Anwendung von openHBCI mit aqmoney – ein Beispiel zur Nutzung der Bank-Dienste aus der Kommandozeile

Dipl. Ing. Christoph Stockmayer, Stockmayer GmbH, Schwaig

1. Wie es dazu kam

Lange Zeit gab es unter Linux kein Bankprogramm im Stil von T-Online, Quicken oder Zvlight. Man war gezwungen, wollte man nicht online im Web mit seinen Konten arbeiten, auf Windows auszuweichen (was auf die Dauer unbefriedigend ist). Im letzten Jahr schien sich dies zu verbessern mit „*moneyplex*“ der Firma Matrica (www.matrica.de). Wir schöpften Hoffnung und bestellten die Linux-Version (nicht ohne vorher nachzufragen, ob die Software für unsere Linux-Distribution/Version und für unsere Bank funktioniere). Die erste Enttäuschung war, daß SuSE7.2 den USB-Chipkartenleser so nicht unterstützt. Also mußte ein serieller Leser bestellt werden um festzustellen, daß die VR-NetWorld-Card von dieser *moneyplex*-Version noch nicht unterstützt wird (auch die „Diskettenversion“, bei der die Bank die HBCI-Daten direkt zur Verfügung stellt, brach bereits beim Austausch des INI-Briefs ab). Also warteten wir auf Version 5 von *moneyplex*, wo versprochen wurde, diese Chip-Karte zu unterstützen. Inzwischen auf SuSE8.1 hochgerüstet, installierten wir gespannt die neue Version Ende vergangenen Jahres. Nach einem erstaunten Erkennen unserer Chip-Karte kam auch gleich das jähe Ende mit dem Abbruch des Datenaustausches beim Holen der Kontendaten (und das Problem der „Diskettenversion“ hat sich auch nicht geändert). Warum es nicht funktionierte, wurde zunächst bei der Bank versucht herauszubekommen (vermutlich sei die Karte gesperrt, man müsse eine neue bestellen ...), dann beim Hersteller von *moneyplex* (eMails, versuchte Telephonate – keine Antwort). Nun waren bereits 4 Monate vergangen und die Geduld war am Ende (nicht erwähnt die vertane Probierzeit und die Investition in das Programm inklusive 2 Chipkartenleser) – es mußte eine andere Lösung gefunden werden (und ohne Windows). Da kam Anfang des Jahres die Meldung, daß es eine Bibliothek nun gibt, die HBCI unter Linux zur Verfügung stellt: openHBCI. Und mit dieser Bibliothek (und der rudimentären Kommandozeilenanwendung aqmoney) konnte innerhalb von wenigen Tagen eine funktionierende Bankverbindung mit Kontenabfrage, Holen der Kontoauszüge und Überweisungen realisiert werden. Diese Programme werden im Folgenden beschrieben. Natürlich möchte ich noch erwähnen, daß gnucash (www.gnucash.org) den HBCI-Zugriff integriert (die momentane Version bekamen wir aber nicht zum Laufen), daß kopenHBCI und gopenHBCI in Arbeit seien, also in absehbarer Zeit auch unter Linux solche (graphischen) Programme zur Verfügung stehen werden.

2. Benötigte Software

Zunächst wird die neue *openHBCI-Bibliothek* benötigt. Man erhält sie von www.openhbci.de. Möchte man mit Chipkarten arbeiten, ist auch noch die Bibliothek *libchipcard* notwendig, die ebenfalls von der obigen Web-Adresse bezogen werden kann. Dann muß außerdem *SSL* installiert sein. Die Installation erfolgt in bekannter Weise (`./configure; make; make install`), es sei den, man findet ein rpm-Paket (z.B. für SuSE8.1): `rpm -i xxx.rpm`. Das Kommandozeilen-Tool *aqmoney* findet man unter aqmoney.sourceforge.net (allerdings nur in einer Quellversion).

3. Initialschritte

Von der Bank ist z.B. eine sogenannte Diskettenversion erforderlich, bei der man die BLZ (Bankleitzahl), eine ID (die HBCI-Benutzerkennung) und die Webadresse des Bankservers bekommt. Außerdem ist ein INI-Brief der Bank vonnöten, der den Bank-Hashcode enthält, der später verglichen werden muß.

Zunächst ist die IP-Adresse des Bankservers zu ermitteln:

```
host hbci02.fiducia.de
hbci02.fiducia.de. has address 195.162.4.242
```

Mit dieser IP-Adresse (hier z.B. `195.162.4.242`), der HBCI-ID (hier `1234567890123456`) und der BLZ (hier `53290000`) wird ein Benutzer angemeldet:

```
aqmoney –configfile=$HOME/finanz/aqmoney.conf \
  --command="createuser" --user="1234567890123456" \
  --institute="53290000" –medium="$HOME/finanz/daten" \
  --security="rdh" --server="195.162.4.242"
```

Dabei wird in der Datei `aqmoney.conf` die Konfiguration abgelegt und in `daten` die Verschlüsselungsangaben. Außerdem wird ein Passwort abgefragt, das bei künftigen Aktionen notwendig ist. *rdh* steht für die Diskettenlösung , *ddv* für die Chipkarte.

