Handbuch zu AqBanking 4

Grafik: © 1986 A.Schunke & M.Preuß

(Stand 31.12.2009)

Inhaltsverzeichnis

Änderungen an diesem Dokument

Datum	Autor	Änderung
2009/12/31	M.Preuß	- Kapitel 6.1.1 abgekürzt (Umstieg auf AqBanking 4), empfiehlt nun die Verwendung von *aqbanking-cli updateconf*
2009/12/20	M.Preuß	- Kapitel über Pin-Dateien hinzugefügt
2009/09/05	M.Preuß	- Kapitel über den Import eines EBICS-Benutzers hinzugefügt
2009/09/03	M.Preuß	- Hinweis bei „aqhbci-tool4 adduser“ hinzugefügt: Absoluten Pfad verwenden!
2009/08/22	M.Preuß	- Kapitel 6.2: „Begriffserklärung“ hinzugefügt
2009/08/21	M.Preuß	- Kapitel 10.5.3: „Spaltentausch nach Betrag“ hinzugefügt
2009/08/21	M.Preuß	- Kapitel 6.2.2.3: „Importieren einer RSA-Chipkarte“ hinzugefügt
2009/07/27	M.Preuß	- Unterkapitel zu möglichen Flags bei der EBICS-Einrichtung hinzugefügt
2009/06/27	M.Preuß	- Kapitel 10.1 eingefügt (Beschreibung des Ablageortes von Profilen)
2009/06/10	M.Preuß	- „aqhbci-tool3“ geändert zu „aqhbci-tool4“ - Änderungen bezüglich aqbanking-cli (jetzt Open-Source, bis auf das EBICS-Modul)
2009/01/31	M.Preuß	- Kapitel über globale Optionen von AqBanking-CLI hinzugefügt - ikonto geändert zu Pecunia, Link angepaßt
2009/01/22	M.Preuß	- Kapitel über Verwendung eines HTTP-Proxy mit Gwenhywfar 3.7.0 hinzugefuegt
2008/11/28	M.Preuß	- HBCI-Einrichtung auf der Konsole: Beschreibung der neuen Parameter für das aqhbci-tool Kommando „adduser“ (--rdhtype=x, --hbciversion=xxx) - Hinweis zu GnuCash und AqBanking4
2008/11/25	M.Preuß	- optionaler Schritt bei der Einrichtung von PIN/TAN auf der Konsole hinzugefügt (iTAN-Methode wählen) - Kapitel 9.3 (Anwendungsbeispiel für AqBanking-CLI2) hinzugefügt - AqBanking-CLI2-Kommando „initdb“ hinzugefügt - AqBanking-CLI2-Kommando „dbrecon“ hinzugefügt - Kapitel 3.3: LibOFX wird nicht mehr benötigt
2008/11/08	M.Preuß	- Dokument umbenannt zu „Handbuch zu AqBanking4“ - Anpassen der Beschreibung von AqBanking-CLI (nun in der Version 2, mit Datenbank) - neue Kommandos von AqBanking-CLI-2 hinzugefügt - Dokument unter die Free Art License 1.3 gestellt
2008/09/14	M.Preuß	- Befehl zur Änderung der PIN im PIN/TAN-Verfahren hinzugefügt - RSA-Chipkarte von Matrica als unterstützte Medien hinzugefügt
2008/08/26	M.Preuß	- Kapitel über CSV-Profile hinzugefügt
2008/07/08	M.Preuß	- Kapitel über die Einrichtung von Pin/TAN über die Konsole eingeführt

		- **aqbanking-cli**-Kommandos *addtrans* und *fillgaps* hinzugefügt - Kapitel 2.3: SEPA-Exporter aufgeführt
2008/06/12	M.Preuß	- Kapitel über EBICS als neues Protokoll hinzugefügt - EBICS-Einrichtungskapitel vervollständigt
2008/06/11	M.Preuß	- Kapitel über die EBICS-Einrichtung hinzugefügt
2008/05/02	M.Preuß	- Kapitel über aqbanking-cli hinzugefügt - Kapitel über den Import von bestehenden Schlüsseldateien hinzugefügt - Kapitel 3.6 „AqBanking-CLI“ hinzugefügt - Kapitel 5.2.2.1 „Neue Schlüsseldatei erzeugen“ umbenannt zu „Schlüsseldatei erzeugen“ (Kapitel mit gleichem Thema bei der grafischen Einrichtung heißt ebenso)
2008/03/12	M.Preuß	- Änderung des Namens des Kapitels „Einrichtung von AqBanking“ zu „HBCI-Einrichtung von AqBanking“, da später noch Kapitel zu EBICS hinzukommen sollen - Neue Anwendungen in die Liste aufgenommen (OrgaMon und iKonto) - Hinzufuegen eines Kapitels zur Verwendung von Proxies - Hinzufügen eines Kapitels zur konsolenbasierten HBCI-Einrichtung
2008/01/29	M.Preuß	- Anpassung an die Änderungen im AqHBCI-Einrichtungs-Assistenten („erweiterte Einstellungen“) - Hinzufügen dieser Änderungsliste - CSV-Profil für VRNetworld hinzugefügt

1 Copyleft

Dieses Dokument ist frei, Sie können es gemäß den Festlegungen der Lizenz „Art Libre“ (***Free Art License*** Version 1.3) weiterverbreiten und/oder modifizieren.
Von dieser Lizenz ausgenommen ist das Titelbild: Es darf ausschließlich in diesem Dokument verwendet und nur mit diesem verbreitet werden.

Ein Exemplar dieser Lizenz findet sich auf der Website von „Copyleft Attitude“ (http://www.artlibre.org) sowie auch auf anderen Websites.

Wer Änderungen an diesem Dokument vornimmt, wird gebeten anschließend folgendes zu tun:
- die Änderungen in der obigen Tabelle am Anfang einfügen (*Änderungen an diesem Dokument*)
- die geänderte Version an mich, den Autor dieses Dokuments (Martin Preuß), senden, damit ich die Änderungen aufnehmen und auf der Website von AqBanking veröffentlichen kann

2 Vorbemerkungen zum Dokument

Kommandos, die in der Konsole einzugeben sind, werden wie im folgenden Beispiel dargestellt:

martin@gwenhywfar:~$ **cp -a .banking .aqbanking**

Ihre Eingabe ist dabei **fett** gedruckt, die Ausgaben des Systems oder aufgerufenen Programmes hingegen sind *kursiv*.

3 AqBanking

AqBanking ist eine Erweiterung für Finanz- und Buchhaltungsanwendungen geschrieben von Martin Preuß.

Es besteht aus 3 Anteilen.

3.1 Online-Banking

AqBanking unterstützt verschiedene Online-Banking-Protokolle

3.1.1 HBCI

Das ist das in Deutschland von den meisten Banken unterstützte Verfahren zum Online-Banking.

Um HBCI verwenden zu können müssen Sie in aller Regel bei Ihrer Bank einen Antrag stellen.

AqBanking unterstützt die folgenden HBCI-Sicherheitsmedien:

- DDV-Chipkarte (DDV0 und DDV1)
- RSA-Chipkarte von Matrica (RDH1)
- RDH-Schlüsseldatei (in einem eigenen Format)
- Pin/TAN gemäß Erweiterung für HBCI 2.2 (einschließlich iTAN)

Unterstützt werden die folgenden Geschäftsvorfälle:

- Abruf von Umsätzen (*Kontoauszüge*)
- Abruf von Kontoständen
- Abrufen, anlegen, entfernen und ändern von Daueraufträgen
- Abrufen, anlegen, entfernen und ändern von terminierten Überweisungen
- Überweisung
- Lastschrift

3.1.2 OFX Direct Connect

Dieses Protokoll wird vornehmlich in den USA, Kanada, Grossbrittanien und vereinzelt Österreich verwendet.

Unterstützt werden die folgenden Geschäftsvorfälle:

- Abruf von Umsätzen (*Kontoauszüge*)
- Abruf von Kontoständen
- Abruf von Wertpapier-Informationen

3.1.3 EBICS

EBICS ist der Nachfolger von FTAM und soll eigentlich seit dem 1.1.2008 von allen deutschen Banken angeboten werden. Praktisch ist das noch nicht der Fall, die meisten Banken dürften es aber inzwischen im Programm haben.

EBICS richtet sich ausschließlich an Geschäftskunden und hat für Privatkunden keine Bedeutung. Die Gründe liegen zum einen in den meist höheren Dienst-Nutzungsgebühren der Banken, den im Vergleich zu HBCI-Software deutlich teureren Anwendungen sowie dem geringeren Spektrum an Geschäftsvorfällen.

Im Gegensatz dazu läßt sich der Zugriff über EBICS aber deutlich besser automatisieren, wodurch es gerade für Betriebe eine gute Alternative zum HBCI darstellt.

Seit Version 0.9.12beta existiert vom Tool ***aqbanking-cli*** eine gesonderte Version mit EBICS-Unterstützung.

Der EBICS-Client von ***aqbanking-cli*** beherrscht allgemeine Uploads und Downloads von Dateien und implementiert außerdem eine Schnittstelle zu AqBanking, die aus dem Tool **aqbanking-cli** heraus benutzbar ist.

Somit stehen die gängigen Kommandos von ***aqbanking-cli*** (für Überweisungen, Lastschriften, Umsatzabruf) auch für EBICS zur Verfügung.

Zur Anwendung kommen derzeit ausschließlich Schlüsseldateien, wie sie auch vom HBCI-Modul verwendet werden, wenngleich mit größeren Schlüssellängen.

3.2 Bank-, Länder und Währungsinformationen

AqBanking enthält Bankinformationen für:

- Deutschland
- USA
- Österreich
- Schweiz

Es enthält außerdem Finanz-Informationen (Ländername, Namen der Landeswährungen, ISO-Landescodes, ISO-Währungscodes etc).

Für Deutschland können außerdem Kontonummern und Bankleitzahlen überprüft werden. Damit scheitert dann keine Überweisung mehr an Tippfehlern.

3.3 Import und Export (Finanzdatenformate)

AqBanking bringt eine Reihe von Importern und Exportern für einige gängige Finanzdateiformate mit. Unter anderem:

- DTAUS
- SWIFT (MT940 und 942)
- OFX und OFC
- OpenHBCI1 (Speicherformat von OpenHBCI1)
- ERI (wird von niederländischen Banken verwendet)

- CSV: Comma-Separated Values, gängiges Format für Spread-Sheet Programme. Das Format der CSV-Datei kann vollständig eingestellt werden. Mitgeliefert werden unter anderem vorbereitete Profile für:
  - American Express Card
  - AqMoney1 und AqMoney2
  - Comdirect
  - MijnPostbank.nl
  - OP Pankki
  - Sparkasse Aachen
  - Steiermärkische Sparkassen
  - T-Online Banking-Modul
  - VR Networld
  - weitere Profile können leicht innerhalb weniger Minuten hinzugefügt werden.

- SEPA (bisher nur Export für pain.001.001.02, SEPA-Überweisungen)

4 Anwendungen für AqBanking

Inzwischen gibt es bereits Anwendungen, die mit AqBanking3 arbeiten.

4.1 KmyMoney

Für die aktuelle CVS-Version von KmyMoney gibt es unter http://www.aqbanking.de/ ein externes Modul, welches **AqBanking3** verwendet.

Die letzte stabile Version von KMyMoney verwendet noch **AqBanking2**, die nächsten Versionen sollen aber ebenfalls **AqBanking3** verwenden können.

Das entsprechende Modul arbeitet derzeit allerdings noch nicht mit **AqBanking4**.

KMyMoney kann Umsätze und den Kontostand abrufen.

4.2 QBankManager

QBankManager ist ebenfalls auf http://www.aqbanking.de/ verfügbar und arbeitet inzwischen ausschließlich mit AqBanking4.

4.3 GnuCash

GnuCash verwendet bisher noch AqBanking2, kann also mit der aktuellen Version von AqBanking noch nicht arbeiten.

Christian Stimming vom GnuCash-Team hat zwar bereits mit der Umstellung begonnen, sucht aber noch nach Mitstreitern für diese Aufgabe.

(http://www.gnucash.org/)

Update: Aktuelle Versionen von GnuCash verwenden inzwischen bereits AqBanking3. Beachten Sie bei Verwendung einer solchen Version die Hinweise im ***Kapitel über den Umstieg von AqBanking2 zu AqBanking3***.

Update: Die SVN-Version von GnuCash arbeitet inzwischen auch schon mit **AqBanking4**.

4.4 OrgaMon

OrgaMon ist ein freies Software-System zur Bewältigung aller typischen IT - Anforderungen eines modernen Unternehmens.

(http://orgamon.org/)

4.5 Pecunia

Eine native Open-Source Anwendung zur Verwaltung von Finanzen unter MacOSX.

(http://www.pecuniabanking.de/Home.html)

4.6 AqBanking-CLI

Hierbei handelt es sich um eine Konsolen-Anwendung für AqBanking3. Sie erlaubt den Abruf von Kontoständen und Umsätzen und das Einreichen von Überweisungen, Lastschriften und Auslandsüberweisungen.

(http://www.aquamaniac.de/sites/aqbanking/cli.php)

AqBanking-CLI gibt es in 2 Versionen:

- ***Open Source Edition*** (bietet **HBCI**)
- ***EBICS Edition*** (kommerzielle Nutzung, bietet **EBICS**)

5 Kontakt

Auf der Webseite http://www.aqbanking.de/ finden Sie Informationen rund um AqBanking. Unter dem dort vorhandenen Link *Contact* finden Sie Hinweise auf Mailinglisten und Bugtracker.

6 HBCI-Einrichtung von AqBanking

6.1 Umstieg auf AqBanking 4

6.1.1 Vorbereiten von AqBanking

Seit Version 4 enthält AqBanking das Kommandozeilen-Tool *aqbanking-cli*. Mit dieser Anwendung kann man unter anderem die Konfiguration aus früheren Version von AqBanking importieren, um sie auch mit aktuellen Versionen zu verwenden.
Dazu öffnen Sie am besten eine Konsole (unter KDE z.B. *Konsole* oder das *Terminal* unter Gnome) und geben das folgende Kommando ein (nur das **fettgedruckte**):

```
martin@gwenhywfar:~$ aqbanking-cli updateconf
```

6.1.2 Vorbereiten der Schlüsseldatei

Sollten Sie bisher eine Schlüsseldatei verwenden, müssen Sie diese unter Umständen aktualisieren. Das ist insbesondere dann der Fall, wenn Sie diese Datei noch mit OpenHBCI erstellt haben.

Wichtig 1: Anwendungen, die **AqBanking2** verwenden, können weiterhin mit dieser Schlüsseldatei arbeiten, auch nach dem Aktualisieren. **OpenHBCI** hingegen wird sehr wahrscheinlich diese Datei **nicht mehr lesen können**.

Wichtig 2: Bitte legen Sie unbedingt vorher eine Sicherheitskopie Ihrer Schlüsseldatei an!!

Führen Sie zum Aktualisieren in einer Konsole das folgende Kommando aus:

```
martin@gwenhywfar:/tmp$ gct-tool update -t ohbci -n test.medium
```

Hinter „**-n**“ müssen Sie den Pfad **Ihrer** Schlüsseldatei angeben, in diesem Beispiel ist es „*test.medium*“.

Daraufhin werden Sie nach dem aktuellen Paßwort der Schlüsseldatei gefragt. Anschließend wird die Datei eingelesen. Ist das Einlesen erfolgreich, werden Sie erneut zur Eingabe eines Paßwortes aufgefordert.

Hierbei handelt es sich nun um das *neue* Paßwort, welches Sie im nächsten Schritt bestätigen müssen. Sie können also bei dieser Gelegenheit das Paßwort ändern.

Nach diesem Schritt ist **AqBanking3** in der Lage, auch sehr alte Schlüsseldateien zu lesen.

6.2 Begriffserklärungen

Im Zusammenhang mit HBCI treten einige Begriffe auf, die an dieser Stelle etwas genauer erklärt werden sollen.

6.2.1 Sicherheitsmedium

Ein Sicherheitsmedium speichert elektronische Schlüssel und dient somit der Chiffrierung, Dechiffrierung, Signatur und Signaturprüfung.

Grundsätzlich unterscheidet man zwei Typen:

- Schlüsseldateien
- Chipkarten

6.2.2 Benutzerkennung, Kundenkennung, VR-Kennung

HBCI definiert Benutzerkennung und Kundenkennung. Bei vielen Banken ist die Benutzerkennung gleich der Kundenkennung.

Die Idee dahinter war, daß ein Benutzer gegenüber der Bank entweder als Privatperson auftreten kann oder aber als Mitarbeiter einer Firma. Im ersten Fall hätte die Person nur Zugriff auf ihre eigenen, persönlichen Konten, als Mitarbeiter aber eventuell auch auf die Konten der Firma.

Somit definiert die Kundenkennung also die Rolle des Benutzers. Je nach Rolle unterscheiden sich dann unter Umständen die Berechtigungen sowie die Liste der Konten, auf die der Benutzer zugreifen darf.

Bei Instituten, die an die GAD angeschlossen sind, gibt es zudem noch die sogenannte VR-Kennung. Dies ist eine lange Folge von Buchstaben und Ziffern, welche man in den entsprechenden Einstellungen anstelle der Kundenkennung eingeben kann.

6.3 Vollständige Neueinrichtung

6.3.1 Mit grafischer Oberfläche

6.3.1.1 Allgemein

Schritt 1: Einrichtungs-Assistenten aufrufen

Sie erreichen den Einrichtungs-Assistenten aus QBankManager heraus über den Menüpunkt *Ändern/Einstellungen*.

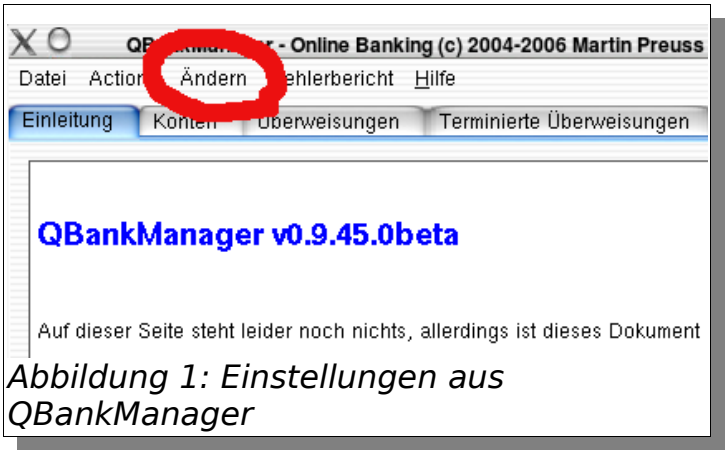

Abbildung 1: Einstellungen aus QBankManager

In KmyMoney erreichen Sie den Assistenten über das Menü *Online Banking/Konfiguriere Online-Banking*.

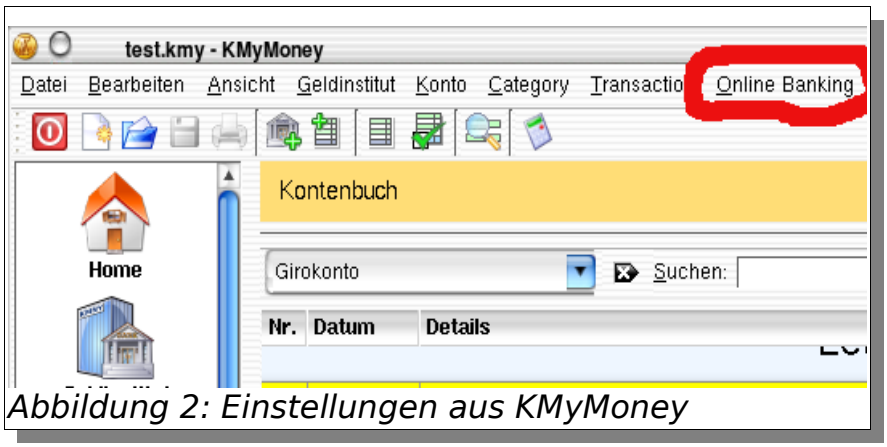

Abbildung 2: Einstellungen aus KMyMoney

Schritt 2: Benutzer-Seite aufrufen

Der Assistent startet mit dem folgenden Bildschirm:

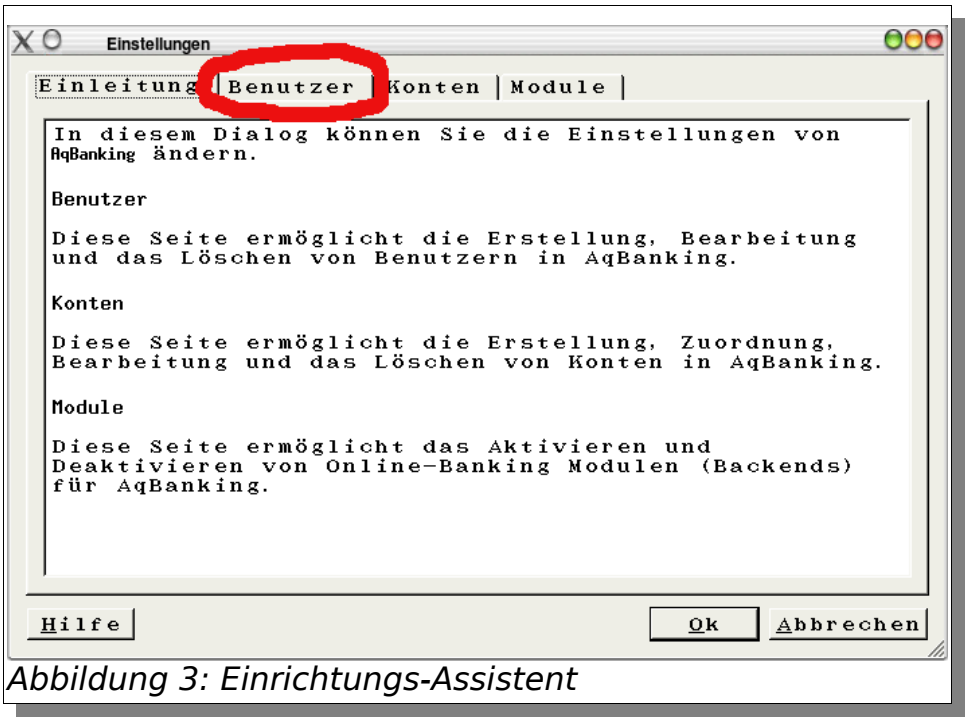

Abbildung 3: Einrichtungs-Assistent

Um HBCI verwenden zu können müssen Sie zunächst einen Benutzer anlegen. Wählen Sie dazu die *Benutzer*-Seite des Assistenten, in dem Sie wie oben angezeigt auf *Benutzer* klicken.

