche unsichtbar gemacht werden (Feldeigenschaft NOACCESS).

Die Deinstallation der Tabelle ARTIZU des Moduls mod_beispiel über das Programm 'reltool' wird im Folgenden beschrieben:

1. Anmelden als ESS-Benutzer (z.b. mx001):

   <benutzer_aktuell>@<rechner>:~ su -
   Password:
   <rechner>:~ # su - mx001

2. Das Modul deaktivieren:

   Dafür muss das Modul 'mod_beispiel' aus der Datei ~/src/modules gelöscht oder der Eintrag auf '0' gesetzt werden:

/* Code-Ausschnitt: mod_beispiel.def */

```
mod_base                        1
mod_ess                         1
mod_adressen                    1
mod_webshop                     1
mod_zm                          1
mod_afa                         1
mod_zahl                        1
mod_dtaus                       1
mod_sammel                      1
mod_anzahlung                   1
mod_provision                   1
mod_provision_zugang            0
mod_produktion                  0
mod_bestellvorschlag            1
mod_einheiten                   0
mod_ticket                      1
mod_wiedervorlage               1
mod_connector                   1
mod_zeit                        1
mod_ups                         1
mod_projekt                     1
mod_beispiel                    0  // deaktivieren des Moduls
```

/* Ausschnitt Ende */

3. Aufruf des Compilermechanismus:
   Um 'make' zu starten, muss man als <benutzer> auf dem <rechner> angemeldet sein und sich im Verzeichnis ~/src befinden.

```
mx001@<rechner>:~ # cd src/
mx001@<rechner>:~/src > make mod_beispielclean
```

dieser Aufruf löscht alle Links (z.B. zu Formularen etc.) aus dem Modul und sorgt dafür, dass keine Datei-Leichen mehr vorhanden sind, die eventuell bei späteren Übersetzungen zu Fehlermeldungen führen könnten.
Anschließend wird neu übersetzt:

```
mx001@<rechner>:~/src > make
```

4. Stoppen der ESS-Prozesse:

   Die ESS-Prozesse müssen gestoppt werden, die Datenbank-Prozesse laufen weiter.
   Dafür braucht man die 'root'-Berechtigung:

```
mx001@<rechner>:~ su -
Password:
<rechner>:~ # sh /etc/pentaprise/ess/mxrc stopmxsd
```

5. Entfernen der überflüssigen Tabelle aus der Datenbank:

   Das Programm 'reltool' wird ausgeführt:

```
pg_dump -U pg_mx001 -s MXDBS | reltool -n mxmask.msk -sfxDM -O -
psql MXDBS -U pg_mx001 < reltool.out
```

6. Neustarten der ESS-Prozesse:

   Nach dem erneuten Starten der ESS-Prozesse (als 'root'-Benutzer) kann das Standard-Programm wieder gestartet werden.

```
<rechner>:~ # sh /etc/pentaprise/ess/mxrc start
```

Index