Dann müssen die Keys ausgetauscht werden:

```
aqmoney --configfile=$HOME/finanz/aqmoney.conf \
  --command="getkeys" --user="1234567890123456" \
  --institute="53290000"
aqmoney --configfile=$HOME/finanz/aqmoney.conf \
  --command="sendkeys" --user="1234567890123456" \
  --institute="53290000"
```

In der Fehlerausgabe (auch umzuleiten mit `2> protokoll`) sieht man die Details.

Im nächsten Schritt werden die Hashcodes verglichen und der INI-Brief erzeugt:

```
aqmoney --configfile=$HOME/finanz/aqmoney.conf \
  --command="iniletter" --user="1234567890123456" \
  --institute="53290000" –key="institute"
```

zum Vergleichen mit dem Brief der Bank (was hoffentlich übereinstimmt!) und

```
aqmoney --configfile=$HOME/finanz/aqmoney.conf \
  --command="iniletter" --user="1234567890123456" \
  --institute="53290000" --key="user"
```

zum Erzeugen des eigenen INI-Briefs, der unterschrieben an die Bank zurückgehen muß. Ist die Bank mit den übertragenen Codes einverstanden schaltet sie den Zugang frei. Ist diese Freischaltung erfolgt, kann's losgehen!

Mit

```
aqmoney --configfile=$HOME/finanz/aqmoney.conf \
  --command="acclist"
```

sieht man (etwas versteckt zwischen den Protokollmeldungen) die Konten, die über diesen Zugang verwendet werden können.

4. Kontenabfrage

Die Kontenabfrage (*balance*) kann mithilfe eines kleinen Shellscripts erfolgen (keine Kontonummer zeigt alle Kontenstände; ansonsten ist mit `--account="12345"` die Kontonummer mitzugeben):

```
#!/bin/sh
# Kontenabfrage

PATH=$PATH:/usr/local/aqmoney/bin
TMP=/tmp/aqmoney$$

echo Kontostand
aqmoney –configfile=$HOME/finanz/aqmoney.conf \
  --command="balance" 2> $TMP

grep "(Saldenrückmeldung)" $TMP | sed 's/,/./g' |
awk -F"[:+-]" '
        { if($11 == "C") betr = $12
          else           betr = $12 * (-1)
          printf("BLZ: %9d   Konto: %12d   am: ", $7, $5)
          printf("%2d.%02d.%4d   Stand: %10.2f %3s\n", $7, $5,
             substr($14,7,2), substr($14,5,2),
             substr($14,1,4), betr, $13) }
        '
rm -f $TMP
```

Nach der Eingabe der PIN (Passwort aus dem Initialschritt) sieht die Ausgabe so aus:

```
BLZ: 53290000 Konto: 1234567890 am: 25.01.2003 Stand: 1525.75 EUR
BLZ: 53290000 Konto: 9876543210 am: 25.01.2003 Stand: -133.17 EUR
BLZ: 53290000 Konto:   24681234 am: 25.01.2003 Stand:  483.66 EUR
BLZ: 53290000 Konto:   36924680 am: 25.01.2003 Stand: 2493.94 EUR
```

Übrigends ist die PIN-Abfrage auch automatisierbar, da sie von der Standard-Eingabe erfolgt:

```
echo 454545454 | aqmoney ...
```

was natürlich die Sicherheit verschlechtert, da die PIN irgendwo als Text unverschlüsselt gespeichert werden muß. Auf der anderen Seite sind nun automatische Konten-Abfragen mit `cron` möglich:

```
crontab -e
```

mit der Zeile:

```
0 5 * * * echo 454545454 | $HOME/finanz/kontostand
```

Damit bekommt man in der ersten eMail des Tages den Überblick!

5. Kontoauszüge

Auch für die Kontoauszüge (*turnover* und *dump*) hilft ein Skript, das mit Kontonummer und Startzeit aufgerufen wird (die Auszüge werden gleichzeitig gespeichert und ausgedruckt):

```
#!/bin/sh
# Auszüge
# auszuege konto dd.mm.yyyy

PATH=$PATH:/usr/local/aqmoney/bin
TMP=/tmp/aqmoney$$
AUS=$HOME/finanz/AUSZUEGE/aus`date '+%Y%m%d'`
BLZ=53290000
ZIEL=$HOME/finanz/AUSZUEGE
FORMDAT=/tmp/ausform$$
ERR=0

if [ $# = 2 ]             # überprüfen Argumente
then
   KNTO=$1
   FDAT=$2
   TDAT=`date '+%Y%m%d'`
else
   echo "usage: `basename $0` konto yyyymmdd" >&2
   exit 1
fi

FDAT=`echo $FDAT |         # Datumseingabe in Form yyyymmdd
   awk -F. '{ mo = $2; if(length($2) == 1) mo = "0" $2;
              dy = $1; if(length($1) == 1) dy = "0" $1;
              yr = $3; if(length($3) == 2) yr = "20" $3;
              print yr mo dy }'`

echo Auszüge
                           # Daten holen
aqmoney --configfile=$HOME/finanz/aqmoney.conf \
  --command="turnover" --account="$KNTO" --fromdate="$FDAT" \
  --todate="$TDAT" 2> $TMP
                           # in Datei schreiben
aqmoney --configfile=$HOME/finanz/aqmoney.conf \
  --command="texport" --account="$KNTO" --balance \
  --transactions --fromdate="$FDAT" --todate="$TDAT" \
  --outfile=$AUS --outformat=txt 2>> $TMP
                           # Auszüge aufbereiten
aqmoney --configfile=$HOME/finanz/aqmoney.conf \
  --command="dump" --account="$KNTO" --balance \
  --transactions --fromdate="$FDAT" --todate="$TDAT" |
  sed 's/,/./g' |
  tee $FORMDAT
```

```
                              # Einrücken
sed 's/^/                   /' $FORMDAT | lp

                              # Fehlerprüfung
if grep ERROR $TMP > /dev/null 2>&1
then
  ERR=1
else
  echo ok
fi

rm -f $TMP
exit $ERR
```