Schritt 3: Neuer Benutzer

Es erscheint das folgende Bild:

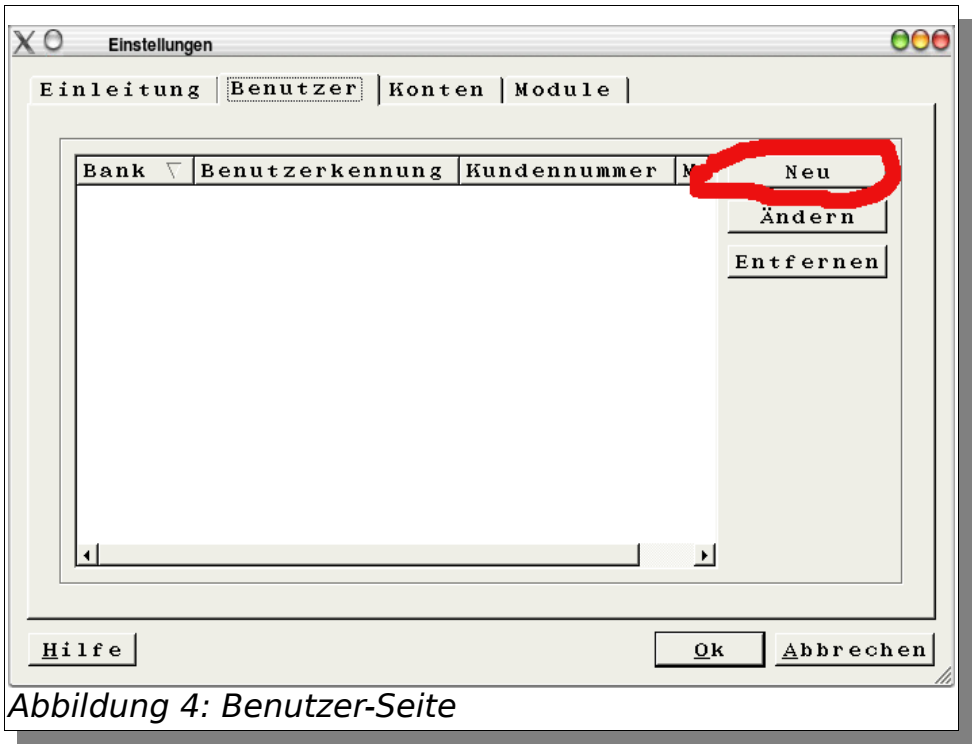

Abbildung 4: Benutzer-Seite

Um einen neuen Benutzer anzulegen, klicken Sie auf *Neu*.

Schritt 4: Banking-Modul auswählen

AqBanking unterstützt mehrere Homebanking-Protokolle.

Für Online-Banking in *Deutschland* müssen Sie das *HBCI*-Modul auswählen.

HBCI steht für ***H***ome***b***anking ***C***omputer ***I***nterface und ist wird von den meisten deutschen Banken angeboten.

Bitte bedenken Sie, daß Sie den Zugang zu HBCI bei Ihrer Bank beantragen müssen. Erst dann bekommen Sie die Zugangsdaten, die Sie in den nächsten Schritten benötigen.

Wählen Sie nun also das Modul *AqHBCI* aus und klicken Sie auf *Ok*.

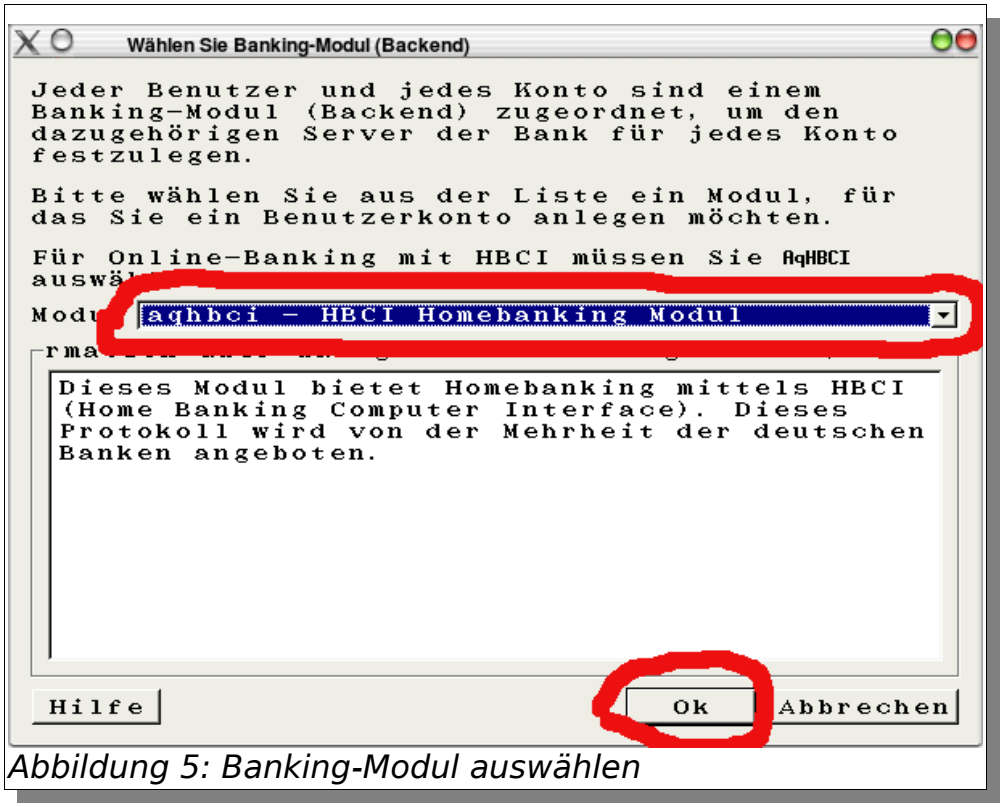

Abbildung 5: Banking-Modul auswählen

Schritt 5: Auswählen der nächsten Schritte

Sie sollten nun den folgenden Bildschirm sehen:

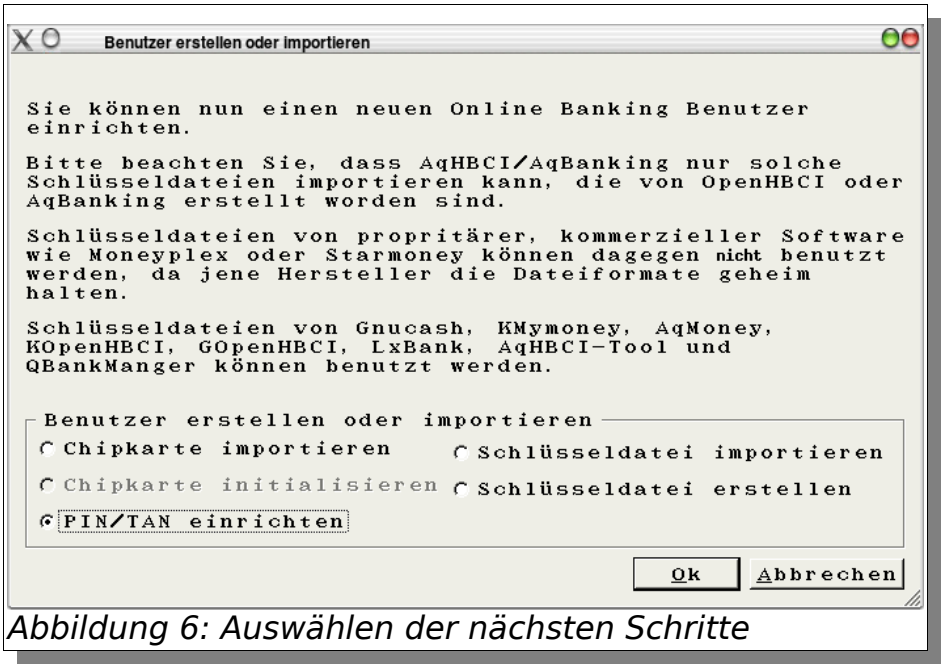

Abbildung 6: Auswählen der nächsten Schritte

Sie sehen hier eine Auswahl an weiteren Schritten.

1) *Chipkarte importieren*:

   Diesen Punkt können Sie wählen, wenn Sie eine Chipkarte haben, die Sie bereits mit anderen Programmen verwendet haben.

   Bitte beachten Sie, daß AqBanking leider nur solche Karten unterstützen kann, für die die Spezifikationen frei erhältlich sind. Dies ist beispielsweise der Fall bei DDV-Karten, wie sie von vielen Sparkassen verwendet werden.

   Probieren Sie diesen Punkt im Zweifelsfall mir Ihrer Chipkarte, im schlimmsten Fall bekommen Sie eine Meldung, die besagt, daß diese Karte nicht unterstützt wird.

2) *Chipkarte initialisieren*:

   Dieser Punkt ist noch nicht implementiert.

3) *Pin/TAN einrichten*:

   Falls Sie auf Ihr Konto mit anderen Anwendungen mittels Pin und TAN zugreifen können, sollten Sie diesen Punkt auswählen.

4) *Schlüsseldatei importieren*:

   Sie können hiermit bereits mit OpenHBCI oder früheren Versionen von AqBanking verwendete Schlüsseldateien weiterverwenden.

   Bitte beachten Sie, daß Schlüsseldateien proprietärer Anwendungen (Moneyplex, StarMoney etc) **nicht** verwendet werden können, da die Hersteller die Formate dieser Dateien nicht öffentlich zugänglich machen.

5) *Schlüsseldatei erstellen*:

   Dieser Punkt erlaubt das Erzeugen von Schlüsseldateien. Diese können allerdings bisher nur mit Anwendungen verwendet werden, die AqBanking verwenden. Das liegt nicht an AqBanking, denn das Format unserer Schlüsseldateien ist im Gegensatz zu dem anderer Anbieter offen.

Wählen Sie hier also aus, was Sie als nächstes tun möchten und klicken Sie dann auf *Ok*.

6.3.1.2 Chipkarte importieren

Schritt 1: Prüfen der Karte

Sobald Sie *Ok* geklickt versucht AqBanking, den Typen der Chipkarte zu ermitteln. Dazu greift es auf den Kartenleser zu.

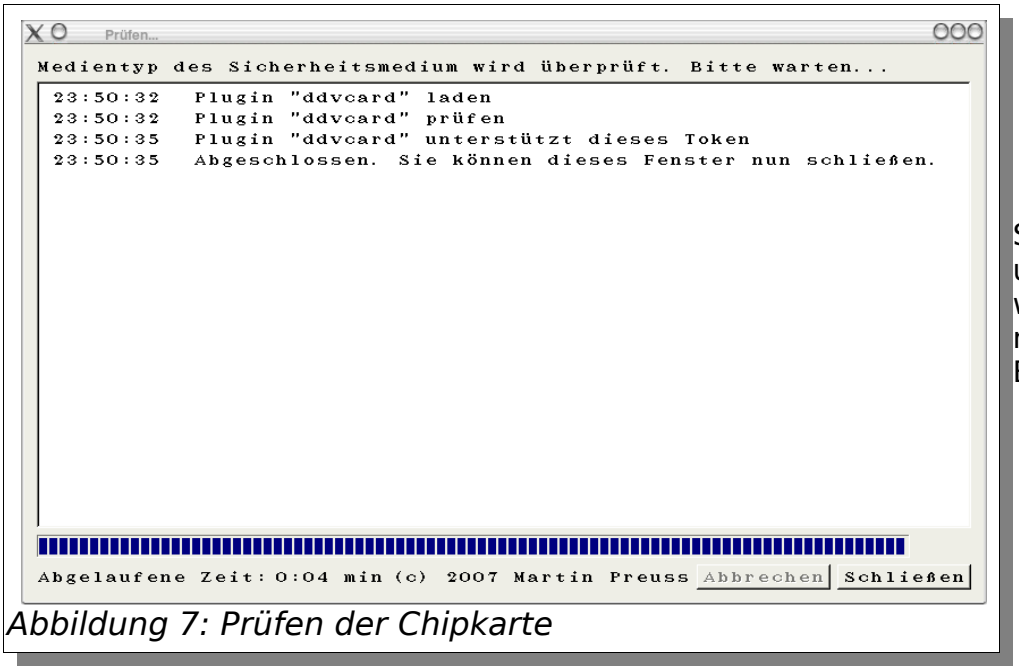

Abbildung 7: Prüfen der Chipkarte

Sollte die Karte unterstützt werden sehen Sie nun diesen Bildschirm:

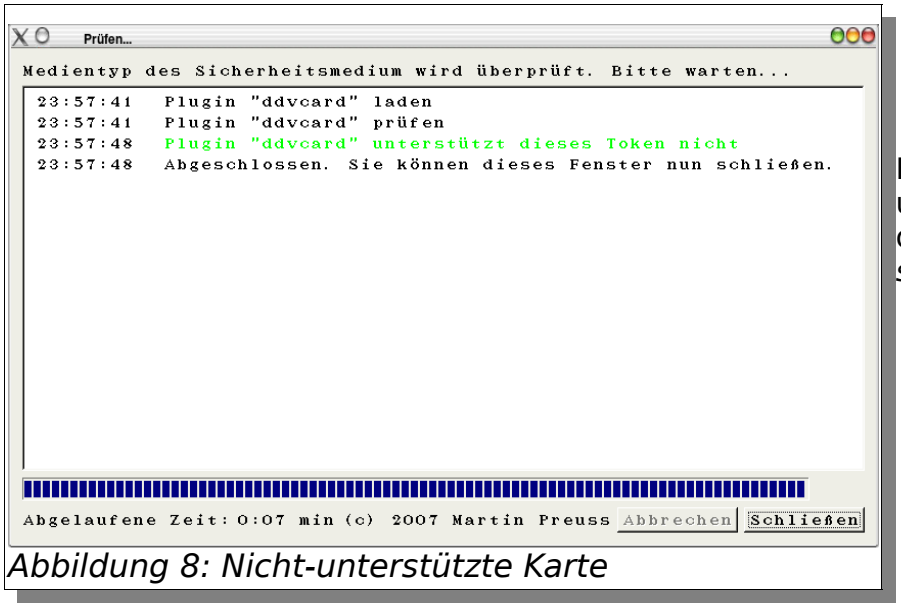

Abbildung 8: Nicht-unterstützte Karte

Falls Ihre Karte nicht unterstützt wird, sieht der Bildschirm eventuell *so* aus:

Schritt 2: Starten des Imports

Wir gehen davon aus, daß Ihre Karte untertützt wird. Daher wird Ihnen das folgende Fenster präsentiert. Klicken Sie auf *Next* um diesen Vorgang durchzuführen.

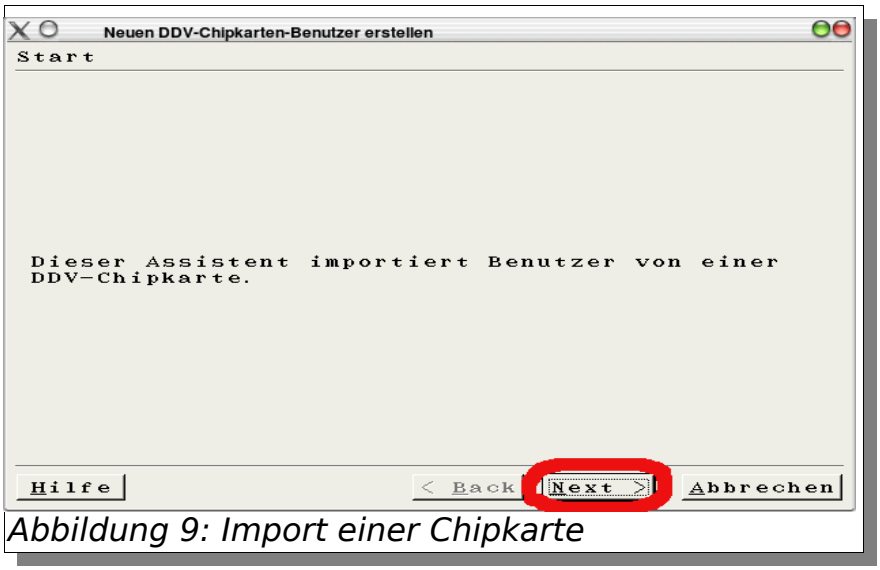

Abbildung 9: Import einer Chipkarte

Schritt 3: Eingabe der Zugangsdaten

Für diesen Schritt benötigen Sie Ihre Unterlagen von der Bank, falls die benötigten Informationen nicht auf der Chipkarte gespeichert sind.

Sie bekommen einen Bildschirm zu sehen, der in etwa wie der folgende aussieht:

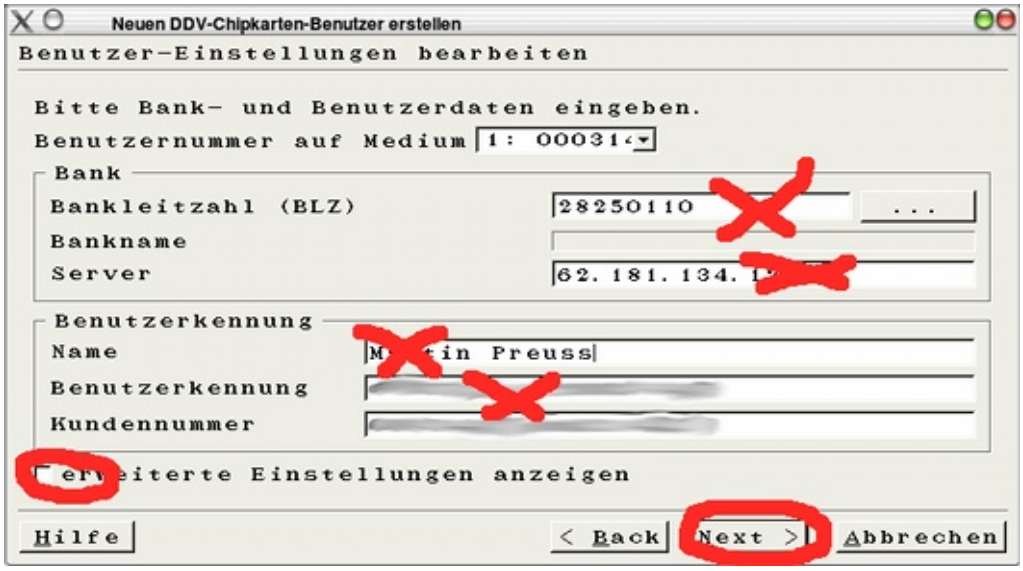

Abbildung 10: Eingabe der Benutzerdaten

An den Stellen, an denen hier ein rotes Kreuz gesetzt ist, müssen zwingend Angaben gemacht werden.

Eventuell sind auf Ihrer Karte bereits die nötigen Daten vorhanden. Sie können beispielsweise oben im Feld *Benutzernummer auf Medium* unter Umständen zwischen mehreren bereits vorgegebenen Einstellungen wählen. Die Pflichtfelder müssen Sie dann gegebenenfalls ergänzen.

- Bankleitzahl: Diese Nummer finden Sie unter anderem auf Ihren Kontoauszügen
- Server: Diese Angabe finden Sie in den Unterlagen, die Ihre Bank Ihnen überlassen hat. Falls nicht, versucht AqBanking, diese Information seiner internen Datenbank zu entnehmen.
- Name: Hier müssen Sie Ihren vollständigen Namen angeben, und zwar so, wie er bei Ihrer Bank als Kontoeigentümer hinterlegt ist (z.B. *Martin Preuss*)
- Benutzerkennung: Diese finden Sie ebenfalls in den Unterlagen von der Bank
- Kundennummer: Falls Sie diese nicht in Ihren Unterlagen von der Bank finden können, lassen Sie dieses Feld leer. AqBanking wird dann zwar noch einmal nachfragen (weil manche Banken dieses Feld wirklich benötigen), aber Sie können das dann immer noch bestätigen.

Wenn Sie die Checkbox unten links („erweitere Einstellungen anzeigen“) aktivieren, erscheinen weitere Einstellungen, die normalerweise nicht geändert werden müssen:

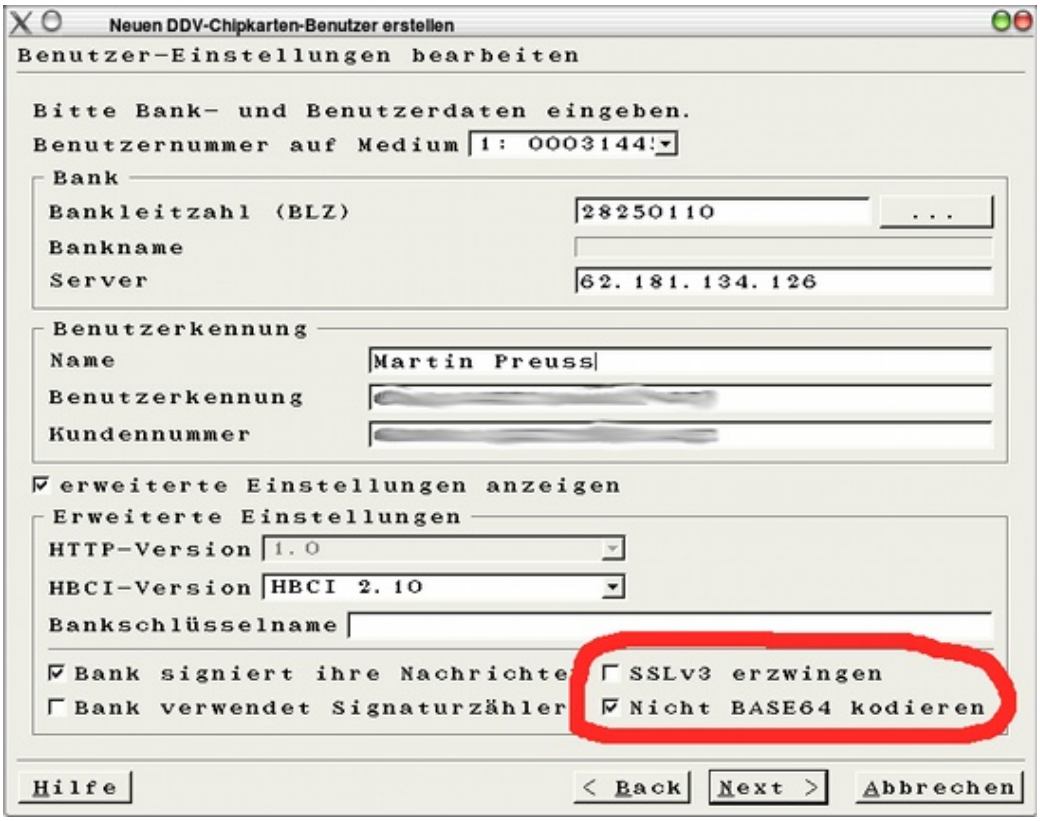

Abbildung 11: Erweiterte Einstellungen

Zu den erweiterten Einstellungen gehören:

- Bankschlüsselname: Diese Feld lassen Sie bitte frei.
- HBCI-Version: Nicht alle Banken unterstützen alle HBCI-Versionen. Wählen Sie hier zunächst eine aus und probieren Sie dann im Fehlerfall andere Versionen (Sie können später von anderen Seiten des Assistenten an diese Stelle zurückkehren).
- SSLv3 erzwingen: Dies ist mit manchen Banken nötig (nur im PIN/TAN-Modus; vor allem Netbank und Sparda-Banken), wird aber von AqBanking automatisch bei Bedarf gesetzt.
- Nicht BASE64 kodieren (nur im PIN/TAN-Modus): Dies ist z.B. nötig mit der **Apobank**

Schritt 4: Kontenliste abrufen

Diesen Bildschirm bekommen Sie nun angezeigt:

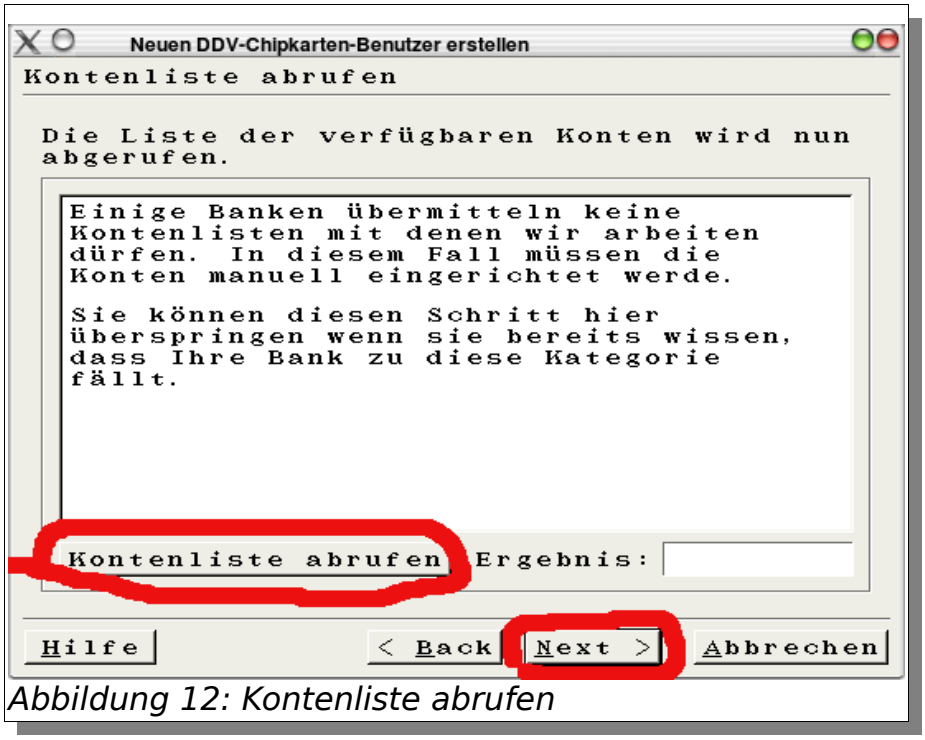

Abbildung 12: Kontenliste abrufen

Klicken Sie hier auf *Kontenliste abrufen*.