6. Überweisung

Für die Überweisungen ist eine kleine Datei notwendig, die alle Transactionsdaten enthält und gleichzeitig auch als Beleg dienen kann:

```
#!/bin/sh
# Überweisung

PATH=$PATH:/usr/local/aqmoney/bin
TMP=/tmp/aqmoney$$
UEB=$HOME/finanz/BELEGE/ueberw`date '+%Y%m%d'`
BLZ=53290000
DAT=$HOME/finanz/kontendaten
ERR=0
gefunden=0

echo Überweisung
echo -e "Von welchem Konto: \c“    # Konto
read KNTO

echo -e "An wen: \c"
read NAME
if [ "$NAME" ]
then
  if grep "^$NAME" $DAT            # gibt es das Ziel schon?
  then
    gefunden=1
  fi
else
  echo "Fehler" >&2
  exit 1
fi

echo -e "Kontonummer: \c"          # Zielkonto
read ZKNTO
if [ "$ZKNTO" ]
then
  :
else
  echo "Fehler" >&2
```

```
  exit 1
fi

echo -e "BLZ: \c"                    # ZielBLZ
read ZBLZ
if [ "$ZBLZ" ]
then
  :
else
  echo "Fehler" >&2
  exit 1
fi

echo -e "Betrag (xxx.yy): \c"       # Überweisungsbetrag
read BETR
if [ "$BETR" ]
then
  :
else
  echo "Fehler" >&2
  exit 1
fi
BETR=`echo "$BETR" | sed 's/\./,/'`

echo -e "Zweck1: \c"                 # Zweck
read ZWECK1
if [ "$ZWECK1" ]
then
  :
else
  echo "Fehler" >&2
  exit 1
fi

echo -e "Zweck2: \c"
read ZWECK2
if [ "$ZWECK2" ]
then
  :
else
  echo "Fehler" >&2
  exit 1
fi

if [ "$gefunden" = 0 ]               # falls neu: merken
then
     echo "$NAME;$ZKNTO;$ZBLZ;$BETR;$ZWECK1;$ZWECK2" >> $DAT
fi
echo $KNTO                           # Kontrolle
echo "$NAME;$ZKNTO;$ZBLZ;$BETR;$ZWECK1;$ZWECK2"
echo -e "ok? \c"
read IN

(                                    # Transaktionsdatei erzeugen
echo "[TRANSACTION]"
echo "institute=\"$BLZ\""
```

```
echo "id=\"$KNTO\""
echo "otherinstitute=\"$ZBLZ\""
echo "otherid=\"$ZKNTO\""
echo "value=\"$BETR:EUR\""
echo "othername=\"$NAME\""
echo "description=\"$ZWECK1\""
echo "description=\"$ZWECK2\""
) > $UEB

                                        # Überweisung durchführen
aqmoney --configfile=$HOME/finanz/aqmoney.conf \
  --command="transfer" --tfile=$UEB 2> $TMP

                                        # Fehlerkontrolle
if grep "Überweisung wurde ausgeführt." $TMP > /dev/null 2>&1
then
  echo ok
else
  ERR=1
fi

rm -f $TMP
exit $ERR
```

Natürlich sind nun weitere Features leicht einbaubar:
- offline-Auswertung der Kontoauszüge und Belege
- Anbindung an ein Buchhaltungsprogramm
- graphische Oberfläche.

Interessant ist aber, daß mit diesen wenigen Komandos und Shell-Zeilen sehr rasch eine funktionierende Bankverbindung realisierbar ist.

7. Adressen/Infos

www.hbci.de	HBCI-Infos
www.openhbci.de	HBCI-Bibliothek und Verweise
aqmoney.sourceforge.net	aqmoney-Kommandozeilen-Programme
nroff -man /usr/local/aqmoney/man/man1/aqmoney.1 | less	Manual-Seite von aqmoney nach Standard-Installation
www.gnucash.org	GNU-Cash-Version mit HBCI-Anbindung
www.matrica.de	kommerzielles HBCI-Programm