Nicht alle Banken untersützen diesen Vorgang, daher können Sie auch im Fehlerfall zur nächsten Seite weitergehen (durch klicken auf *Next*).

Falls Ihre Bank diesen Vorgang jedoch unterstützt – und das tun die meisten – ersparen Sie sich später die manuelle Eingabe der Kontoinformationen.

Im Erfolgsfall bekommen Sie dieses Fenster angezeigt:

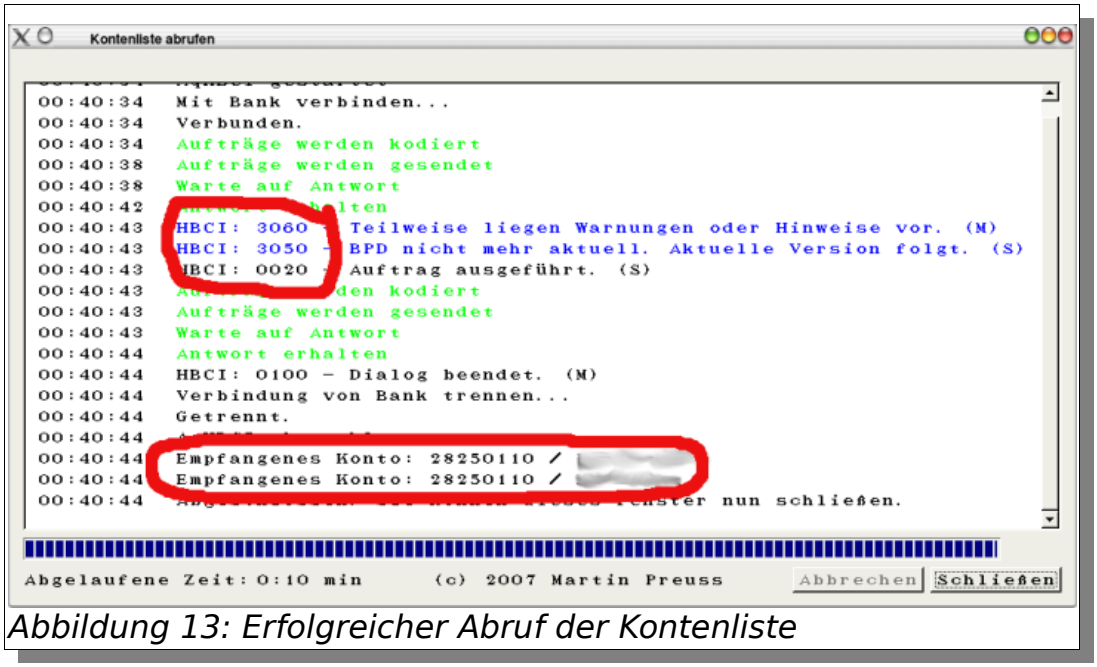

Abbildung 13: Erfolgreicher Abruf der Kontenliste

Dies ist auch ein gutes Beispiel für das Fortschrittsfenster von AqBanking. Es zeigt Warnungen in **blau** an, weniger wichtige Informationen in **grün**, normale Informationen in **schwarz** und Fehler in **rot**.

Es zeigt außerdem die abgelaufene Zeit (als Aktivitätsanzeige) sowie den Fortschritt des aktuellen Vorganges.

Sie sehen an diesem Beispiel, daß zwei Konten empfangen wurden.

Schließen Sie nun dieses Fortschrittsfenster und klicken Sie *Next* im Assistenten.

Schritt 5: Abschluß der Einrichtung

Wenn bis hierher alles erfolgreich verlief, sollten Sie den folgenden Bildschirm erhalten:

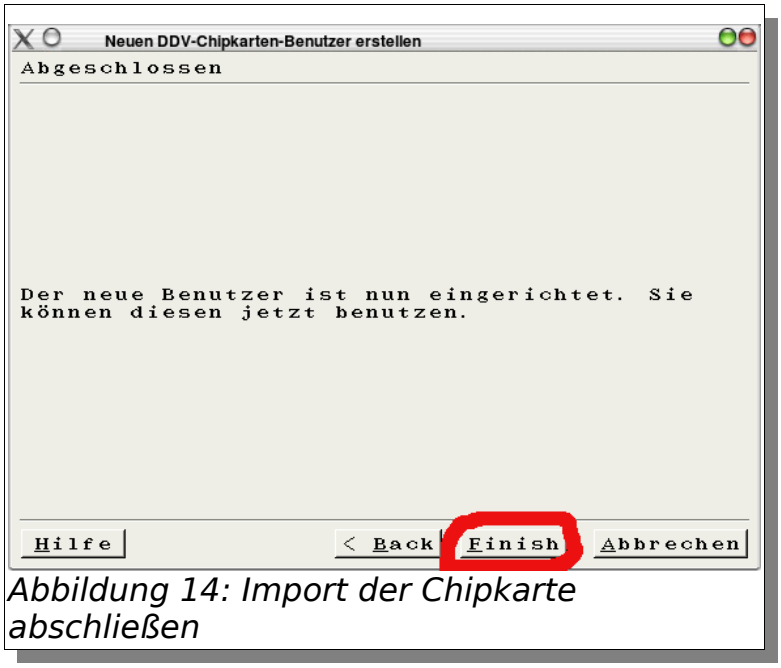

Abbildung 14: Import der Chipkarte abschließen

Sie müssen hier nur noch *Finish* anklicken, um den Import abzuschließen.

Im Benutzerfenster des ersten Dialog-Fensters sollten Sie nun den neu angelegten Benutzer sehen können:

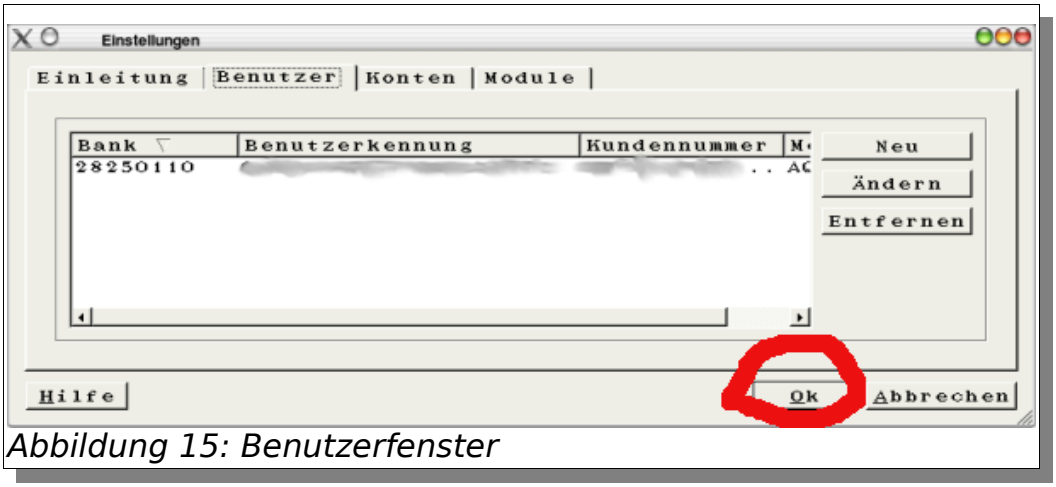

Abbildung 15: Benutzerfenster

6.3.1.3 Pin/TAN einrichten

Bedenken Sie, daß es sich hierbei um Pin/TAN über HBCI handelt, dieses Verfahren hat nichts mit dem *Web-Banking* zu tun, welches manche Banken zusätzlich anbieten, und welches üblicherweise auch mit Pin und TAN arbeitet.

Für Pin/TAN über HBCI müssen Sie normalerweise einen gesonderten Antrag bei Ihrer Bank stellen.

Schritt 1: Zugangsdaten eingeben

Im Pin/TAN-Modus wird kein CryptToken verwendet. Außerdem ist hier die zu verwendende HBCI-Version auf HBCI 2.2 bzw. FinTS 3.00 festgelegt, da PIN/TAN nur für diese Versionen spezifiziert ist.

Daher sind auf dem entsprechenden Bildschirm diese beiden Felder nicht änderbar:

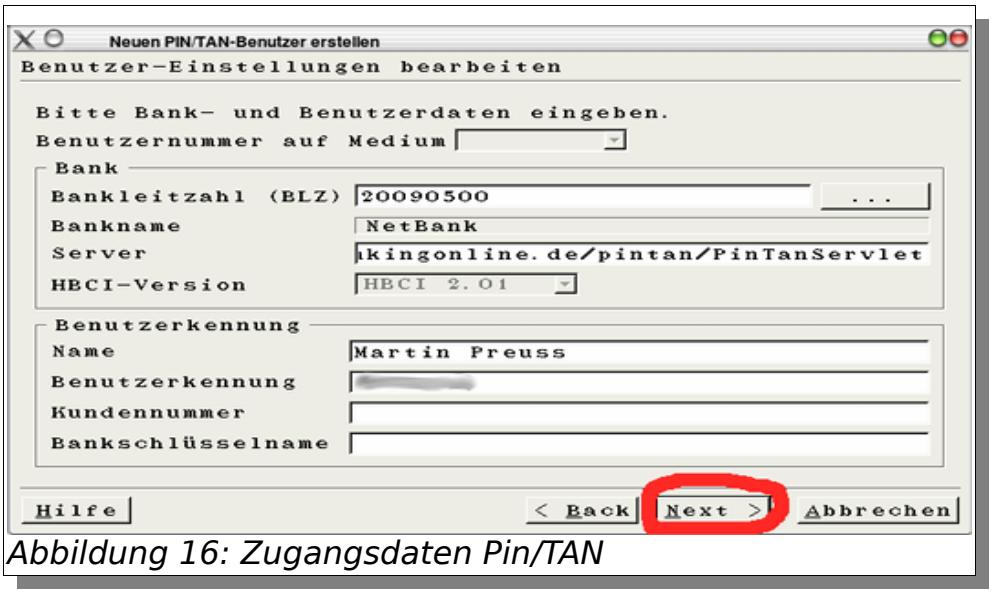

Abbildung 16: Zugangsdaten Pin/TAN

Die anderen Einstellungen sind genauso vorzunehmen, wie beim *Importieren einer Chipkarte* beschrieben.

Wichtig ist hier noch, daß unter *Server* die korrekte URL eingegeben wird, also beispielsweise *https://www.bankserver.org/pintan/PinTanServlet*.

Diese Einstellung können Sie Ihren Unterlagen von der Bank entnehmen.

Schritt 2: Abrufen des SSL-Zertifikates

In diesem Schritt wird eine Verbindung zum Server aufgebaut und es wird versucht das SSL-Zertifikat des Servers abzurufen.

Diese Zertifikate stellen sicher, daß Sie tatsächlich mit dem Rechner Ihrer Bank verbunden werden.

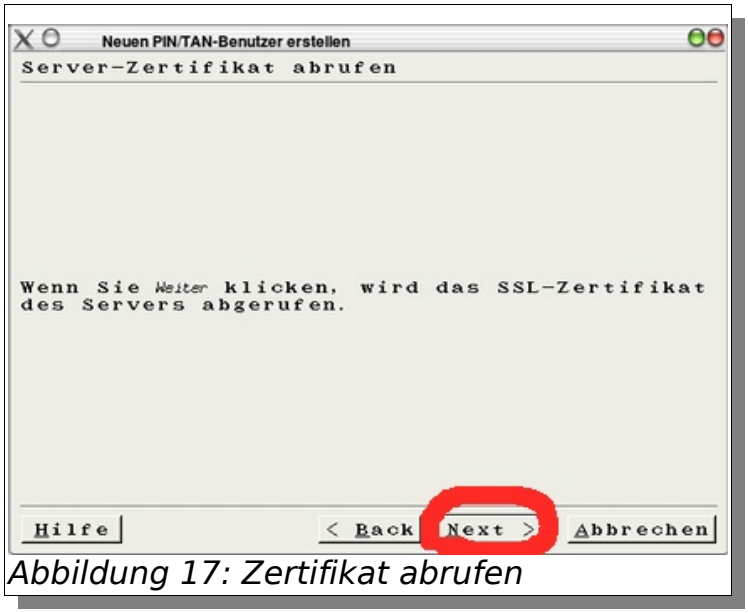

Abbildung 17: Zertifikat abrufen

Sollte die Verbindung zustande kommen, wird Ihnen das folgende Fenster präsentiert:

Abbildung 18: Zertifikat empfangen

Besonderes Augenmerk sollten Sie auf die Zeile *Status* legen. Manche Banken verwenden selbstsignierte Zertifikate. In diesem Fall erscheint an dieser Stelle eine andere Meldung, und Sie sollten sich genau überlegen, ob Sie das Zertifikat akzeptieren wollen.

Schritt 3: Abrufen der Systemkennung

Für die Kommunikation mit der Bank wird eine sogenannte *Systemkennung* benötigt. Diese wird für jede Anwendung erteilt und muß nun abgerufen werden.

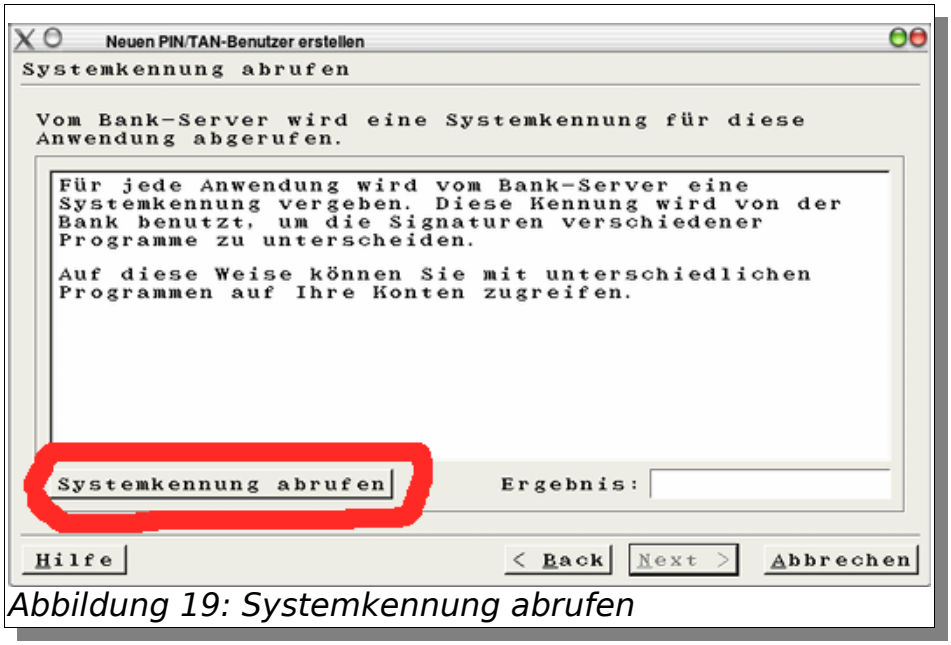

Abbildung 19: Systemkennung abrufen

Klicken Sie auf *Systemkennung abrufen* um den Vorgang zu starten. Erst, wenn die Systemkennung erfolgreich abgerufen werden konnte, kann der *Next*-Knopf geklickt werden.

Für den Vorgang wird ein Fortschrittsfenster geöffnet, in dem Sie den Fortschritt mitverfolgen können. Je nach Server muß *AqBanking* eventuell mehrfach mit der Bank verbinden, um die nötigen Informationen abzurufen.

Im *Ergebnis*-Feld sehen Sie das Resultat.

Schritt 4: Abrufen der Kontenliste

Siehe hierzu *Schritt 4* im Kapitel *Chipkarte importieren*.

Anschließend ist der Benutzer vollständig eingerichtet.

6.3.1.4 Schlüsseldatei erzeugen

Dieser Assistent erlaubt Ihnen eine neue Schlüsseldatei anzulegen. Diese Datei kann anschließend nur von Programmen verwendet werden, die Online-Banking über AqBanking betreiben (GnuCash, KMyMoney, QbankManager).

Insbesondere proprietäre Anwendungen wie Moneyplex oder StarMoney können diese Dateien nicht verwenden. Das liegt allerdings nicht an AqBanking, denn das Format und die Struktur der Datei ist offen zugänglich.

Schritt 1: Dateinamen wählen

In diesem Fenster können Sie den Dateinamen wählen. Die Datei darf noch nicht existieren (um eine existierende Datei zu verwenden wählen Sie stattdessen *Schlüsseldatei importieren*).

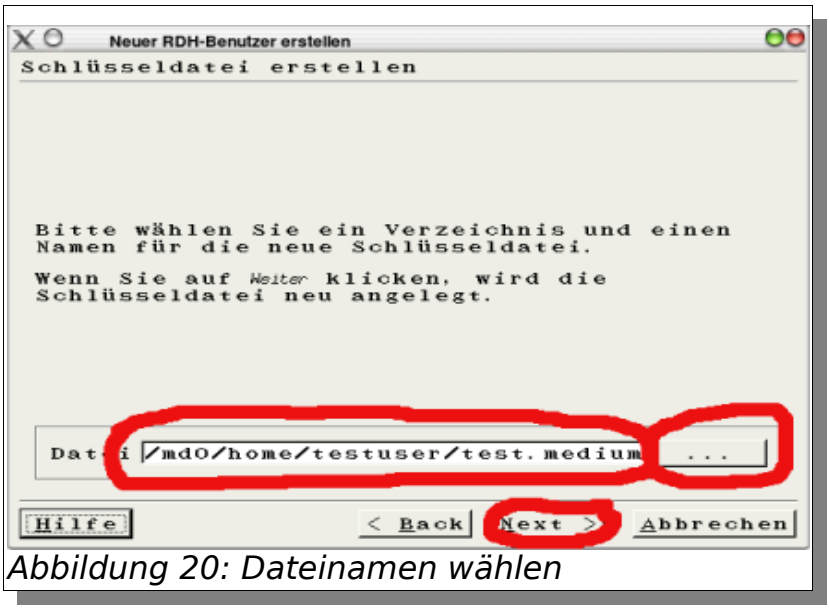

Abbildung 20: Dateinamen wählen

Sie können entweder den Pfad und Namen direkt angeben oder Sie klicken auf den rechts markierten Knopf um einen Dateidialog zu öffnen.

Schritt 2: Passwort wählen

Sie müssen nun ein Passwort für die neue Datei wählen. Bitte wählen Sie ein sicheres Passwort, d.h. eines, welches Buchstaben, Ziffern und Sonderzeichen enthält.

Sie sollten allerdings bei den Sonderzeichen keine Umlaute verwenden sondern nur solche Zeichen, die sich auf den Zifferntasten befinden (so wie „$%&/()=_?+*-:;.,“).

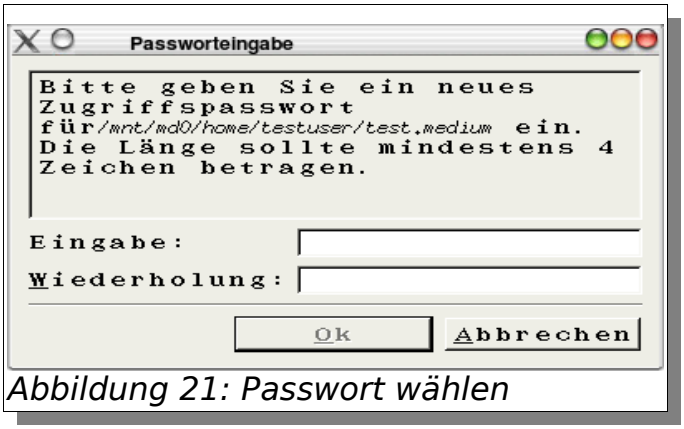

Abbildung 21: Passwort wählen

Zur Sicherheit müssen Sie das Passwort wiederholen (falls Sie sich einmal vertippen sollten).

Schritt 2: Zugangsdaten eingeben

Die Eingabe der Zugangsdaten verläuft ebenso wie unter *Chipkarte importieren* beschrieben.

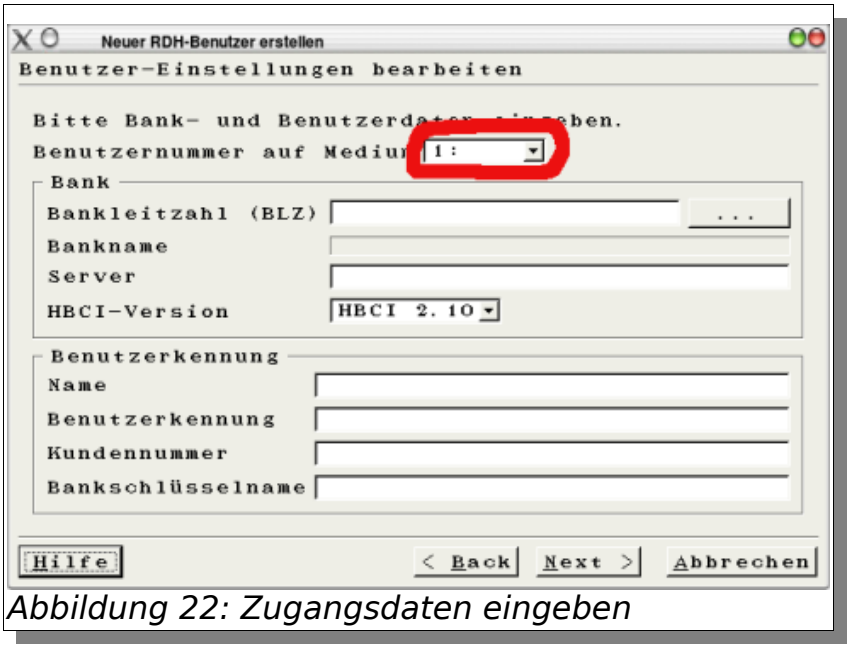

Abbildung 22: Zugangsdaten eingeben

Es gibt hier allerdings eine Besonderheit: Eine neu erzeugte Schlüsseldatei speichert immer nur einen Benutzer.

Schritt 3: Öffentliche Schlüssel des Servers abrufen

Im HBCI-Protokoll werden Nachrichten im Normalfall digital signiert und verschlüsselt übertragen.

Dafür sind insgesamt 4 Schlüssel vorgesehen:

- geheimer Signierschlüssel des Benutzers
- geheimer Crypto-Schlüssel des Benutzers (wird verwendet, um ankommende Nachrichten zu entschlüsseln)
- öffentlicher Signierschlüssel der Bank (manche Banken signieren ihre Nachrichten nicht, daher wird in diesem Fall auch kein solcher Schlüssel verwendet)
- öffentlicher Crypto-Schlüssel der Bank (wird verwendet, um Nachrichten an die Bank zu verschlüsseln)

Hier findet das Prinzip der asymmetrischen Verschlüsselung Anwendung. Hierbei besteht ein Schlüssel aus zwei Anteilen:

- geheimer Anteil: Diesen darf nur der Besitzer des Schlüssels kennen. Sobald er dritten zugänglich wird, ist der Schlüssel kompromittiert und muß ausgetauscht werden
- öffentlicher Anteil: Diesen darf jeder kennen, der Nachrichten mit Ihnen austauschen will (in diesem Fall: Die Bank)

Wenn Sie also eine Nachricht an die Bank verschlüsseln wollen, verwenden Sie dazu den bekannten, öffentlichen Crypto-Schlüssel der Bank. Die so verschlüsselte Nachricht kann dann nur noch von der Bank entschlüsselt werden, weil nur diese den dazu notwendigen geheimen Anteil des Schlüssels kennt.

Ebenso verhält es sich mit dem Signierschlüssel: Eine digital Unterschrift wird unter Verwendung des geheimen Anteiles des Schlüssels erzeugt. Anschließend genügt der öffentliche Anteil um die Signatur zu überprüfen.

Eine digitale Signatur beweist, daß die unterschriebene Nachricht tatsächlich vom vermeintlichen Sender stammt. Damit kann die Bank beispielsweise beweisen, daß eine Nachricht tatsächlich von Ihnen stammt.

Leider bieten manche Banken Ihnen aber nicht den umgekehrten Weg: In diesem Fall unterzeichnet die Bank ihre eigenen Nachrichten nicht, somit haben Sie im Prinzip nicht die Möglichkeit einwandfrei festzustellen, ob die Nachricht tatsächlich von der Bank stammt.

Glücklicherweise unterzeichnen aber die meisten Banken ihre Nachrichten.

Zum Abruf der öffentlichen Schlüssel der Bank wird Ihnen das folgende Fenster präsentiert:

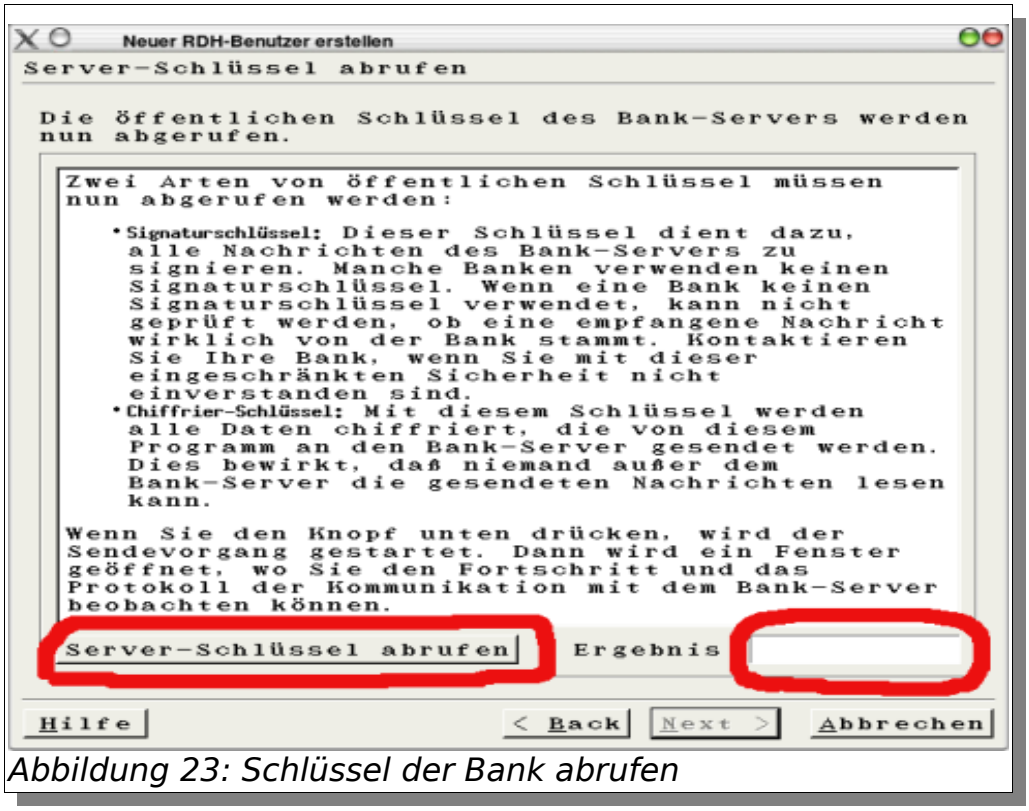

Abbildung 23: Schlüssel der Bank abrufen

Sobald Sie auf *Server-Schlüssel abrufen* klicken, wird ein Fortschrittsfenster geöffnet, welches Ihnen den Fortschritt dieser Aktion und eventuelle Hinweise oder Fehlermeldungen anzeigt.

Sie werden hier unter Umständen eine Warnung sehen, daß die Signatur der Bank nicht überprüft werden kann. Das ist in dieser Situation normal, denn Sie rufen ja gerade erst die Schlüssel der Bank ab.

Wenn das Ergebnis positiv ist, gelangen Sie mir Klick auf *Next* zum nächsten Schritt der Einrichtung.

Schritt 4: Schlüssel der Bank überprüfen

Sie müssen nun die im vorigen Schritt abgerufenen Schlüssel überprüfen. Dazu haben Sie vermutlich von der Bank einen Brief bekommen, in dem der sogenannte *Hashwert* über den Schlüssel abgedruckt ist.

Es handelt sich dabei um eine Folge von Ziffern und Buchstaben. In dem mit *X* markierten Fenster können Sie nach dem Herunterblättern diesen Hashwert ablesen und müssen diesen mit den Unterlagen der Bank vergleichen.

Manche Banken veröffentlichen diesen Wert auf auf ihrer Webseite.

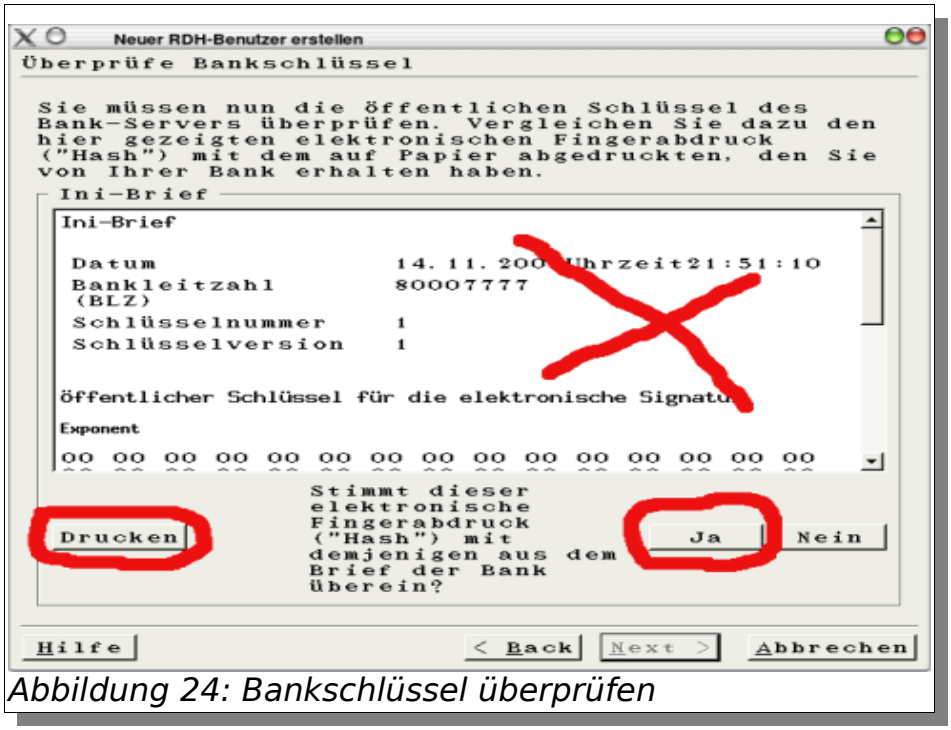

Abbildung 24: Bankschlüssel überprüfen

Sollte der Wert übereinstimmen, klicken Sie auf *Ja*. Damit werden die weiteren Schritte freigeschaltet.

Sollte dieser Wert nicht übereinstimmen, melden Sie sich unbedingt bei Ihrer Bank!

Wichtig: Sie dürfen die Schlüssel nicht akzeptieren, wenn der Wert nicht stimmt!!!

Schritt 5: Eigene Schlüssel erzeugen

Mit diesem Schritt werden neue Schlüssel des Benutzers erzeugt.

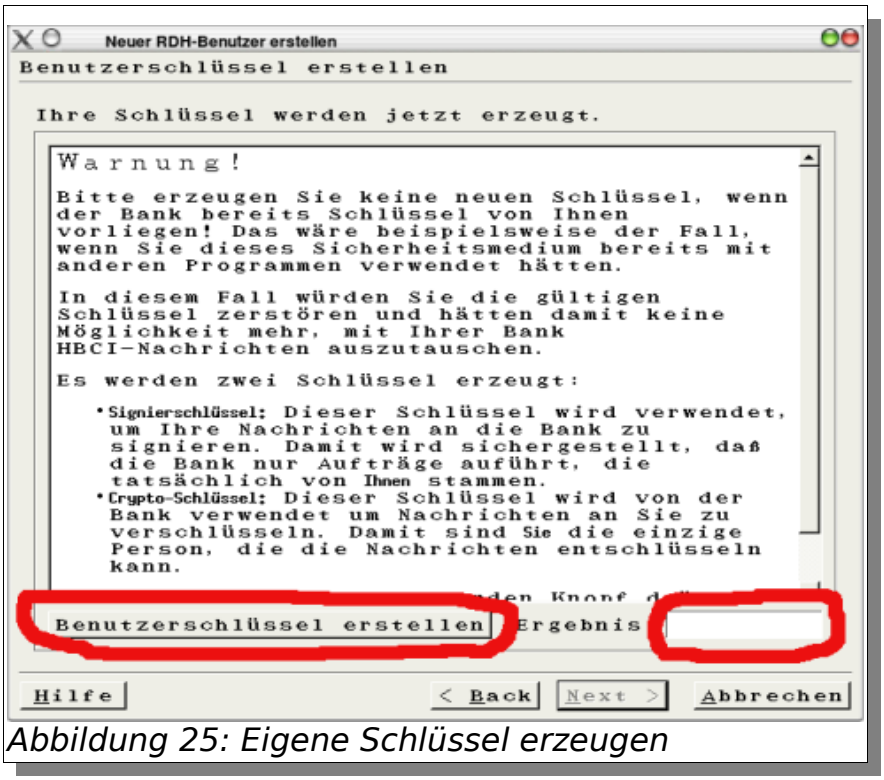

Abbildung 25: Eigene Schlüssel erzeugen

Wenn Sie hier auf *Benutzerschlüssel erstellen* klicken, werden die Schlüssel erzeugt und in der Schlüsseldatei gespeichert. Im Erfolgsfall gelangen Sie dann mit Klick auf *Next* zum nächsten Schritt.

Schritt 6: Eigene Schlüssel übermitteln

Die im vorigen Schritt erzeugten Schlüssel müssen nun an die Bank übertragen werden.

Wichtig: Sie dürfen nur dann neue Schlüssel an die Bank senden, wenn der Bank noch keine Schlüssel von Ihnen vorliegen!

Das wäre beispielsweise dann der Fall, wenn Sie unter dieser Benutzerkennung bereits Online-Banking mit einer anderen Anwendung betreiben.

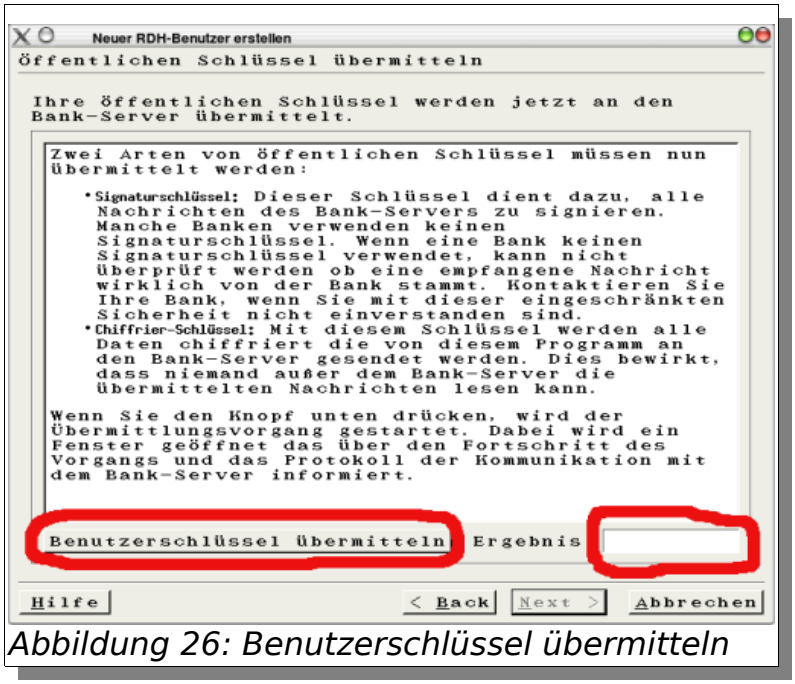

Abbildung 26: Benutzerschlüssel übermitteln

Sobald Sie auf *Benutzerschlüssel übermitteln* klicken, öffnet sich wieder ein Fortschrittsfenster, welches Sie über den Fortgang der Aktion unterrichtet.

Im Idealfall erhalten Sie dabei Meldungen wie im folgenden:

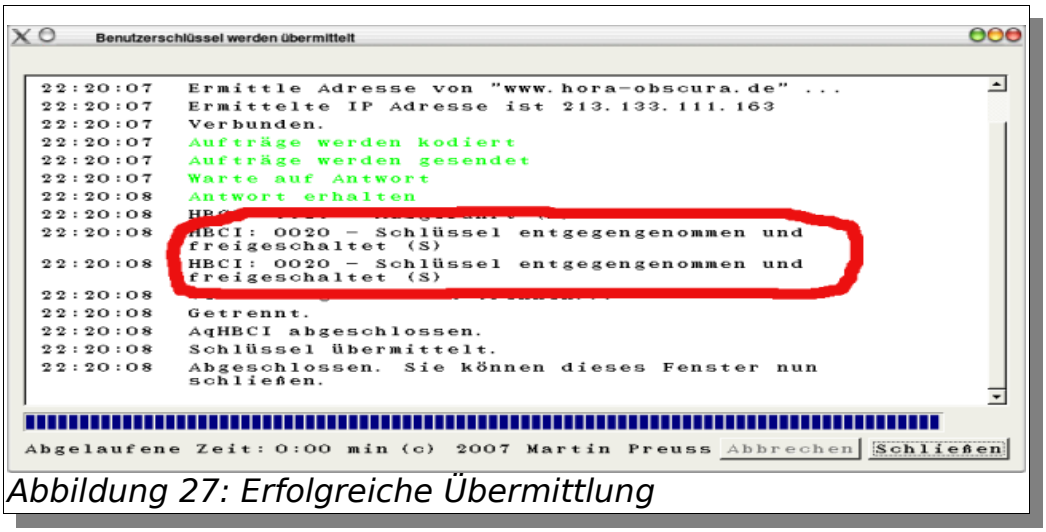

Abbildung 27: Erfolgreiche Übermittlung

Schritt 7: INI-Brief drucken

Zu den gerade erzeugten und gesendeten Schlüssel wird in diesem Schritt ein Dokument erstellt, welches Sie ausdrucken und per Post oder Fax an die Bank schicken.

Wichtig: Vergessen Sie nicht, den Brief handschriftlich zu unterzeichnen!

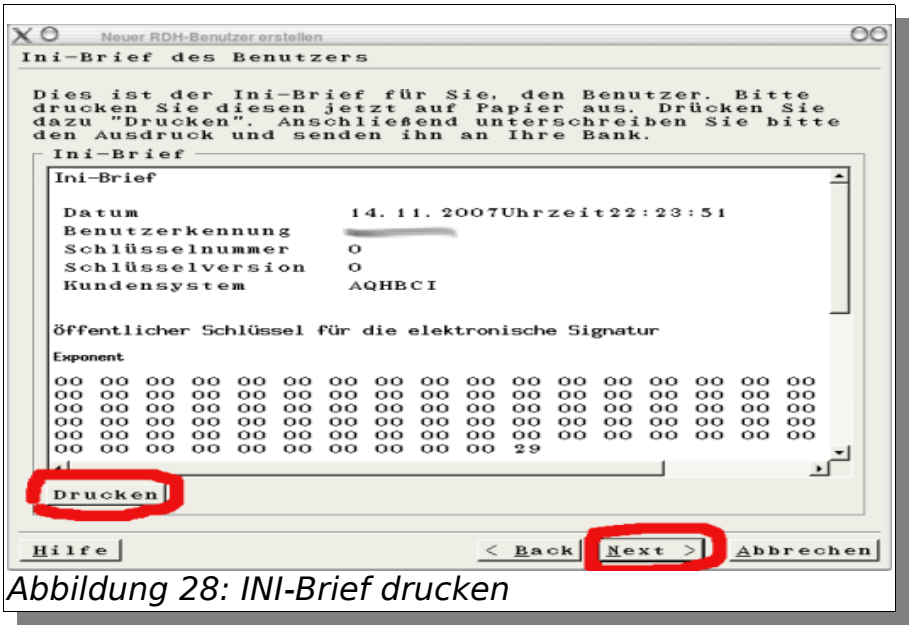

Abbildung 28: INI-Brief drucken

Dieses Fenster kennen Sie bereits aus einem der vorigen Schritte.

Sobald Sie den Brief ausgedruckt und verschickt haben, ist der erste Abschnitt der Einrichtung beendet.

Sie müssen nun auf die Antwort der Bank warten, denn die Bank vergleicht den Hashwert über Ihre gesendeten Schlüssel mit dem in diesem INI-Brief um sicherzugehen, daß es sich tatsächlich um Ihre Schlüssel handelt.

Sobald die Bank diese Überprüfung vorgenommen hat, werden Sie – meiste ebenfalls per Post – über das Ergebnis informiert.

Erst dann können Sie die Einrichtung fortsetzen.

Schritt 8: Einrichtung fortsetzen

Nachdem die Bank Ihnen die Korrektheit der Schlüsselübertragung bestätigt hat, können Sie nun die Einrichtung fortsetzen.

Dazu öffnen Sie wieder die Benutzerseite, wählen den neu angelegten Benutzer aus und klicken auf *Ändern*.

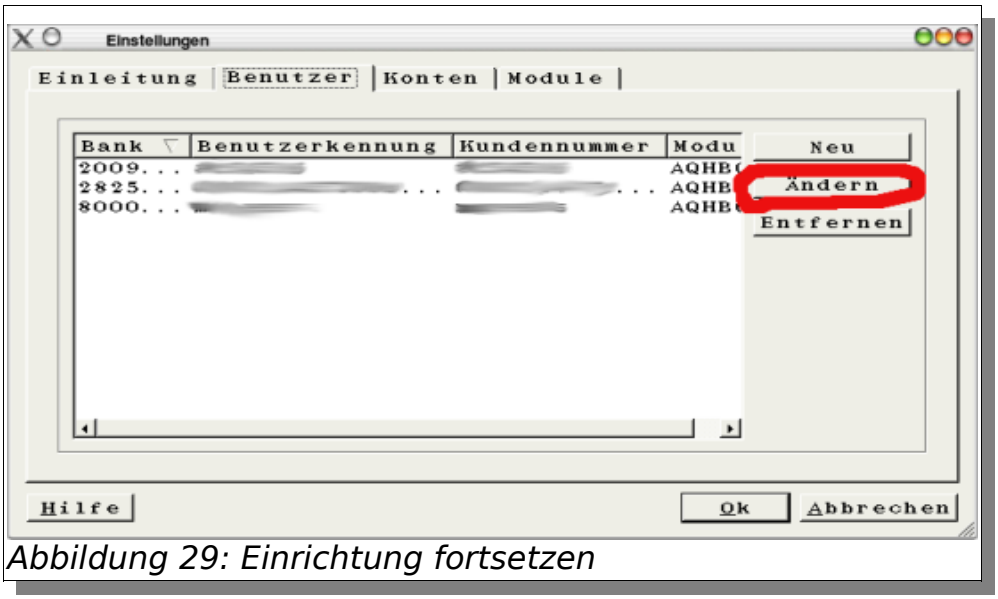

Abbildung 29: Einrichtung fortsetzen

In dem sich dann öffnenden Fenster wählen Sie dann den Reiter HBCI.

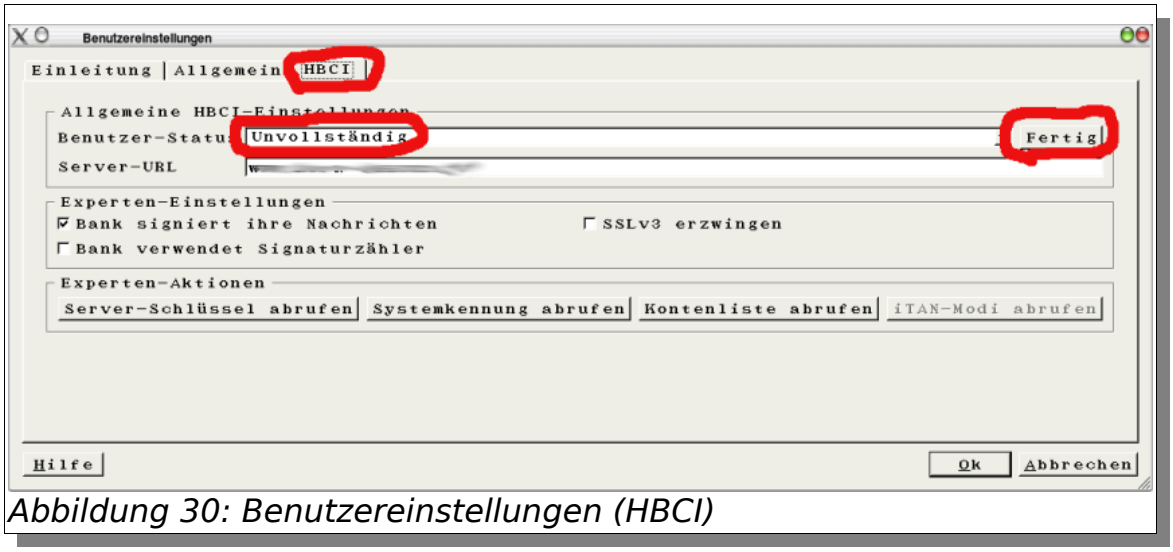

Abbildung 30: Benutzereinstellungen (HBCI)

Sie sehen hier unter *Benutzer-Status*, daß die Einrichtung noch unvollständig ist. Um die Einrichtung abzuschließen, klicken Sie auf *Fertig*.

Schritt 9: Systemkennung abrufen

Diesen Schritt können Sie im Kapitel *Pin/Tan einrichten* nachlesen.

Schritt 10: Kontenliste abrufen

Diesen Schritt können Sie im Kapitel *Chipkarte importieren* nachlesen.

Schritt 11: Einrichtung abgeschlossen

Sie sehen nun im Benutzerfenster, daß sich der Status des Benutzers zu *Aktiv* geändert hat.

6.3.1.5 Schlüsseldatei importieren

Mit diesem Menüpunkt können Sie eine Schlüsseldatei, die Sie bisher mit OpenHBCI- oder AqBanking-basierten Anwendungen benutzt haben, weiterhin verwenden.

Bei diesen Anwendunge handelt es sich insbesondere um GnuCash, KmyMoney, LxLinux, QbankManager, AqMoney, GOpenHBCI und KOpenHBCI.

Leider können wir keine Dateien verwenden, die von proprietären Anwendungen erzeugt wurden (StarMoney, Moneyplex etc), da diese Hersteller die Formate ihrer Schlüsseldateien nicht herausgeben.

Das Format der von AqBanking und OpenHBCI verwendeten Dateien hingegen ist öffentlich zugänglich, und freie Projekte wie HBCI4Java verwenden diese Informationen auch um ihrerseits Importer für unsere Schlüsseldateien zu erstellen.

Schritt 1: Datei auswählen

Zunächst muß die Datei angegeben werden.

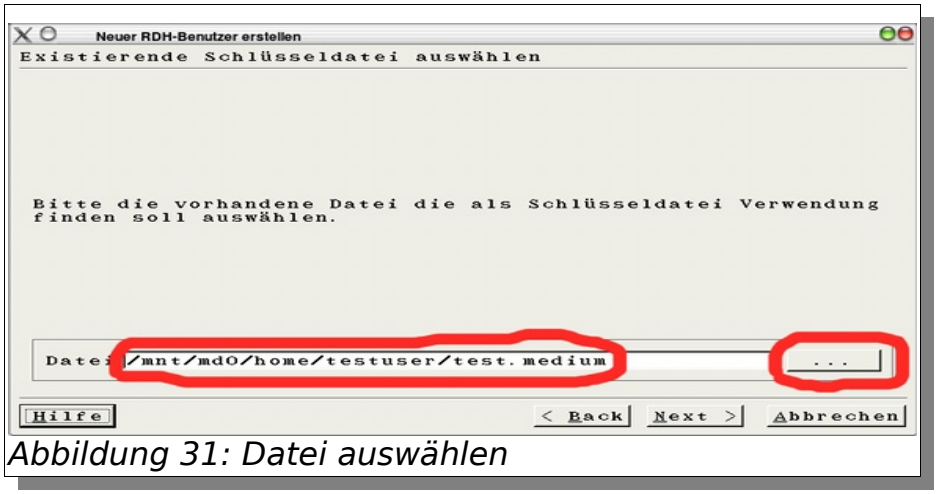

Abbildung 31: Datei auswählen

Sie können hier entweder direkt die Datei angeben oder mit klick auf den rechtsseitigen Button den Dateiauswahl-Dialog verwenden.

Schritt 2: Schlüsseldatei überprüfen

AqBanking versucht in diesem Schritt herauszufinden, welches Modul diese Datei unterstützt. Bisher gibt es nur **ein** Modul, aber sobald wir die Informationen für andere Dateiformate haben, können wir prinzipiell auch weitere Formate unterstützen.

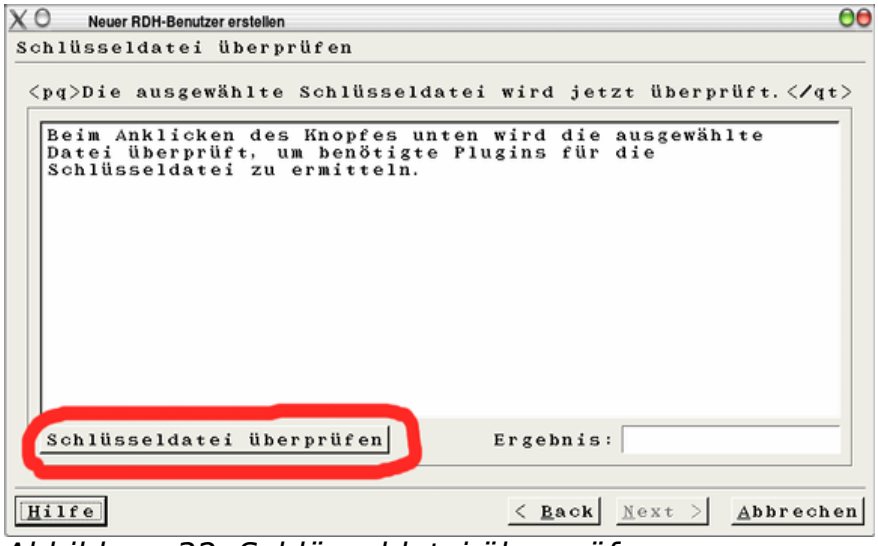

Abbildung 32: Schlüsseldatei überprüfen

Klicken Sie nun also auf *Schlüsseldatei überprüfen*. Sobald rechts als *Ergebnis* eine positive Aussage zu sehen ist, können Sie mit *Next* zum nächsten Schritt wechseln.

Schritt 3: Zugangsdaten eingeben

Die Eingabe der Zugangsdaten verläuft ebenso wie unter *Schlüsseldatei erzeugen* beschrieben.

AqBanking versucht so viele Informationen wie möglich aus der Schlüsseldatei zu lesen. Wundern Sie sich aber nicht, wenn noch keine Informationen ausgefüllt sind, denn nicht jede Version von AqBanking/OpenHBCI schreibt diese Daten (Benutzerkennung, Server-Adresse etc) auch in die Schlüsseldatei.

Schritt 4: Systemkennung abrufen

Siehe hierzu ebenfalls unter *Schlüsseldatei erzeugen*.

Schritt 5: Kontenliste abrufen

Siehe hierzu ebenfalls unter *Schlüsseldatei erzeugen*.

Schritt 6: Abschließen

Da Sie die Schlüsseldatei bereits mit anderen Programmen verwendet haben, ist hier ein Austausch von Schlüsseln und Ini-Briefen nicht erforderlich.

Sie können somit ab sofort mit dem neu eingerichteten Benutzer arbeiten.

6.3.2 Einrichtung auf der Konsole

6.3.2.1 Schlüsseldatei erzeugen

AqBanking verwendet spezielle Module der Bibliothek ***Gwenhywfar***, sogenannte *CryptToken*.

Es existieren bisher CryptToken für OpenHBCI-Schlüsseldateien und DDV-Chipkarten der Sparkassen. Es kommen zukünftig noch weitere hinzu.

Durch Verwendung dieser Module muß AqBanking nicht geändert werden, wenn neue CryptToken unterstützt werden.

Schritt 1: Erzeugen der Datei

Um eine neue Schlüsseldatei zu erzeugen verwenden wir das ***gct-tool***.

```
martin@gwenhywfar:~$ gct-tool create -t ohbci -n /home/martin/test.medium
```

Man gibt hier mit ***-t*** den Namen des CryptToken-Modules an (in diesem Fall soll eine Datei erzeugt werden, wie sie von ***OpenHBCI*** erstmals eingeführt wurde, daher der Name **ohbci**).

Mit ***-n*** wird der Name der zu erzeugenden Datei angegeben.

Sie werden nun zweimal aufgefordert, ein Passwort zu vergeben. Bitte verwenden Sie dazu ein **sicheres Paßwort**. Sie sollten allerdings auf Sonderzeichen (insbesondere Umlaute) verzichten, da diese je nach Kodierung und Zeichensatz Ihres Systems Probleme verursachen.

Mögliche Zeichen sind Alle Buchstaben und Ziffern sowie die Zeichen „$%&/()=?!+-_“.

Die so erzeugte Schlüsseldatei enthält noch keine Schlüssel. Diese werden erst in einem späteren Schritt angelegt.

Schritt 2: Anlegen eines HBCI-Benutzers in AqBanking

Das folgende Kommando legt einen neuen HBCI-Benutzer in AqBanking an:

```
martin@gwenhywfar:~$ aqhbci-tool4 adduser -t ohbci -n /home/martin/test.medium
                     --context=1
                     -b 88888888 -u USER01 -c KUNDE01
                     -s http://www.bankserver.de
                     -N „Martin Preuss“
                     --rdhtype=2
                     --hbciversion=300
```

Hier sind nun einige Parameter nötig.

Die Parameter **-t** und **-n** kennen wir bereits vom ***gct-tool***, sie geben hier das zu verwendende *CryptToken* an.

Wichtig: Bitte geben Sie hier für den Parameter **-n** den absoluten Pfad an, ansonsten funktioniert AqBanking nur, wenn die Anwendung aus des gleichen Verzeichnis aufgerufen wird, in dem dieses Kommando ausgeführt wurde.

Manche *CryptToken* können mehrere Benutzer verwalten, daher muß man zwingend angeben, welchen davon man verwenden möchte. Das *OHBCI*-Modul kennt nur *einen* Benutzer, daher geben wir hier **--context=1** an.

Mit **-b** wird die *Bankleitzahl* angegeben. Normalerweise ist das die Bankleitzahl der Bank, bei der Sie Ihr Konto haben. Manche Server verwenden jedoch eine andere Bankleitzahl. Bitte entnehmen Sie die korrekte Angabe den HBCI-Unterlagen Ihrer Bank.

Der Parameter **-u** gibt Ihre *Benutzerkennung* an. Im Gegensatz zur *Kundenkennung* (welche mit **-c** angegeben werden kann) ist diese Angabe ebenfalls zwingend. Falls in Ihren HBCI-Unterlagen der Bank keine Angabe zur *Kundenkennung* zu finden ist, können Sie das Argument **-c** weglassen. In diesem Fall wird intern dafür ebenfalls die Benutzerkennung verwendet.

Für Überweisungen muß AqBanking den Namen des Kontoinhabers kennen, daher müssen Sie diesen mit **-N** angeben.

Außerdem muß AqBanking noch die *Adresse* des Servers erfahren, die Sie mit **-s** angeben müssen. Falls Sie diesen Parameter weglassen, wird AqBanking versuchen, diese Adresse selbst herauszufinden (AqBanking verfügt dazu über eine Serveradress-Datenbank). Sollte dies nicht gelingen, müssen Sie entweder bei Ihrer Bank nachfragen oder diese Information den HBCI-Unterlagen entnehmen.

Als besonders hilfreich hat sich dazu ansonsten die WWW-Seite „***http://www.hbci-zka.de/institute/institut_auswahl.htm***“ erwiesen.

Der Parameter **–rdhtype=x** ist optional. Er dient dazu, den RDH-Modus auszuwählen. Standardmäßig wird RDH1 verwendet (ältestes RDH-Verfahren, seit HBCI 2.01).

Seit FinTS3 – dem Nachfolger von HBCI – gibt es nun weitere RDH-Modi mit z.T. deutlich größeren Schlüssellängen.

Neuerdings erlauben viele Banken nicht mehr die Einreichung von Schlüsseln für RDH1 (z.B. die GAD-Banken). In diesem Fall muß der RDH-Modus gewählt werden. Welche

RDH-Version Ihre Bank unterstützt erfährt man meistens im Anschreiben der Bank, Modus „2“ dürfte aber von den meisten Banken angeboten werden (u.a. auch von der GAD).

Ein weiterer optionaler Parameter ist **–hbciversion=xxx**. Man kann damit die HBCI-Protokollversion angeben, die für den Zugang verwendet werden soll.

Nicht alle Banken unterstützen alle Versionen, daher kann man diese nun angeben. Mögliche Werte sind **201**, **210**, **220** und **300**.

AqBanking versucht die angegebene Schlüsseldatei testweise zu öffnen, Sie werden daher nach dem Paßwort gefragt.

Schritt 3: Abholen der Serverschlüssel

Sie müssen nun die öffentlichen Schlüssel des Servers abrufen.

```
martin@gwenhywfar:~$ aqhbci-tool4 getkeys -c KUNDE01
```

Sie erhalten hier vermutlich eine Warnung, daß die Signatur des Servers nicht überprüft werden konnte, weil kein Schlüssel vorliegt. Das ist in diesem Fall völlig normal, da wir ja die Schlüssel gerade erst abrufen.

Manche Server signieren ihre eigenen Nachrichten nicht, in diesem Fall erhalten Sie eine Meldung, die besagt, daß der Signier-Schlüssel nicht abgerufen werden konnte. Das können Sie ebenfalls ignorieren.

Sie können sich nun die Schlüssel ansehen. Dazu verwenden wir wieder das **gct-tool**:

```
martin@gwenhywfar:~$ gct-tool showkey -t ohbci -n /home/martin/test.medium
```

Nach Eingabe des Paßwortes bekommen Sie nun einen oder mehrere Schlüssel angezeigt, jeweils mit Modulus und Exponent (den Anteilen eines RSA-Schlüssels).

Schritt 4: Erzeugen der eigenen Schlüssel

Eine neue Schlüsseldatei verfügt noch über keine Schlüssel, diese müssen wir daher nun erzeugen.

```
martin@gwenhywfar:~$ aqhbci-tool4 createkeys -c KUNDE01
```

Beachten Sie bitte, daß Sie hier mit **-c** die *Kundenkennung* angeben müssen. Wie bereits beschrieben entspricht diese der *Benutzerkennung*, wenn Ihre Bank keine Kundenkennungen verwendet.

Bedenken Sie aber auch, daß Sie diesen Schritt nicht mehr ausführen sollten, wenn Sie die Schlüssel aus dieser Datei bereits an die Bank gesendet haben!

Schritt 5: Senden der eigenen Schlüssel an die Bank

Das folgende Kommando sendet die gerade erzeugten Schlüssel an die Bank.

martin@gwenhywfar:~$ **aqhbci-tool4 sendkeys -c KUNDE01**

Bitte beachten Sie, daß Sie hier eine Fehlermeldung erhalten, wenn bei der Bank bereits Schlüssel von Ihnen vorliegen.

Das ist insbesondere dann der Fall, wenn Sie bereits Online-Banking unter der gleichen Benutzerkennung betreiben (z.B. mit einer anderen Anwendung).

Sollte das der Fall sein, antworten manche Banken auch mit einem Verbindungsabbruch.

In einem solchen Fall bleiben Ihnen mehrere Möglichkeiten:

1) Beantragen einer neuen, zusätzlichen Benutzerkennung, die Sie ausschließlich mit AqBanking verwenden
2) Beantragen der Zurücksetzung Ihrer bisher verwendeten Benutzerkennung. Dann können Sie allerdings anschließend nur noch mit AqBanking Online-Banking unter dieser Benutzerkennung betreiben.

Schritt 6: Ausdrucken des INI-Briefes

Nachdem Sie die Schlüssel an die Bank gesendet haben, müssen Sie nun einen sogenannten *INI-Brief* erzeugen und unterschrieben an die Bank senden.

Diesen Brief verwendet die Bank, um die von Ihnen gesendeten Schlüssel zu überprüfen. Somit ist sichergestellt, daß nicht irgendjemand anderes in Ihrem Namen Schlüssel einsendet und Zugriff auf Ihr Konto erhält.

martin@gwenhywfar:~$ **aqhbci-tool4 iniletter -c KUNDE01 >ini.txt**

Nach Eingabe des Paßwortes erzeugt dieses Kommando die Datei **ini.txt**, welche Sie ausdrucken, unterschreiben und an die Bank senden müssen.

Mit der Option **–html** können Sie die Ausgabe auch im HTML-Format erzeugen lassen.

Nach ein paar Tagen wird die Bank Ihren Zugang freischalten, falls der Schlüssel mit den Angaben im INI-Brief übereinstimmt.

Schritt 7: Abrufen der Systemkennung

Der Server der Bank ordnet jedem Kundensystem (d.h. Jeder Anwendung) eine sogenannte *Systemkennung* zu. Diese muß nun abgerufen werden.

```
martin@gwenhywfar:~$ aqhbci-tool4 getsysid -c KUNDE01
```

Dieser Befehl funktioniert allerdings erst, nachdem die Bank Ihren Zugang freigeschaltet hat.

Schritt 8: Abrufen der Kontenliste

Die meisten Banken erlauben den Abruf der Kontenliste. Damit entfällt die manuelle Eingabe der Konten.

```
martin@gwenhywfar:~$ aqhbci-tool4 getaccounts -c KUNDE01
```

Sie können nun überprüfen, ob dabei Kontenlisten empfangen wurden:

```
martin@gwenhywfar:~$ aqhbci-tool4 listaccounts
```

Werden hier keine Konten angezeigt, müssen Sie die Konten von Hand anlegen, z.B. mit dem folgenden Befehl:

```
martin@gwenhywfar:~$ aqhbci-tool4 addaccount
                     -b 88888888 -a 12345 -n „Test-Konto“
                     -N “Heinz Ketchup“
```

Mit **-b** geben Sie die Bankleitzahl des Kontos an, **-a** die Kontonummer und optional mit **-n** den Namen des Kontos. Mit **-N** geben Sie den Namen des Kontoinhabers an. Dies wird beispielsweise für Zahlungsaufträge benötigt.

6.3.2.2 Schlüsseldatei importieren

AqBanking kann bestehende Schlüsseldateien weiterverwenden, wenn sie durch frühere AqBanking- oder OpenHBCI-Anwendungen erzeugt wurden.

Schlüsseldateien anderer Anwendungen können leider nicht importiert werden, da deren Formate von den Herstellern nicht veröffentlicht werden.

Schritt 1: Anlegen eines HBCI-Benutzers in AqBanking

Das folgende Kommando legt einen neuen HBCI-Benutzer in AqBanking an:

```
martin@gwenhywfar:~$ aqhbci-tool4 adduser -t ohbci -n /home/martin/test.medium
                     --context=1
                     -b 88888888 -u USER01 -c KUNDE01
                     -s http://www.bankserver.de
                     -N „Martin Preuss“
```

Dieser Schritt wird ausführlich erklärt im Schritt 1 des vorigen Kapitels.

Schritt 2: Abrufen der Systemkennung

Der Server der Bank ordnet jedem Kundensystem (d.h. Jeder Anwendung) eine sogenannte *Systemkennung* zu. Diese muß nun abgerufen werden.

```
martin@gwenhywfar:~$ aqhbci-tool4 getsysid -c KUNDE01
```

Dieser Befehl funktioniert allerdings erst, nachdem die Bank Ihren Zugang freigeschaltet hat.

Schritt 3: Abrufen der Kontenliste

Die meisten Banken erlauben den Abruf der Kontenliste. Damit entfällt die manuelle Eingabe der Konten. Dieser Schritt wird ebenfalls im vorigen Kapitel näher erläutert.

6.3.2.3 Importieren einer RSA-Chipkarte

AqBanking kann bestimmte ältere RSA-Chipkarten weiterverwenden. Gemeint sind hiermit ältere Karten, die auf dem STARCOS-System basieren. Dies sind beispielsweise die bei Matrica erhältlichen Karten, aber auch manche Kreditinstitute haben bis vor einiger Zeit noch solche Karten herausgeben. Hier kochen allerdings viele Banken ihr eigenes Süppchen, da zum Zeitpunkt der Einführung dieser Karten kein verbindlicher Standard existierte. Hinzukommt, daß Giesecke & Devrient – die Hersteller dieser Karten – die Spezifikationen ihrer Karten nicht herausgegeben haben.

Falls Sie also eine RSA-Karte haben, können Sie probieren, ob AqBanking dieser verwenden kann.

Prinzipiell unterscheidet sich das Importieren einer RSA-Karte nur in einem Punkt vom Importieren einer Schlüsseldatei: In der Erzeugung des Benutzers.

Schritt 1: Anlegen eines HBCI-Benutzers in AqBanking

Das folgende Kommando legt einen neuen HBCI-Benutzer in AqBanking an:

```
martin@gwenhywfar:~$ aqhbci-tool4 adduser -t starcoscard
                     --context=1
                     -b 88888888 -u USER01 -c KUNDE01
                     -s www.bankserver.de
                     -N „Martin Preuss“
```

Dieser Schritt wird ausführlich erklärt im Schritt 1 der vorigen Kapitel. Sie sehen hier aber, dass sich die Einrichtung des Benutzers nur im verwendeten Parameter für **-t** unterscheidet (*starcoscard* statt *ohbci*).

Die weiteren Schritte können Sie dann in den vorigen Kapiteln (insbesondere bei „Importieren einer Schlüsseldatei“) nachlesen.

6.3.2.4 PIN/TAN-Zugang einrichten

Sie können mit AqBanking3 auch Pin/TAN-Zugänge auf der Konsole einrichten.

Schritt 1: Anlegen eines HBCI-Benutzers in AqBanking

Das folgende Kommando legt einen neuen HBCI-Benutzer in AqBanking an:

```
martin@gwenhywfar:~$ aqhbci-tool4 adduser -t pintan
                     --context=1
                     -b 88888888 -u USER01 -c KUNDE01
                     -s https://www.bankserver.de
                     -N „Martin Preuss“
                     --hbciversion=300
```

Dieser Schritt wird ausführlich erklärt im Schritt 1 des vor-vorigen Kapitels. Interessant ist hier, daß als Argument für ***-t*** nicht ***ohbci***, sondern ***pintan*** angegeben werden muß. Außerdem entfällt hier ***-n***. Wie Sie sehen können Sie auch die zu verwendende HBCI-Protokoll-Version angeben.

Schritt 2: Optionales Setzen von Benutzerflags

Normalerweise ist dieser Schritt nicht nötig. In besonderen Fällen müssen aber bestimmte Sondereinstellungen vorgenommen werden, bevor der Server kontaktiert werden kann.

```
martin@gwenhywfar:~$ aqhbci-tool4 adduserflags
                     -b 88888888 -u USER01 -c KUNDE01
                     -f forceSsl3
```

Die zu setzende Sondereinstellung wird hier mit ***-f*** angegeben. Häufigste Einstellung ist hier ***forceSsl3***, da mit manchen Servern nur mit dieser Einstellung Kontakt hergestellt werden kann. Weitere mögliche Flags sind: *bankDoesntSign*, *bankUsesSignSeq*, *ignoreUpd*, und *noBase64*.

Normalerweise sollten Sie diese Flags aber nicht setzen, es sei denn, Sie wissen ganz genau, was Sie tun.

Schritt 3: Abrufen der Systemkennung

Siehe gleichnamigen Schritt des vorigen Kapitels.

Schritt 4: Optionales wählen der iTAN-Methode

Mitunter ist es für AqBanking nicht möglich, automatisch die richtige iTAN-Methode auszuwählen. Sie sehen das meist an Fehlermeldungen bezüglich des verwendeten „Zweischritt-Verfahrens“. In diesem Fall muß die iTAN-Methode manuell gewählt werden.

Lassen Sie sich zunächst die verfügbaren Methoden anzeigen:

```
martin@gwenhywfar:~$ aqhbci-tool4 listitanmodes
                     -b 88888888 -u USER01 -c KUNDE01
TAN Methods
- 997 (2): iTAN (Indiziertes TAN-Verfahren) [available]
- 996 (2): mobileTAN (mobileTAN) [not available]
```

Sie sehen in diesem Beispiel, daß die Bank die Methoden „iTAN“ und „mobileTAN“ kennt. Allerdings ist davon nur die Methode „iTAN“ (997) verfügbar. Wählen Sie diese nun aus:

```
martin@gwenhywfar:~$ aqhbci-tool4 setitanmode
                     -b 88888888 -u USER01 -c KUNDE01
                     -m 997
```

Schritt 5: Abrufen der Kontenliste

Siehe gleichnamigen Schritt des vorigen Kapitels.

Schritt 6: Pin ändern

Falls es sich um die erstmalige Einrichtung eines PIN/TAN-Zuganges handelt, müssen Sie eventuell die PIN ändern, bevor Sie Bankgeschäfte über HBCI erledigen können. Verwenden Sie dazu das Kommando *changepin*.

```
martin@gwenhywfar:~$ aqhbci-tool4 changepin
                     -b 88888888 -u USER01 -c KUNDE01
```

7 EBICS-Einrichtung von AqBanking

7.1 Einrichtung auf der Konsole

7.1.1 Schlüsseldatei erzeugen

Schritt 1: Erzeugen der Schlüsseldatei

Zunächst muß eine leere Schlüsseldatei angelegt werden.

```
martin@gwenhywfar:~$ gct-tool create -t ohbci -n test.medium
```

Mit diesem Befehl wird die Schlüsseldatei ***test.medium*** erzeugt. Diese Datei enthält noch keine Schlüssel, diese werden in einem späteren Schritt erzeugt.

Schritt 2: Anlegen eines EBICS-Benutzers

Hierfür wird nun das neue Tool ***aqebics-tool*** eingesetzt. Es ist Bestandteil des erweiterten ***aqbanking-cli***-Paketes mit EBICS-Unterstützung.

```
martin@gwenhywfar:~$ aqebics-tool adduser
                     -t ohbci
                     -n /home/martin/test.medium
                     --context=1
                     -b 20050550
                     -H HOSTNAME
                     -N „Martin Preuss“
                     -u TEILNEHMERKENNUNG
                     -c KUNDENKENNUNG
                     -s „https://ebics.testbank.de“
```

Mit ***-t*** wird hier das CryptToken-Modul gewählt. Einziges bisher zulässiges Modul ist ***ohbci*** (also Schlüsseldatei vom OpenHBCI-Typ, wie sie auch für HBCI verwendet wird).

Der Pfad zur zu verwendenden Schlüsseldatei wird mit ***-n*** angegeben. Hier sollte stets ein absoluter Pfad verwendet werden, damit diese Datei immer gefunden wird.

Ein CryptToken kann unter Umständen mehrere Benutzer speichern. Schlüsseldateien speichern jedoch immer nur einen Benutzer, daher ist hier *--context=1* anzugeben.

Die Bankleitzahl wird mit ***-b*** angegeben.

Für EBICS ist außerdem die Einstellung für den Namen des EBICS-Servers mit ***-H***

anzugeben. Diese Einstellung entnehmen Sie den Unterlagen Ihrer Bank.

Des weiteren benötigen Sie noch eine *Teilnehmerkennung* sowie eine *Kundenkennung*. Beides finden Sie ebenfalls in den Unterlagen von Ihrer Bank. Geben Sie die Teilnehmerkennung mit ***-u*** an und die Benutzerkennung mit ***-c***.

Als nächstes muß noch die URL des Servers angegeben werden (mit ***-s***).

Schritt 3: Optionales Setzen von Benutzer-Flags

Für manche Bankserver müssen noch bestimmte Benutzerflags gesetzt werden. Dies geschieht generell mit dem folgenden Kommando:

```
martin@gwenhywfar:~$ aqebics-tool adduserflags
                     -c BENUTZERKENNUNG
                     -f FLAG
```

Der Parameter ***-f FLAG*** kann mehrfach angegeben werden, jeweils einmal pro zu setzendem Flag.

Die folgenden Flags können gesetzt werden (Groß- und Kleinschreibung ist dabei nicht relevant):

Flag	Beschreibung
ForceSSLv3	Erzwingt die Verwendendung einer bestimmten SSL-Version für die Verschlüsselung der EBICS-Nachrichten. Sollten beim Versenden von Aufträgen Meldungen über TLS- oder SSL-Fehler auf der Konsole erscheinen, so muss dieses Flag gesetzt werden
UseIZL	Lastschriften können über verschiedene Auftragsarten versendet werden: Entweder als normaler Zahlungsauftrag (*IZV=“Inlandszahlungverkehr“*) oder als eigenständiger Auftrag (*IZL=„Inlandszahlungsverkehr:Lastschrift“*). Wird dieses Flag gesetzt, so wird die Auftragsart IZL für Lastschriften verwendet.
TimestampFix1	Manche Server verstehen nur bestimmte Zeitstempel-Formate. Falls Ihr Server immer alle Nachrichten ablehnt und womöglich noch ein falsches Datum als Grund angibt (oder aber Fehlermeldungen über Doppeleinreichungen o.ä. Zeigt) sollten Sie dieses Flag setzen. Damit wird das Zeitstempelformat „YYYY-MM-DDThh:mm:ss.000Z“ verwendet (also in GMT).
noEu	Für den seltenen Fall, dass Sie Aufträge für den Benutzer immer ohne sogenannte Elektronische Unterschrift versenden wollen, müssen Sie dieses Flag setzen. Normalerweise benötigen Sie dieses Flag nicht. Falls Sie es verwenden, müssen Sie mit geeigneter anderer Software die elektronischen Unterschriften nachreichen (oder aber den Auftrag nachträglich schriftlich bei der Bank authorisieren).

Schritt 4: Schlüssel erzeugen

Jetzt müssen die zu verwendenden Schlüssel in der Schlüsseldatei erzeugt werden. Es werden 3 Schlüssel erzeugt:

- ein Authentifikations-Schlüssel (wird für die Übertragungssicherung verwendet)
- ein Signier-Schlüssel (wird für elektronische Unterschriften von Aufträgen verwendet)
- ein Verschlüsselungs-Schlüssel (wird für die Dechiffrierung der Banknachrichten verwendet)

Diese Schlüssel werden mit dem folgenden Kommando erzeugt:

```
martin@gwenhywfar:~$ aqebics-tool createkeys -c BENUTZERKENNUNG
```

Schritt 5: Schlüssel an die Bank senden

Die soeben erzeugten Schlüssel müssen nun an die Bank gesendet werden.

```
martin@gwenhywfar:~$ aqebics-tool sendkeys -c BENUTZERKENNUNG
```

Schritt 6: INI-Briefe erzeugen und an die Bank schicken/faxen

Damit die Bank Ihre Schlüssel überprüfen und somit sicherstellen kann, daß diese Schlüssel von Ihnen stammen, müssen sogenannte INI-Briefe erzeugt werden. Diese enthalten einige kryptographische Informationen die der Bank die Prüfung ermöglichen.

Es handelt sich dabei um 2 Briefe, die jeweils andere Schlüssel beschreiben:

- einer für den Authentifikations- und Verschlüsselungsschlüssel (andere Anwendungen verwenden hierfür oft den gleichen Schlüssel)
- einer für den Signatur-Schlüssel

```
martin@gwenhywfar:~$ aqebics-tool iniletter -c BENUTZERKENNUNG
```

```
martin@gwenhywfar:~$ aqebics-tool hialetter -c BENUTZERKENNUNG
```

Beide Befehle senden ihre Ausgabe jeweils auf die Konsole, daher sollten Sie diese Ausgabe z.B. in eine Datei umleiten und diese dann ausdrucken und unterschreiben.

Anschließend müssen Sie warten, bis die Bank Ihre Schlüssel freischaltet. Erst dann können Sie mit dem nächsten Schritt weitermachen.

Schritt 7: Abrufen der Server-Schlüssel

Der Server verwendet ebenfalls Schlüssel, die Sie in diesem Schritt abrufen müssen. Erst dann kann die Verbindung zwischen Ihrem Rechner und der Bank abgesichert werden.

```
martin@gwenhywfar:~$ aqebics-tool getkeys -c BENUTZERKENNUNG
```

Schritt 8: Prüfen der Server-Schlüssel

Um sicherzugehen, daß empfangenen Schlüssel wirklich vom Server stammen, müssen nun die Hashwerte überprüft werden. Diese finden Sie für gewöhnlich in den Unterlagen von Ihrer Bank.

```
martin@gwenhywfar:~$ aqebics-tool hialetter -c BENUTZERKENNUNG --bankkey
```

Sie sollten nur dann mit diesem Bankzugang arbeiten, wenn die Hashwerte der Server-Schlüssel uebereinstimmen. Andernfalls sollten Sie unbedingt Kontakt zu Ihrer Bank aufnehmen!

Schritt 9: Abrufen der Kontenliste

In diesem Schritt wird versucht, die Kontenliste abzurufen. Dies sollte mit den meisten Servern gelingen. Falls nicht, müssen Sie die Konten manuell hinzufügen.

```
martin@gwenhywfar:~$ aqebics-tool getaccounts -c BENUTZERKENNUNG
```

Schritt 10: Konto manuell hinzufügen

Falls Sie im vorigen Schritt die Meldung erhalten haben, daß der Server den Abruf von Benutzerinformationen nicht erlaubt, müssen Sie Ihre Konten manuell hinzufügen.

Verwenden Sie dazu das Kommando ***addaccount*** wie im folgenden Beispiel:

```
martin@gwenhywfar:~$ aqebics-tool addaccount
                     -b 88888888
                     -a 1234567890
                     -n „KONTO_NAME“
                     -u USER01 -c KUNDE01
                     --owner=„Martin Preuss“
```

Mit ***-b*** geben Sie die Bankleitzahl an, mit ***-a*** die Kontonummer und mit ***--owner*** den Namen des Kontoinhabers.

Jedes Konto muß einem Benutzer zugeordnet werden, daher müssen Sie diesen mit ***-u*** oder mit ***-c*** (oder beiden) angeben.

Der Übersichtlichkeit wegen sollten Sie dem Konto auch einen Namen geben (mit ***-n***).

7.1.2 Schlüsseldatei importieren

Schritt 1: Anlegen eines EBICS-Benutzers

Hierfür wird nun das neue Tool ***aqebics-tool*** eingesetzt. Es ist Bestandteil des erweiterten ***aqbanking-cli***-Paketes mit EBICS-Unterstützung.

```
martin@gwenhywfar:~$ aqebics-tool adduser
                     -t ohbci
                     -n /home/martin/test.medium
                     --context=1
                     -b 20050550
                     -H HOSTNAME
                     -N „Martin Preuss“
                     -u TEILNEHMERKENNUNG
                     -c KUNDENKENNUNG
                     -s „https://ebics.testbank.de“
                     --import
```

Die Argumente hier entsprechen denen im vorigen Kapitel. Neu ist hier lediglich der Parameter –-**import**. Hiermit das EBICS-Backend angewiesen, den neuen Benutzer als *aktiv* zu betrachten. Sie sollten also direkt mit diesem Zugang arbeiten können.

Anschließend kann die Einrichtung wie im vorigen Kapitel beschrieben ab **Schritt 9** fortgesetzt werden.

8 Verwenden von PIN-Dateien

Für die automatisierte Verwendung von AqBanking-CLI oder des Tools *aqebics-tool* können Sie die Passwörter oder PINs in einer Datei speichern und diese Datei beim Aufruf angeben.

Grundsätzlich sollten Sie bedenken, daß Sie damit die Sicherheit des Bank-Zugriffes deutlich einschränken. Jeder, der in den Besitz dieser Datei gelangt, hat Zugrif auf die damit abgesicherten Konten!

Unter Umständen ist diese Speicherung der Pin bzw. des Passwortes für die Schlüsseldatei auch durch Ihren Vertrag mit der Bank zum Online-Banking ***untersagt***. Dies sollten Sie unbedingt klären, bevor Sie dieses Feature verwenden.

8.1 Pindatei für HBCI erzeugen

Das folgende Kommando erzeugt Einträge für die HBCI-Konten und schreibt das Ergebnis in die Datei mit dem Namen ***PINDATEI***.

```
martin@gwenhywfar:~$ aqhbci-tool4 mkpinlist >PINDATEI
```

Sie müssen anschließend diese Datei mit einem Editor öffnen und die nicht benötigten Einträge entfernen. Für die benötigten Einträge können Sie dann die PIN bzw. das Passwort einfügen.

8.2 Pindatei für EBICS erzeugen

Analog können Sie sich mit dem folgenden Kommando Einträge für die EBICS-Konten ausgeben lassen:

```
martin@gwenhywfar:~$ aqebics-tool mkpinlist >PINDATEI
```

8.3 Absicherung des Zugriffes auf die Pindatei

Um die Pindatei abzusichern empfiehlt sich folgendes Vorgehen.

8.3.1 Benutzer *aqbanking* anlegen

Verwenden Sie dazu das Kommando *adduser* (auf Debian-Systemen) oder falls es nicht vorhanden ist *useradd*.

```
martin@gwenhywfar:~$ addgroup aqbanking
                     adduser aqbanking
                     adduser aqbanking aqbanking
```

8.3.2 Tools dem Benutzer *aqbanking* zuordnen

Die Tools *aqbanking-cli*, *aqhbci-tool4* und *aqebics-tool* sollten nun diesem Benutzer zugeordnet werden:

```
martin@gwenhywfar:~$ chown aqbanking:aqbanking /usr/bin/aqbanking-cli
                     chown aqbanking:aqbanking /usr/bin/aqhbci-tool4
                     chown aqbanking:aqbanking /usr/bin/aqebics-tool
```

8.3.3 Berechtigungen der Anwendungen anpassen

Der Trick besteht nun darin, dass man das SETUID-Bit der Berechtigungen setzt. Damit wird eine Anwendung, die von einem berechtigten Benutzer aufgerufen wird, unter der Benutzerkennung des Besitzers dieser Datei ausgeführt.

Außerdem können wir bei dieser Gelegenheit auch gleich die Liste der Benutzer einschränken, die diese Anwendungen ausführen dürfen.

```
martin@gwenhywfar:~$ chmod u=rxs,g=rx,o=  /usr/bin/aqbanking-cli
                     chmod u=rxs,g=rx,o=  /usr/bin/aqhbci-tool4
                     chmod u=rxs,g=rx,o=  /usr/bin/aqebics-tool
```

Bitte beachten Sie das Leerzeichen hinter dem *o*=.

8.3.4 Berechtigungen der PIN-Datei anpassen

Mit den folgenden Kommandos legen Sie fest, daß nur der Benutzer aqbanking die Pin-Datei lesen darf:

```
martin@gwenhywfar:~$ chown aqbanking:aqbanking PINDATEI
                     chmod u=rw,g=,o= PINDATEI
```

Somit dürfen also nicht einmal Sie selbst unter Ihrer eigenen Benutzerkennung auf diese Pindatei zugreifen, und genau das ist die Idee hinter diesen Änderungen.

8.3.5 Warum das ganze?

Das Ergebnis ist nun, dass die angegebenen Anwendungen nur von einem Benutzer aufgerufen werden können, der sich in der Gruppe *aqbanking* befindet. Ausserdem werden diese Anwendungen unter der Benutzerkennung *aqbanking* ausgeführt.

Weil also die Anwendungen unter der Benutzerkennung *aqbanking* laufen, können diese – und **nur** diese - auch die Pin-Datei lesen.

Da aber die Berechtigungen für die Pin-Datei so strikt gewählt wurden, daß nicht einmal Sie selbst unter Ihrer eigenen Benutzerkennung diese Datei lesen können, kann das auch kein anderer Prozess, der unter Ihrer Benutzerkennung läuft.

Somit ist die Pindatei weitestgehend vor unbefugtem Auslesen geschützt.

Beachten Sie aber, daß der Super-User (root) diese Datei dennoch in jedem Fall lesen kann.

9 Verwenden von Proxies

9.1 AqBanking4

Seit Version 3.7.0 enthält Gwenhywfar Dank Andreas Steinmetz Unterstützung für HTTP-Proxies. Um einen solchen zu verwenden, muß die Umgebungsvariable „GWEN_PROXY“ die URL des Proxy-Servers enthalten, also beispielsweise:

```
martin@gwenhywfar:~$ export GWEN_PROXY=mein.proxy.de:8080“
martin@gwenhywfar:~$ QBankManager
```

Damit wird Gwenhywfar angewiesen, den Proxy-Server auf mein.proxy.de (Port 8080) zu verwenden.

Ab Version 3.10.1 kann Gwenhywfar auch Benutzername und Passwort für einen HTTP-Proxy verwenden. Für den Benutzer martin mit dem Passwort secret würde das Kommando also wie folgt ausehen:

```
martin@gwenhywfar:~$ export
                     GWEN_PROXY=martin:secret@mein.proxy.de:8080“
martin@gwenhywfar:~$ QBankManager
```

9.2 AqBanking3

AqBanking3 hat derzeit noch keine direkte Unterstützung für Proxies. Dank ***Micha Lenk*** kann man sich unter Linux aber mit dem folgenden Trick behelfen.

Zunächst muß ein Pearl-Skript heruntergeladen werden (http://www.stud.uni-karlsruhe.de/~uinx/devel/proxytunnel.pl).

Schritt 1: Starten des Skripts

```
martin@gwenhywfar:~$ proxytunnel.pl -l 3000
                     -p PROXYSERVER:8080
                     -d BANKSERVER:3000
```

Für **PROXYSERVER** muß die URL des Proxy-Servers angegeben werden, für

BANKSERVER die Adresse des *Bankservers*.

Durch diesen Aufruf wartet das Skript auf eingehende Verbindungen auf Port *3000* (also dem Port, der für *HBCI* verwendet wird).

Schritt 2: Umleiten ausgehender HBCI-Nachrichten auf das Skript

```
martin@gwenhywfar:~$ iptables -t nat -A OUTPUT -p tcp
                     --dst hbci.bank.de
                     --dport 3000
                     -j DNAT
                     --to-destination 127.0.0.1:3000
```

Für diesen Schritt werden **root**-Rechte benötigt. Damit werden ausgehende Anfragen an den *Bankserver* umgeleitet auf das auf Ihrem Rechner laufende Skript, welches seinerseits über den angegebenen Proxy an den Bankserver weiterleitet.

10 AqBanking-CLI

AqBanking-CLI ist ein Kommandozeilen-basiertes Tool für einige Aufgaben

des Online-Bankings. Es enthält AqBanking4 und Gwenhywfar3.

10.1 Neues in AqBanking-CLI-2

AqBanking-CLI-2 verwendet das neue **AqBanking4** und profitiert daher von dessen neuen Fähigkeiten. Hier ist insbesondere die Unterstützung für den parallelen Betrieb mehrerer **AqBanking4**-Anwendungen (zur gleichen Zeit) zu nennen.

Ermöglicht wird dies durch kurzfristiges partielles automatisches Sperren (locking) und neuladen der Konfigurationsdaten vor und während eines Online-Banking-Zugriffes. Damit ist sichergestellt, daß sich mehrere parallel arbeitende Programme nicht unkontrolliert beeinflussen und dennoch alle Anwendungen jeweils auf die aktuellste Konfiguration zugreifen.

AqBanking-CLI 2 enthält außerdem die neue Bibliothek **AqFinance** mit dessen Hilfe die folgenden Kommandos hinzugekommen sind:

- *dbinit*: Initialisiert die von AqFinance verwaltete Datenbank
- *dbrecon*: Gleicht offene Überweisungen mit vorhandenen Umsatzdaten ab
- *dblisttrans*: Exportiert eine query-gesteuerte Auswahl von Umsatzdaten via AqBanking
- *dblisttransfers*: Exportiert eine query-gesteuerte Auswahl von Überweisungen via AqBanking

Erweitert wurden außerdem die folgenden Kommandos:

- *import*: Dieses Kommando kann nun auch Umsatzdaten und Überweisungen in die AqFinance-Datenbank importieren
- *request*: Importiert wenn gewünscht heruntergeladene Daten von der Bank direkt in die AqFinance-Datenbank
- *transfer*, *transfers*: Diese Kommandos können nun eingereichte Überweisungen direkt in die AqFinance-Datenbank übernehmen und den Status des Auftrages darin vermerken (damit stehen diese Aufträge für den automatischen Abgleich zur Verfügung)

Neu ist auch ein Kommando zum Import von Konfigurationen früherer AqBanking-Versionen:

- *updateconf*: Dieses Kommando sucht nach der neuesten Konfiguration (in der Reihenfolge AqBanking4, 3, 2). Ist diese neueste Konfiguration keine von AqBanking4 sondern älter, wird sie importiert, so daß man anschließend mit dieser Konfiguration arbeiten kann.

10.1.1 AqFinance

Diese interne Bibliothek dient der Speicherung von Umsatzdaten und Überweisungen. Sie enthält außerdem Funktionen zum automatischen Abgleich von Überweisungen mit Hilfe von Umsatzdaten.

Dadurch kommt AqBanking-CLI2 an die Funktionalität des alten AqMoney heran und bietet zusätzliche Fähigkeiten.

Features von AqFinance:

- Import von Umsatzdaten und Überweisungen via AqBanking
- Export von Umsatzdaten und Überweisungen via AqBanking
- automatischer Abgleich von Überweisungen mit importierten Umsatzdaten
- Auswahl von Umsatzdaten und Überweisungen durch Query-Strings, z.B. **($date>="20080101") && ($date<="20081231")**
- Intern können Anfragen auch verschachtelt werden, d.h. eine folgende Anfrage (Query) kann sich auf das Ergebnis einer vorangegangenen Anfrage beziehen

10.2 Allgemeines zu Auswahlausdrücken

Insbesondere im Umgang mit der internen Datenbank von AqBanking-CLI-2 wird in den nächsten Kapiteln öfter die Rede von sogenannten *Auswahlausdrücken* sein.

Dabei handelt es sich um Ausdrücke, anhand derer beispielsweise Umsatzdaten aus der Gesamtmenge ausgewählt werden können. Ein solcher Ausdruck besteht grundsätzlich aus 3 Anteilen:

- Operand 1
- Vergleichsoperator („==“, „!=“, „>“, „<“, „>=“, „<=“)
- Operand 2

Will man also den einen Umsatz auswählen, der die ID „1“ trägt, würde der Ausdruck folgendermaßen lauten:

$id==“1“

Es können auch mehrere solcher Ausdrücke verknüpft werden, wie in:

($date>=“20080101“) && ($date<=“20081231“)

Mögliche Verknüpfungen sind:

- „&&“: beide Teilausdrücke müssen erfüllt sein
- „||“: mindestens einer der Teilausdrücke muß erfüllt sein

Teilausdrücke werden von Klammern umschlossen.

Das Dollarzeichen („$“) leitet einen Feldnamen ein. Für Umsätze sind beispielsweise die folgenden Feldnamen erlaubt:

Feldname	Typ	Beschreibung
Id	ID	Eindeutige ID des Umsatzes
LocalCountry	String	Staat in dem die eigene Bank Ihren Sitz hat
LocalBankCode	String	Bankleitzahl der eigenen Bank
LocalAccountNumber	String	Kontonummer des eigenen Kontos
LocalIban	String	IBAN des eigenen Kontos
LocalBic	String	BIC Code der eigenen Bank
LocalName	String	Name des Kontoinhabers des eigenen Kontos
RemoteCountry	String	Staat in dem die andere Bank Ihren Sitz hat
RemoteBankCode	String	Bankleitzahl der anderen Bank
RemoteAccountNumber	String	Kontonummer des anderen Kontos
RemoteIban	String	IBAN des anderen Kontos
RemoteBic	String	BIC Code der anderen Bank
RemoteName	String	Name des Kontoinhabers des anderen Kontos
Date	Datum	Buchungsdatum (Format immer: YYYYMMDD)
ValutaDate	Datum	Datum der Wertstellung bei Umsätzen, Datum des automatischen Abgleiches bei Überweisungen (siehe Kommando *dbrecon*)
Value	Betrag	Betrag (mit Vorzeichen, wir im rationalen Format gespeichert, also in der Form ZÄHLER/NENNER)
Currency	String	ISO-Währungscode (z.B. „EUR“, „USD“)
FiId	String	Eindeutige Kennung zugeordnet durch die Bank (wird bei HBCI nicht verwendet, wohl aber bei OFX DirectConnect)
TextKey	Int	Textschlüssel (z.B. „51“ für normale Überweisungen, „04“ bzw. „05“ für Lastschriften/Einzugsverfahren etc).Wird nicht von allen Banken übertragen.
TextKeyExt	Int	Textschlüsselergänzung (optionale zusätzliche Angaben zum Textschlüssel)
InfoText	String	Optionale Umsatzbeschreibung, z.B. „GEHALT“, „LASTSCHRIFT“, „DAUERAUFTRAG“. Wird nicht von allen Banken übertragen.
CustomerRef	String	Kundenreferenz; eine Zeichenkette, die bei der Überweisung angegeben wurde. Wird nicht von allen Banken übertragen.
BankRef	String	Bankreferenz, wird nicht von allen Banken übertragen.
Purpose	String	Einzeiliger Verwendungszweck. Wenn im Original mehrere Verwendungszweckzeilen vorhanden waren, werden diese durch Leerzeichen getrennt aneinandergereiht.
Status	Int	Statuscode für Aufträge; bisher erlaubte Werte: **0**: keiner **1**: Auftrag wird gerade gesendet

		2: Auftrag von der Bank angenommen **3**: Auftrag von der Bank abgelehnt **4**: Auftrag angenommen, Ausführung aber noch nicht gewiß **5**: Auftrag wurde durch eingehende Umsatzdaten bestätigt (automatisch) **6**: Auftrag wurde durch eingehende Umsatzdaten bestätigt (manuell durch den Benutzer)

Tabelle 1: Erlaubte Feldnamen für Umsätze und Überweisungen

10.3 Anwendungsbeispiel

Zunächst sollte die interne Datenbank initialisiert werden:

```
martin@gwenhywfar:~$ aqbanking-cli dbinit
```

Anschließend können Überweisungen/Lastschriften eingereicht werden, z.B.:

```
martin@gwenhywfar:~$ aqbanking-cli transfer
                     -b 28250110
                     -a 1234567890
                     --rbank=28250110
                     --raccount=1234567890
                     --rname=“Heinz Ketchup“
                     -v „12,34:EUR“
                     -p „VERWENDUNGSZWECK1“
                     --usedb
```

Die Option „--usedb“ sorgt dafür, daß sowohl die Überweisungen selbst als auch deren Ergebnis in der internen Datenbank gespeichert werden.

Man kann sie sich schon mal anschauen:

```
martin@gwenhywfar:~$ aqbanking-cli dblisttransfers
                     --exporter=csv
                     --profile=full
```

Dadurch werden alle Überweisungen angezeigt. Wir können die Ausgabe aber auch einschränken:

```
martin@gwenhywfar:~$ aqbanking-cli dblisttransfers
                     --exporter=csv
                     --profile=full
                     -q „\$status==\“4\““
```

Hier wurde ein Query-String angegeben; nur die Umsätze mit dem Status „4“ (=Auftrag von der Bank angenommen, aber Ausführung noch ungewiss) werden exportiert.

Am Folgetag sollte diese Überweisung in den Umsätzen auftauchen, daher rufen wir diese also nun ab:

```
martin@gwenhywfar:~$ aqbanking-cli request
                     -b 28250110
                     -a 1234567890
                     --transactions
                     --usedb
```

Auch hier sorgt der Schalter „--usedb“ wieder dafür, daß die empfangenen Daten in die interne Datenbank geschrieben werden.

Falls unsere Überweisung vom Vortag nun von der Bank ausgeführt wurde, können wir AqBanking-CLI2 den Status der Überweisung anpassen lassen (Abgleich der Überweisung). Dies geschieht mit:

```
martin@gwenhywfar:~$ aqbanking-cli dbrecon
```

Sie können sich anschließend die Überweisungen erneut anzeigen lassen (siehe oben). Um Überweisungen zu sehen, die heute (z.B. 25.11.2008) abgeglichen wurden, verwenden wir einen etwas anderen QueryString:

```
martin@gwenhywfar:~$ aqbanking-cli dblisttransfers
                     --exporter=csv
                     --profile=full
                     -q „($status==“5“) && ($valutadate==“20081125“)“
```

Zur Vereinfachung wurden die „\“-Zeichen in diesem Beispiel weggelassen, aber beachten Sie bitte, daß Sie dieses Zeichen den speziellen Zeichen voranstellen müssen, die von der Kommandozeile normalerweise interpretiert werden (wie beispielsweise das Dollarzeichen, die Klammern und das Anführungszeichen).

Der Status „5“ (=Auftrag wurde durch eingehende Umsätze automatisch bestätigt) wird vom Kommando „dbrecon“ für alle automatisch abgeglichenen Überweisungen gesetzt. Gleichzeitig wird das Feld „$valutadate“ der entsprechenden Überweisung auf das aktuelle Datum gesetzt.

Durch intelligente Wahl der QueryStrings kann man nun beispielsweise die heute abgeglichenen Überweisungen anzeigen lassen, oder aber solche anzeigen lassen, die schon zu lange nicht abgeglichen sind (für Mahnungen oder Überpüfungen etc).

10.4 Kommandos

10.4.1 Optionen

AqBanking-CLI unterscheidet zwischen globalen Optionen und Optionen, die zu einem bestimmten Kommando gehören.

Der generelle Aufruf von AqBanking-CLI ist:

```
aqbanking-cli GLOBALE_OPTIONEN KOMMANDO KOMMANDO_OPTIONEN
```

Globale Optionen erscheinen also ***vor*** dem Kommando, Optionen zu Kommandos erscheinen **nach** dem Kommando.

Option	Beschreibung
-D VERZEICHNIS	Mit dieser Option können Sie das AqBanking-Konfiguationsverzeichnis vorgeben. Sie sollten dies allerdings nur tun, wenn Sie dazu einen guten Grund haben. Ansonsten wird das Standardverzeichnis ($HOME/.aqbanking) verwendet
-n	Schaltet den nicht-interaktiven Modus ein. In diesem Modus versucht aqbanking-cli ohne Rückfragen an den Benutzer auszukommen. Sollten allerdings wichtige Rückfragen auftreten, bricht das Programm mit einer Fehlermeldung ab. Im nicht-interaktiven Modus muss eine Pin-Datei verwendet werden (siehe -P)
-P DATEI	Diese Option gibt die zu verwendende Pin-/Passwortdatei an. Bei Verwendung dieser Datei entfällt die Nachfrage nach Pins und Passwörtern, wenn diese in der Datei aufgeführt sind.

Table 2: Globale Optionen

Eine Pindatei für HBCI erzeugen Sie durch das folgende Kommando:

```
martin@gwenhywfar:~$ aqhbci-tool4.sh mkpinlist
```

Für EBICS gilt analog:

```
martin@gwenhywfar:~$  aqebics-tool.sh mkpinlist
```

In die so erzeugte Datei müssen Sie die entsprechenden Pins und Passwörter einfügen. Sie können übrigends die Pindatei von HBCI und EBICS zu einer gemeinsamen Datei zusammenfügen.

10.4.2 updateconf

Dieses Kommando versucht bei fehlender Konfiguration für AqBanking4 eine ältere Konfiguration zu importieren. Unterstützt wird derzeit der Import von Konfigurationen für AqBanking2 und AqBanking3.

Dieses Kommando muß beim Wechsel von AqBanking-CLI-1 zu AqBanking-CLI-2 einmalig aufgerufen werden.

10.4.3 dbinit

Dieses Kommando von AqBanking-CLI2 initialisiert einmalig die interne Datenbank. Erst nach Aufruf dieses Kommandos kann die Datenbank für weitere Kommandos verwendet werden.

10.4.4 request

Dieses Kommando ruft Kontoinformationen ab, wie Kontostände, Umsätze,

Liste der Daueraufträge und Liste der offenen terminierten Überweisungen.

10.4.4.1 Optionen

[-b PARAM] [--bank=PARAM]	Bankleitzahl
[-a PARAM] [--account=PARAM]	Kontonummer
[-N PARAM] [--bankname=PARAM]	Name der Bank
[-n PARAM] [--accountname=PARAM]	Name des Kontos
[-c PARAM] [--ctxfile=PARAM]	Name der zu verwendenden CTX-Datei. In dieser Datei werden die abgerufenen Informationen speichert. Andere Kommandos verwenden diese Datei als Eingabe. Diese Option kann nicht zusammen mit [--usedb] verwendet werden.
[--transactions]	Mit dieser Option werden Umsätze abgerufen und in der

	CTX-Datei gespeichert.
[--balance]	Mit dieser Option werden Kontostände abgerufen und in der CTX-Datei gespeichert.
[--sto]	Mit dieser Option werden Dauerauftraege abgerufen und in der CTX-Datei gespeichert.
[--dated]	Mit dieser Option werden terminierte Überweisungen abgerufen und in der CTX-Datei gespeichert
[--fromdate=PARAM]	Mit dieser Option kann man das Startdatum für den Abruf von Umsätzen festlegen
[--todate=PARAM]	Mit dieser Option kann man das Enddatum für den Abruf von Umsätzen festlegen
[--usedb]	Ist diese Option vorhanden, werden die empfangenen Umsätze in der internen Datenbank gespeichert. Diese Option kann nicht zusammen mit [-c] verwendet werden.

10.4.4.2 Beispiele

- Abruf von Umsätzen für das Konto *1234567890* bei der *Sparkasse WHV* und speichern der Informationen in der Datei *test.ctx*:

```
martin@gwenhywfar:~$ aqbanking-cli request
                     -b 28250110
                     -a 1234567890
                     -c test.ctx
                     --transactions
```

- Abruf von Kontoständen und Umsätzen von allen Konten bei der Sparkasse WHV:

```
martin@gwenhywfar:~$ aqbanking-cli request
                     -b 28250110
                     -c test.ctx
                     --transactions
                     --balance
```

Man sieht an diesen Beispielen, dass man auch die Informationen mehrerer Konten in einem Rutsch abrufen kann.

Man muss im übrigen nicht alle Optionen angeben, um ein Konto auszuwählen. Die Optionen, die tatsächlich angegeben sind, werden verwendet, um die Konten auszuwählen, für die Informationen abgerufen werden sollen.

10.4.5 listaccs

Dieses Kommando zeigt alle Konten an, die AqBanking verwaltet. Die Liste kann durch Angabe von Parametern eingeschränkt werden.

Die Liste besteht aus einem Konto pro Zeile, die Felder einer Zeile werden durch ein TAB (Zeichen ASCII 9) getrennt.

Felder:

1	„Account“ (fester String)
2	Bankleitzahl des Kontos
3	Kontonummer
4	Name der Bank (falls bekannt)
5	Name des Kontos (falls bekannt)

10.4.5.1 Optionen

[-b PARAM] [--bank=PARAM]	Bankleitzahl
[-a PARAM] [--account=PARAM]	Kontonummer
[-N PARAM] [--bankname=PARAM]	Name der Bank
[-n PARAM] [--accountname=PARAM]	Name des Kontos
[-c PARAM] [--ctxfile=PARAM]	Name der zu verwendenden CTX-Datei. In dieser Datei werden die abgerufenen Informationen speichert. Andere Kommandos verwenden diese Datei als Eingabe.

10.4.5.2 Beispiele

- Anzeigen aller Konten:

```
martin@gwenhywfar:~$ aqbanking-cli
```

- Anzeigen aller Konten bei der Sparkasse WHV:

```
martin@gwenhywfar:~$ aqbanking-cli -b 28250110
```

10.4.6 senddtazv

Mit diesem Konto kann eine extern vorbereitete DTAZV-Datei (für Auslandszahlungen) an die Bank gesendet werden.

Dieses Kommando verwaltet Tages-Zählerdateien pro Konto, für das mit diesem Befehl DTAZV-Dateien versendet wurden.

Der Name der Tageszählerdateiei wird wie folgt zusammengesetzt:

„*BANKLEITZAHL-KONTONUMMER-DATUM*.cnt“

Das Datum wird in folgendem Format verwendet: JJJJMMDD (Jahr, Monat, Tag).

10.4.6.1 Optionen

[-b PARAM] [--bank=PARAM]	Bankleitzahl
[-a PARAM] [--account=PARAM]	Kontonummer
[-N PARAM] [--bankname=PARAM]	Name der Bank
[-n PARAM] [--accountname=PARAM]	Name des Kontos
[-c PARAM] [--ctxfile=PARAM]	Name der zu verwendenden CTX-Datei. In dieser Datei werden die abgerufenen Informationen speichert. Andere Kommandos verwenden diese Datei als Eingabe.
[-i PARAM] [--infile=PARAM]	Name der DTAZV-Datei
[-p PARAM] [--cdir]	Angabe des Ordners, in dem die Tageszählerdateien erwartet und gespeichert werden.

10.4.6.2 Beispiele

- Senden der Datei test.dtazv für das Konto 1234567890 bei der SPK WHV:

```
martin@gwenhywfar:~$ aqbanking-cli senddtazv
                     -b 28250110
                     -a 1234567890
                     -c test.ctx
                     -i test.dtazv
```

- Das gleiche Kommando für den Fall, dass man nur ein einziges Konto bei der Sparkasse WHV hat:

```
martin@gwenhywfar:~$ aqbanking-cli senddtazv
                     -b 28250110
                     -c test.ctx
                     -i test.dtazv
```

- Das gleiche Kommando für den Fall, dass man überhaupt nur ein Konto hat:

```
martin@gwenhywfar:~$ aqbanking-cli senddtazv
                     -c test.ctx
                     -i test.dtazv
```

10.4.7 listbal

Mit diesem Befehl kann man sich die Kontostände anschauen, die mit dem Befehl *request* abgerufen und in einer CTX-Datei gespeichert wurden.

Die Liste besteht aus einem Kontostand pro Zeile, die Felder einer Zeile werden durch ein TAB (Zeichen ASCII 9) getrennt.

Felder:

1	„Account“ (fester String)
2	Bankleitzahl des Kontos
3	Kontonummer
4	Name der Bank (falls bekannt)
5	Name des Kontos (falls bekannt)
6	Datum des gebuchten Kontostandes (DD.MM.JJJJ)
7	Uhrzeit des gebuchten Kontostandes (hh:mm)
8	Betrag des gebuchten Kontostandes
9	Währung des gebuchten Kontostandes
10	Datum des notierten Kontostandes (DD.MM.JJJJ)
11	Uhrzeit des notierten Kontostandes (hh:mm)
12	Betrag des notierten Kontostandes
13	Währung des notierten Kontostandes

10.4.7.1 Optionen

[-b PARAM] [--bank=PARAM]	Bankleitzahl
[-a PARAM] [--account=PARAM]	Kontonummer
[-N PARAM] [--bankname=PARAM]	Name der Bank
[-n PARAM] [--accountname=PARAM]	Name des Kontos
[-c PARAM] [--ctxfile=PARAM]	Name der zu verwendenden CTX-Datei. In dieser Datei werden die abgerufenen Informationen speichert. Andere Kommandos verwenden diese Datei als Eingabe.

[-o PARAM] [--outfile=PARAM]	Name der Ausgabedatei. Wenn diese Option nicht angegeben ist, werden die Daten an die Standardausgabe gesendet (stdout)

10.4.7.2 Beispiele

- Ausgabe der empfangenen Kontostände für Konten der Sparkasse WHV:

```
martin@gwenhywfar:~$ aqbanking-cli listbal
                     -b 28250110
                     -a 1234567890
                     -c test.ctx
                     -o test.out
```

- Ausgabe aller empfangenen Kontostände in die Datei test.out:

```
martin@gwenhywfar:~$ aqbanking-cli listbal
                     -c test.ctx
                     -o test.out
```

10.4.8 listtrans

Dieser Befehl gibt empfangene Umsätze aus einer CTX-Datei aus. Dabei kann das Ausgabe-Format frei aus den möglichen Export-Modulen von AqBanking gewählt werden. Standardmäßig wird das Export-Modul *csv* mit dem Profil *default* verwendet.

10.4.8.1 Optionen

[-b PARAM] [--bank=PARAM]	Bankleitzahl
[-a PARAM] [--account=PARAM]	Kontonummer
[-N PARAM] [--bankname=PARAM]	Name der Bank
[-n PARAM] [--accountname=PARAM]	Name des Kontos
[-c PARAM] [--ctxfile=PARAM]	Name der zu verwendenden CTX-Datei. In dieser Datei werden die abgerufenen Informationen speichert. Andere Kommandos verwenden diese Datei als Eingabe.
[-o PARAM] [--outfile=PARAM]	Name der Ausgabedatei. Wenn diese Option nicht angegeben ist, werden die Daten an die Standardausgabe gesendet (stdout)
[--exporter=PARAM]	Name des Export-Modules (Standard: *csv*)
[--profile=PARAM]	Name des Export-Profiles (Standard: *default*)
[--profile-file=PARAM]	Name der Profildatei, in der das Profil mit dem unter "--profile" angegebenen Namen gesucht werden soll. Wenn diese Option fehlt, wird das Profil in den systemweit installierten Profilen gesucht

10.4.8.2 Beispiele

- Ausgabe der Umsätze für das Konto 1234567890 bei der Sparkasse WHV:

```
martin@gwenhywfar:~$ aqbanking-cli listtrans
                     -b 28250110
                     -a 1234567890
                     -c test.ctx
```

- Ausgabe der Umsätze mit dem systemweiten Profil *comdirect* des CSV-Exporters in die Datei *test.out*:

```
martin@gwenhywfar:~$ aqbanking-cli listtrans
                     -c test.ctx
                     -o test.out
                     --exporter=csv
                     --profile=comdirect
```

10.4.9 chkacc

Mit diesem Befehl kann vorab eine Kombination aus Bankleitzahl und Kontonummer geprüft werden.

10.4.9.1 Optionen

[--rcountry=PARAM]	ISO-Code des Staates in dem das Konto geführt wird (Standard: de)
--rbank=PARAM	Bankleitzahl des zu testenden Kontos
--raccount=PARAM	Kontonummer des zu testenden Kontos

10.4.9.2 Rückgabecodes

0	Kombination ist ok
1	Fehler bei den Argumenten
2	Fehler bei der Initialisierung von AqBanking
3	Gesichert ungültige Kombination
4	Unbekannte Bank oder unbekannt, ob die Kombination stimmt
5	Fehler bei der Deinitialisierung von AqBanking

10.4.9.3 Beispiele

- prüfe Konto 1234567890 bei der Sparkasse WHV:

```
martin@gwenhywfar:~$ aqbanking-cli chkacc
                     --rbank=28250110
                     --raccount=1234567890
```

10.4.10 chkiban

Mit diesem Befehlt kann eine IBAN auf Gültigkeit geprüft werden.

10.4.10.1 Optionen

--iban=PARAM	Zu testende IBAN

10.4.10.2 Rückgabecodes

0	Kombination ist ok
1	Fehler bei den Argumenten
2	Fehler bei der Initialisierung von AqBanking
3	ungültige IBAN
5	Fehler bei der Deinitialisierung von AqBanking

10.4.11 transfer

Mit diesem Befehl kann eine einzelne Überweisung von der Kommandozeile ausgeführt werden.

Das Ergebnis findet sich in der CTX-Datei und läßt sich durch das Kommando *listtransfers* betrachten.

10.4.11.1 Optionen

[-b PARAM] [--bank=PARAM]	Bankleitzahl
[-a PARAM] [--account=PARAM]	Kontonummer
[-N PARAM] [--bankname=PARAM]	Name der Bank
[-n PARAM] [--accountname=PARAM]	Name des Kontos
[-c PARAM] [--ctxfile=PARAM]	Name der zu verwendenden CTX-Datei. In dieser Datei werden die abgerufenen Informationen speichert. Andere Kommandos verwenden diese Datei als Eingabe. Diese Option kann nicht zusammen mit [--usedb] verwendet werden.
--rname=PARAM	Name des Empfängers
[--rcountry=PARAM]	ISO-Code des Staates in dem das Konto des Empfängers geführt wird (Standard: de)
--rbank=PARAM	Bankleitzahl des Empfängers
--raccount=PARAM	Kontonummer des Empfängers
--value=PARAM	Betrag und Währung (z.B. „123,45:EUR“)
-p PARAM --purpose=PARAM	Verwendungszweck (kann mehrfach angegeben werden)
[-t PARAM] [--textkey=PARAM]	Textschlüssel für die Überweisung (Standard: 51)
[--force-check]	Wenn angegeben, dann muss die Bankverbindung des Empfängers auf jeden Fall positiv überprüfbar sein (dazu wird KtoBlzCheck verwendet). Schlägt die Prüfung fehl, wird die Überweisung nicht an die Bank weitergegeben.
[--usedb]	Ist diese Option vorhanden, werden die zu sendenden Aufträge in der internen Datenbank gespeichert. Diese Option kann nicht zusammen mit [-c] verwendet werden.

10.4.11.2 Rückgabecodes

0	Auftrag ausgeführt (Ergebnis siehe CTX-Datei)
1	Fehler bei den Argumenten
2	Fehler bei der Initialisierung von AqBanking
3	Fehler bei der Ausführung des Befehles
4	Unbekannte Bank oder unbekannt, ob die Bankverbindung des Empfängers korrekt ist
5	Fehler bei der Deinitialisierung von AqBanking

10.4.12 transfers

Mit diesem Befehl können mehrere Überweisungen aus einer Datei ausgeführt werden. Das Ergebnis findet sich in der CTX-Datei und läßt sich durch das Kommando *listtransfers* betrachten.

10.4.12.1 Optionen

[-b PARAM] [--bank=PARAM]	Bankleitzahl
[-a PARAM] [--account=PARAM]	Kontonummer
[-N PARAM] [--bankname=PARAM]	Name der Bank
[-n PARAM] [--accountname=PARAM]	Name des Kontos
[-c PARAM] [--ctxfile=PARAM]	Name der zu schreibenden CTX-Datei. Diese enthält anschließend die Liste der ausgeführten/versuchten Überweisungen Diese Option kann nicht zusammen mit [--usedb] verwendet werden.
[-f PARAM] [--infile=PARAM]	Name der Eingabedatei. Wenn diese Option nicht angegeben ist, werden die Daten in der Standardeingabe erwartet (stdin).
[--importer=PARAM]	Name des Import-Modules (Standard: *csv*)
[--profile=PARAM]	Name des Import-Profiles (Standard: *default*)
[--profile-file=PARAM]	Name der Profildatei, in der das Profil mit dem unter "--profile" angegebenen Namen gesucht werden soll. Wenn diese Option fehlt, wird das Profil in den systemweit installierten Profilen gesucht
[--force-check]	Wenn angegeben, dann muss die Bankverbindung jedes Empfängers auf jeden Fall positiv überprüfbar sein (dazu wird KtoBlzCheck verwendet). Schlägt die Prüfung fehl, wird keine Überweisung an die Bank weitergegeben.
[--usedb]	Ist diese Option vorhanden, werden die zu sendenden Aufträge in der internen Datenbank gespeichert. Diese Option kann nicht zusammen mit [-c] verwendet werden.

10.4.12.2 Rückgabecodes

0	Auftrag ausgeführt (Ergebnis siehe CTX-Datei)
1	Fehler bei den Argumenten
2	Fehler bei der Initialisierung von AqBanking
3	Fehler bei der Ausführung des Befehles
4	Unbekannte Bank oder unbekannt, ob die Bankverbindung des Empfängers korrekt ist
5	Fehler bei der Deinitialisierung von AqBanking

10.4.13 listtransfers

Mit diesem Kommando können die Überweisungen/Lastschriften betrachtet werden, die vorher gesendet wurden.

Dabei kann das Ausgabe-Format frei aus den möglichen Export-Modulen von AqBanking gewählt werden. Standardmäßig wird das Export-Modul *csv* mit dem Profil *cli-transfers* verwendet.

10.4.13.1 Optionen

[-b PARAM] [--bank=PARAM]	Bankleitzahl
[-a PARAM] [--account=PARAM]	Kontonummer
[-N PARAM] [--bankname=PARAM]	Name der Bank
[-n PARAM] [--accountname=PARAM]	Name des Kontos
[-c PARAM] [--ctxfile=PARAM]	Name der einzulesenden CTX-Datei. Diese Datei muß vorher durch eines der Kommandos *transfer*, *transfers*, *debitnote* oder *debitnotes* erzeugt worden sein.
[-o PARAM] [--outfile=PARAM]	Name der Ausgabedatei. Wenn diese Option nicht angegeben ist, werden die Daten an die Standardausgabe gesendet (stdout)
[--exporter=PARAM]	Name des Export-Modules (Standard: *csv*)
[--profile=PARAM]	Name des Export-Profiles (Standard: *cli-transfers*)
[--profile-file=PARAM]	Name der Profildatei, in der das Profil mit dem unter "--profile" angegebenen Namen gesucht werden soll. Wenn diese Option fehlt, wird das Profil in den systemweit installierten Profilen gesucht
[-S PARAM] [--status=PARAM]	hiermit kann nach Status der Überweisung gefiltert werden. Gültige Angaben sind: – *none* (keine Angabe, zeigt alle an) – *pending* (zeigt vorgemerkte Auftrage an) – *accepted* (zeigt bestätigt akzeptierte A. an) – *rejected* (zeigt zurückgewiesene A. an)

10.4.13.2 Beispiele

- Ausgabe der Überweisungen für das Konto 1234567890 bei der Sparkasse WHV:

```
martin@gwenhywfar:~$ aqbanking-cli listtransfers
                     -b 28250110
                     -a 1234567890
                     -c test.ctx
```

- Ausgabe der Umsätze mit dem systemweiten Profil "comdirect" des CSV-Exporters in die Datei "test.out":

```
martin@gwenhywfar:~$ aqbanking-cli listtransfers
                     -b 28250110
                     -a 1234567890
                     -c test.ctx
                     -o test.csv
                     --exporter=csv
                     --profile=comdirect
```

10.4.14 debitnote

Mit diesem Befehl kann eine einzelne Lastschrift von der Kommandozeile ausgeführt werden.

Das Ergebnis findet sich in der CTX-Datei und läßt sich durch das Kommando *listtransfers* betrachten.

Optionen und Beispiele siehe Befehl *transfer*.

10.4.15 debitnotes

Mit diesem Befehl können mehrere Lastschriften aus einer Datei ausgeführt werden.

Das Ergebnis findet sich in der CTX-Datei und läßt sich durch das Kommando *listtransfers* betrachten.

Optionen und Beispiele siehe Befehl *transfers*.

10.4.16 versions

Dieses Kommando zeigt die verwendeten Version an von:

- Gwenhywfar
- AqBanking
- AqBanking-CLI

10.4.17 addtrans

Mit diesem Befehl kann eine einzelne Überweisung von der Kommandozeile in einen ImExporter-Context eingetragen werden.

Das Ergebnis findet sich in der CTX-Datei und läßt sich durch das Kommando

listtransfers betrachten.

10.4.17.1 Optionen

[-b PARAM] [--bank=PARAM]	Bankleitzahl
[-a PARAM] [--account=PARAM]	Kontonummer
[-N PARAM] [--bankname=PARAM]	Name der Bank
[-n PARAM]	Name des Kontos

[--accountname=PARAM]	
[-c PARAM] [--ctxfile=PARAM]	Name der zu verwendenden CTX-Datei. In dieser Datei werden die abgerufenen Informationen speichert. Andere Kommandos verwenden diese Datei als Eingabe.
--rname=PARAM	Name des Empfängers
[--rcountry=PARAM]	ISO-Code des Staates in dem das Konto des Empfängers geführt wird (Standard: de)
--rbank=PARAM	Bankleitzahl des Empfängers
--raccount=PARAM	Kontonummer des Empfängers
--value=PARAM	Betrag und Währung (z.B. „123,45:EUR“)
-p PARAM --purpose=PARAM	Verwendungszweck (kann mehrfach angegeben werden)
[-t PARAM] [--textkey=PARAM]	Textschlüssel für die Überweisung (Standard: 51)
[--force-check]	Wenn angegeben, dann muss die Bankverbindung des Empfängers auf jeden Fall positiv überprüfbar sein (dazu wird KtoBlzCheck verwendet). Schlägt die Prüfung fehl, wird die Überweisung nicht an die Bank weitergegeben.

10.4.17.2 Rückgabecodes

0	Auftrag ausgeführt (Ergebnis siehe CTX-Datei)
1	Fehler bei den Argumenten
2	Fehler bei der Initialisierung von AqBanking
3	Fehler bei der Ausführung des Befehles
4	Unbekannte Bank oder unbekannt, ob die Bankverbindung des Empfängers korrekt ist
5	Fehler bei der Deinitialisierung von AqBanking

10.4.18 fillgaps

Mit diesem Befehl werden Umsätze/Überweisungen in einem bestehenden ImExporter-Context durch bekannte Informationen aus der AqBanking-Konfiguration aufgefüllt. Beispielsweise können dadurch die IBAN, die BIC oder andere Informationen in die jeweiligen Transaktionen eingetragen werden, falls diese fehlen.

10.4.18.1 Optionen

[-c PARAM] [--ctxfile=PARAM]	Name der zu verwendenden CTX-Datei. In dieser Datei werden die abgerufenen Informationen gespeichert. Andere Kommandos verwenden diese Datei als Eingabe.

10.4.18.2 Rückgabecodes

0	Auftrag ausgeführt (Ergebnis siehe CTX-Datei)
1	Fehler bei den Argumenten
2	Fehler bei der Initialisierung von AqBanking
3	Fehler bei der Ausführung des Befehles
4	Unbekannte Bank oder unbekannt, ob die Bankverbindung des Empfängers korrekt ist
5	Fehler bei der Deinitialisierung von AqBanking

10.4.19 dblisttrans

Dieser Befehl gibt Umsätze aus der internen Datenbank aus. Dabei kann das Ausgabe-Format frei aus den möglichen Export-Modulen von AqBanking gewählt werden. Standardmäßig wird das Export-Modul *csv* mit dem Profil *default* verwendet.

10.4.19.1 Optionen

[-q PARAM]	Ausdruck, der die zu exportierenden Umsätze definiert (z.B. **$date==“20081031“** um alle Umsätze vom 31.10.2008 zu exportieren). Siehe Kapitel über Auswahlausdrücke.
[-o PARAM] [--outfile=PARAM]	Name der Ausgabedatei. Wenn diese Option nicht angegeben ist, werden die Daten an die Standardausgabe gesendet (stdout)
[--exporter=PARAM]	Name des Export-Modules (Standard: *csv*)
[--profile=PARAM]	Name des Export-Profiles (Standard: *default*)
[--profile-file=PARAM]	Name der Profildatei, in der das Profil mit dem unter "--profile" angegebenen Namen gesucht werden soll. Wenn diese Option fehlt, wird das Profil in den systemweit installierten Profilen gesucht

10.4.19.2 Beispiele

- Ausgabe der Umsätze für das Konto 1234567890 bei der Sparkasse WHV:

```
martin@gwenhywfar:~$ aqbanking-cli dblisttrans
                     -q „\($localAccountNumber==\“1234567890\“) && (\
                     $localBankCode==\“28250110\“)“
```

Beachten Sie hier, daß Sie Sonderzeichen, die normalerweise von der Eingabeaufforderung verwendet werden, einen Backslash voranstellen müssen (also „\ $“ statt nur „$“, weil die Eingabe aufforderung sonst versucht eine Umgebungsvariable einzusetzen).

– Ausgabe der Umsätze mit dem systemweiten Profil *comdirect* des CSV-Exporters in die Datei *test.out*:

```
martin@gwenhywfar:~$ aqbanking-cli dblisttrans
                     -o test.out
                     --exporter=csv
                     --profile=comdirect
```

10.4.20 dblisttransfers

Dieses Kommando arbeitet ähnlich wie dblisttrans, gibt allerdings Überweisungsaufträge aus statt der Umsatzdaten.

10.4.21 dbrecon

Dieses Kommando dient dem automatischen Abgleich noch offener Überweisungen und Lastschriften der internen Datenbank mit Hilfe von Umsatzdaten.

Dazu geht dieses Kommando die Liste der offenen Aufträge durch (also solche, bei den die Bank gemeldet hat, daß der Auftrag zwar angenommen aber noch nicht ausgeführt wurde, d.h. $status==4) durch und sucht nach passenden Umsatzdaten.

Dabei ist dieses Kommando auch in der Lage Sammelaufträge abzugleichen.

Werden passende Umsatzdaten gefunden, wird der Status des jeweiligen Auftrages geändert auf „5“ (=automatisch abgeglichen). Außerdem wird in das Feld „$valutadate“ des Auftrages das aktuelle Datum eingetragen.

Dieses Kommando funktioniert natürlich nur, wenn Sie bei den Kommandos „transfer“, „transfers“, „debitnote“, „debitnotes“ und „request“ auch den Schalter „--usedb“ verwendet haben, weil nur dann die entsprechenden Daten auch in der internen Datenbank landen.

10.5 Allgemeine Rückgabecodes

0	Auftrag ausgeführt
1	Fehler bei den Argumenten
2	Fehler bei der Initialisierung von AqBanking
3	Fehler bei der Ausführung des Befehles (z.B. wenn eine Anfrage nicht an die Bank gesendet werden kann etc)
4	Fehler bei der Nachbearbeitung eines Befehles (z.B. beim Schreiben der CTX- oder Ausgabedatei)
5	Fehler bei der Deinitialisierung von AqBanking

11 Profile für den CSV-Importer/Exporter

11.1 Vorbereitete Profile

AqBanking liefert eine Reihe von Profilen mit. Profile für Importer und Exporter finden sich im Installationsverzeichnis:

„$PREFIX/share/aqbanking/imexporters/*IMPORTERNAME*/profiles“.

Im Falle von CSV-Profilen lautet der Pfad demnach:

„$PREFIX/share/aqbanking/imexporters/***csv***/profiles“. Jede der Dateien in diesem Verzeichnis repräsentiert ein Profil. Die Datei „full.conf“ enthält beispielsweise das Profil mit dem Namen "*full*“.

11.2 Genereller Aufbau

Sie sehen hier den einfachsten Fall einer CSV-Profildatei.

Es handelt sich dabei um eine einfache Konfigurationsdatei wie sie von AqBanking, Libchipcard und anderen Projekten verwendet wird.

Sie besteht aus Variablen und Gruppen. Variablen haben meist ihren Typen vorangestellt (z.B. „int“, „char“).

Gruppen werden durch den Namen und eine folgende offene geschweifte Klammer eingeleitet und durch eine geschlossene geschweifte Klammer beendet.

Gruppen dürfen beliebig tief verschachtelt sein.

In der Beispieldatei gibt es als mehrere Variablen (z.B. „name“, „import“ etc) und zwei Gruppen („params“ und „columns“, wobei „columns“ eine Untergruppe von „params“ ist).

Durch das Gatterzeichen eingeleitete Kommentare werden ignoriert.

Die Variablen außerhalb der Gruppe „params“ gelten für die Profildateien aller Formate, wohingegen der Inhalt der Gruppe „params“ spezifisch ist für den jeweiligen Im-/Exporter (in diesem Fall: CSV).

```
char name=“PSK“
char shortDescr=“Oesterr. Postbank“
char longDescr=“CSV-Format PSK“
int import=“1“
int export=“1“

char groupNames=“transaction“
char dateFormat=“DD.MM.YYYY“
int utc=“0“

char subject=“transactions“
char valueFormat=“float“

params {
  int quote=“0“
  int title=“0“
  char delimiter=“;“

  columns {
    1=“localAccountNumber“
    2=“purpose“
    3=date“
    4=“valutaDate“
    5=“value/value“
    6=“value/currency“
  } # columns
} # params
```

Text 1: Einfache CSV-Profildatei

11.3 Wichtige allgemeine Variablen

11.3.1 name, shortDescr, import, export

Grundsätzlich muss eine solche Profildatei mindesten die Variable „name“ enthalten sowie „shortDescr“. Vorhanden sein sollten ebenfalls „import“ und „export“, damit AqBanking feststellen kann, ob mit diesem Profil der Import und/oder Export möglich ist.

11.3.2 DateFormat

Diese Variable gibt an, wie das Datum in den zu importierenden/exportierenden Dateien zu interpretieren ist.

Im Beispiel wird das Format „DD.MM.YYYY“ verwendet. Dabei steht ein „D“ jeweils für eine Ziffer des Tages im Monat, „M“ für jeweils eine Ziffer des Monats und „Y“ für jeweils eine Ziffer des Jahres. Ein korrektes Datum nach diesem Format wäre also „01.01.2008“.

Es gibt allerdings auch Sonderfälle: Das obige Format geht von führenden Nullen aus, also „01.01.2008“ statt „1.1.2008“.

Wenn man nun aber auch diesen Fall abdecken möchte, empfiehlit sich folgendes Format: „*D.*M.YYYY“. In diesem Fall steht beispielsweise „*D“ für eine unbegrenzte Anzahl von Ziffern. Trifft der Datumsparser auf ein „*“ liest er die ganze Zahl bis zur nächsten nicht-Ziffer; beim Datum „01.12.2008“ im Format „*D.*M.YYYY“ würde also zunächst die „01“ eingelesen, bis der erste Punkt erreicht ist. Im Formatstring steht an dieser Stelle das „D“, und somit weiß der Parser, daß es sich um die Angabe des Tages handelt.

Wird das „*“ nicht verwendet, geht der Parser von einer festen Anzahl von Ziffern aus.

11.3.3 Utc

Diese Variable gibt an, ob die Zeitangaben in den CSV-Dateien in der lokalen Zeitzone des Benutzers liegen oder in der Universal Time Coodinated. Normalerweise ist ersteres der Fall (also utc=0).

11.3.4 ValueFormat

Intern speichert AqBanking Beträge als sogenannte rationale Zahlen ab. Das bedeutet in diesem Zusammenhang, dass ein Betrag in Format *Zähler/Nenner* gespeichert wird.

Will man CSV-Dateien exportieren, um sie mit anderen Anwendungen weiterzubearbeiten, macht diese Darstellung unter Umständen Schwierigkeiten. Daher kann mit dieser Variable auch das bekannte Fließkommaformat angefordert werden. Der wert dafür wäre in diesem Fall „float“.

Der Wert „rational“ ist die Standardvorgabe und führt zur Speicherung als *Zähler/Nenner*.

11.4 Wichtige Variablen für CSV

Wie beschrieben finden sich die spezifischen Variablen für das CSV-Modul innerhalb

der Gruppe „params“.

11.4.1 Quote

Dies betrifft insbesondere den Export im CSV-Format. Hat diese Variable den Wert „1“ werden alle Spalten in Anführungszeichen verpackt. Bei einem Wert von „0“ unterbleibt dies.

11.4.2 Title

In manchen CSV-Dateien enthält die erste Zeile die Überschriften der einzelnen Spalten. Dies erhöht die Lesbarkeit für den Benutzer, aber ein Import-Modul muß diese Zeile ignorieren.

Hat diese Variable den Wert „1“ wird die erste Zeile beim Import übergangen, beim Wert von „0“ nicht.

Beim Export wird diese Variable ebenfalls ausgewertet: Hat sie den Wert „1“ wird eine Zeile mit den Überschriften der einzelnen Spalten vorangestellt.

11.4.3 Delimiter

Mit dieser Variable kann angegeben werden, welches Zeichen als Spaltentrenner dient. In den meisten Fällen ist es ein Semikolon oder en Komma. Manche Dateien verwenden auch ein Tabulator-Zeichen (ASCII: 9) oder das Leerzeichen.

Sonderwerte sind daher „TAB“ für Tabulator-Zeichen und „SPACE“ für Leerzeichen.

11.4.4 IgnoreLines

Manche CSV-Dateien enthalten ein paar Zeilen am Anfang, die nicht im CSV-Format vorliegen. Diesen sogenanten Kopfsatz kann man mit dieser Variable ignorieren lassen. Geben Sie dazu die Anzahl der Zeilen an, die am Anfang der Datei ignoriert werden sollen.

11.4.5 Gruppe „columns“

Diese Gruppe enthält für jede Spalte der CSV-Datei die Bezeichnung des Inhaltes, wie sie intern von AqBanking verwendet wird. Die folgenden Tabellen geben einen Überblick über die erlaubten Werte. Die am häufigsten vorkommenden Werte sind **fettgedruckt**.

11.4.5.1 Allgemeine Werte

Name	***Beschreibung***
LocalCountry	Ländercode des eigenen Kontos
LocalBankCode	Bankleitzahl des eigenen Kontos
LocalAccountNumber	Kontonummer des eigenen Kontos
LocalIBAN	IBAN des eigenen Kontos
LocalBIC	BIC (SWIFT-Code) der eigenen Bank
LocalName	Name des Kontoinhabers des eigenen Kontos

RemoteCountry	Ländercode des Gegenkontos
RemoteBankCode	Bankleitzahl des Gegenkontos
RemoteAccountNumber	Kontonummer des Gegenkontos
RemoteIBAN	IBAN des Gegenkontos
RemoteBIC	BIC (SWIFT-Code) der Gegenbank
RemoteName[0]	Name des Kontoinhabers des Gegenkontos, erste Zeile
UniqueId	Eindeutige Kennung, kann von Anwendungen gesetzt werden, wird von AqBanking selbst ansonsten nicht verwendet
ValutaDate	Datum der Wertstellung
Date	Datum der Buchung
Value/Value	Die Gruppe „value“ enthält zwei Variablen: „value“ und „currency“. Diese Angabe hier enthält folglich den Betrag, und ...
Value/Currency	... dieser die Währung (3-Zeichen-ISO-Code, also „EUR“ für den Euro, „USD“ für US-Dollar etc)
TextKey	Optionaler Textschlüssel, wird nicht von allen Modulen unterstützt
TransactionKey	HBCI: Buchungsschlüssel
CustomerReference	Kundenreferenz
BankReference	Bankreferenz
TransactionCode	Nur HBCI: spezieller 3-stelliger Geschäftsvorfallcode
TransactionText	Buchungstext (wird nicht von allen Banken übertragen, kann dann aber Werte annehmen wie „LASTSCHRIFT“ oder „DAUERAUFTRAG“
PrimaNota	Primanota
Fiid	Eindeutige Kennung des Vorfalles, vergeben von der Bank (wird nur im OFX-Protokoll verwendet, nicht aber beispielsweise im deutschen HBCI)
Purpose[0]	Verwendungszweck, 1. Zeile (es können beliebig viele Zeilen im CSV-Format verwendet werden)
Category[0]	Kategorie, 1. Zeile (wird von Anwendungen wie QBankManager gesetzt, kann aber auch in manchen Dateiformaten vorkommen. Meistens ist es aber leer)

11.4.5.2 Zusätzliche Namen für Überweisungen/Lastschriften

Name	***Beschreibung***
Type	Transaktionstyp (z.B. „transaction“, „transfer“, „debitNote“, „euTransfer“)
SubType	Untertyp (wird nicht von allen Modulen verwendet), z.B. „check“, „bookedDebitNote“, „standingOrder“ etc.
Status	Bei Aufträgen: Status des Auftrages, z.B. „accepted“, „rejected“, „pending“

11.5 Spezielfälle

11.5.1 Getrennte Spalten für positive/negative Beträge

Manche Banken verwenden unterschiedliche Spalten für Zugänge und Abgänge. Damit kann AqBanking seit Version 3.7.1 umgehen.

Setzen Sie in diesem Fall die allgemeine Variable „int splitValueInOut=1“.

In der CSV-spezifischen Gruppe „params/columns“ können Sie dann statt „value/value“ jeweils eigene Werte für Zu- und Abgänge verwenden: „valueIn/value“ für Eingänge und „valueOut/value“ für Ausgänge (siehe Beispiel *bankaustria.conf*).

11.5.2 Zusatzfeld für die Angabe des Betragsvorzeichens

Die niederländische Bank „Mijnpostbank“ verwendet ein Format, welches das Vorzeichen in einer zusätzlichen Spalte kodiert. Auch dies wird von AqBanking unterstützt.

Verwenden Sie in diesem Fall die allgemeinen Variablen:

```
Columns {
  1=“date“
  2=“remoteName“
  3=“localAccountNumber“
  4=“remoteAccountNumber“
  5=“posNeg“
  6=“value/value“
  7=“purpose“
}
```

Text 2: Spalten bei eigenem Vorzeichenfeld

Variable	Bedeutung
Int usePosNegField=“1“	Verwendung einer eigenen Spalte für das Vorzeichen
char posNegFieldName=“posNeg“	Name der Spalte, wie er in der Gruppe „params/columns“ verwendet wird
char positiveValues=“C“	Wenn in der Spalte „posNeg“ dieser Wert steht („C“), wird der Wert als positiv angesehen. Es können hier auch mehrere, durch Kommata getrennte Werte angegeben werden
char negativeValues=“D“	Wenn in der Spalte „posNeg“ dieser Wert steht („D“), wird der Wert als negativ angesehen. Es können hier auch mehrere, durch Kommata getrennte Werte angegeben werden
int defaultIsPositive=“1“	Wenn nichts angegeben ist (das Feld also leer ist oder der Wert darin nicht in *positiveValues* oder *negativeValues* auftaucht) ist der Wert positiv (wenn hier „0“ steht: negativ).

Die Gruppe „params/columns“ könnte in diesem Beispiel wie nebenstehend aussehen.

11.5.3 Spaltentausch nach Betrag

Die Postbank Frankfurt/Main tauscht Empfänger und Auftraggeber in Abhängigkeit vom Betrag. Seit 4.1.3 kann AqBanking auch damit umgehen.

Dazu werden die folgenden allgemeinen Variablen verwendet:

Variable	Bedeutung
int switchLocalRemote=“1“	Schaltet das Vertauschen der Angaben der Spalten *localName* und *remoteName* ein
int switchOnNegative="0"	Wenn die obige Variable eingeschaltet ist (also einen Wert ungleich *0* hat) kann man hiermit angeben, in welchem Fall getauscht werden soll: Ist der Wert hier *1*, wird bei negativen Werten getauscht, ansonsten bei positivem Werten.