Sonderdruck
aus
Ausgabe 17/00

DDBAC für Fortgeschrittene

Das Wurzelobjekt von DDBAC ist BACBanking. Alle folgenden Objekte klinken sich hier ein, etwa die HBCI-Kontakte mit den einzelnen Konten. Sie befinden sich im Objekt BACCustomer. Den Datenaustausch mit dem Rechenzentrum erledigt das Objekt BACDialog, das ebenfalls über BACBanking initialisiert wird. In den Access-Formularen erzeugt die Zeile

```
Dim objBanking As New BACBanking
```

zuerst eine Instanz von BACBanking. Die einzelnen Kontakte in Customers erreicht man über

```
Set aContact = objBanking.Customers(Index)
```

Und die entsprechenden Konten mit der Zeile

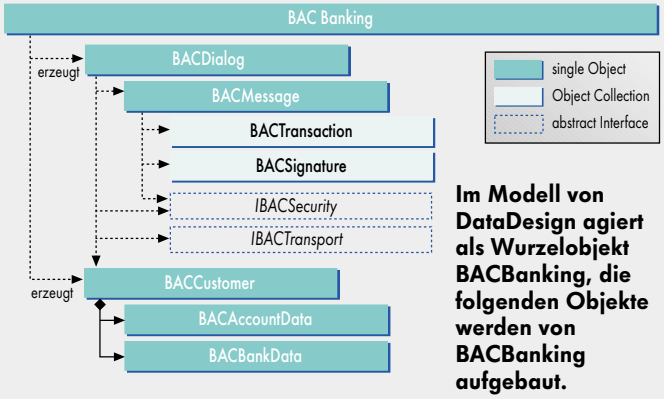
```
Set aAccount = aContact.AccountData.
Segments(Index)
```

Die Instanz aAccount enthält das HIUPD-Segment (Userparameterdaten). Dieses speichert mehr oder weniger ausführlich Kontodaten. Die Bank kann zum Beispiel übermitteln, für welche Geschäftsvorfälle der Benutzer autorisiert ist. Manche Banken lassen den Benutzer allerdings nicht wissen, was er eigentlich darf.

In der Access-Datenbank macht das Formular frmShowHBCIData nichts anderes, als die Daten des HIUPD-Segmentes auszugeben. Die Ansteuerung eines bestimmten Kontos erweist sich als umständlich. Während ein Kontakt über die Funktion FindCustomer gefunden wird, wartet DDBAC mit keinem Äquivalent für Konten auf. Zu Bestimmung eines Kontos reicht die Kontonummer allein nicht aus, man muss alle Konten eines Kontaktes durchgehen und dabei Kontonummer, Bankleitzahl und Währung in Betracht ziehen. Beispielsweise besitzen Großbanken mehrere Bankleitzahlen, das heißt, die Bankleitzahl des Kontaktes muss nicht zwangsläufig mit der des Kontos übereinstimmen. Auch die Kombination von Bankleitzahl und Kontonummer ist nicht eindeutig.

Manche Banken führen Fremdwährungskonten mit derselben Nummer. Eindeutigkeit bringt erst die Angabe des Länderkennzeichens; für Deutschland ist gemäß ISO-Norm die '280' vorgesehen. Da aber HBCI bisher nur in Deutschland eingesetzt wird und die europaweite Anwendung zurzeit nicht mehr als ein frommer Wunsch ist, kann man auf die Abfrage des Landes vorerst verzichten, daher reichen Kontonummer, Bankleitzahl und Währung.

Bevor frmShowHBCIData Informationen preisgibt, muss das Konto in der Datenbank angelegt werden. Dafür sind die Formulare frmAccountmanager und frmEditAccount zuständig. Die Anlage und die Synchronisation eines Kontaktes kann man getrost DDBAC



Im Modell von DataDesign agiert als Wurzelobjekt BACBanking, die folgenden Objekte werden von BACBanking aufgebaut.

Quelle: DataDesign

überlassen; den Auftrag dazu gibt die Zeile

```
objBanking.RunNewContactWizard
```

Wenn der Kontakt erfolgreich angelegt und synchronisiert ist, stehen die Kontodaten im HIUPD-Segment auf der Festplatte. Das Formular frmEditAccount liest sie ähnlich wie frmShowHBCIData aus und speichert sie in einer Liste, aus der sich das gewünschte Konto auswählen lässt.

Kontaktaufnahme

Der Hauptteil der HBCI-Kommunikation verbirgt sich im Formular frmTransmitData. Der Benutzer kann entscheiden, was an die Bank übertragen werden sollen. Dazu berechnet das Formular beim Öffnen alle möglichen Aktionen, beispielsweise Saldo- und Umsatzabfrage sowie die Übertragung anstehender Überweisungen. Alle zusätzliche Aktionen müssen hier berücksichtigt werden.

Hat der Benutzer entschieden, was er machen will, und die Übertragung mit einem Klick auf 'Verbindung aufbauen' gestartet, geht das Programm jeden Kontakt durch und schaut nach, ob Aufträge vorhanden sind. Ist dies der Fall, kontaktiert es das Rechenzentrum.

Die Implementierung dieser Routine in VisualBasic erweist sich als umständlich, was vor allem an der PIN-Abfrage liegt, denn eine normale Inputbox zeigt die Eingabe in Klartext an. Also muss frmTransmitData das Formular frmEnterPIN aufrufen, das jedes Eingabezeichen als Sternchen anzeigt; allerdings macht das zusätzliche Formular eine echte For-Schleife zunichte.

NextContact liefert jeweils den nächsten Kontakt. Falls für diesen Aufträge vorliegen, erscheint die PIN-Abfrage, die wiederum ProcessContact startet und die eingegebene PIN übergibt. Als Erstes erzeugt ProcessContact mit

Sonderdruck aus Ausgabe 17/00


```
Set objDialog =
objCustomer.NewDialog(sPIN)
```

eine neue BACDialog-Instanz und beginnt den Dialog mit

```
objDialog.BeginDialog
```

Gemäß der HBCI-Spezifikation kann die Bank nun nach Belieben Nachrichten schicken, die das Programm in der Statusbox anzeigt. Auf jede gesendete Nachricht antwortet der Bankrechner, zum Beispiel bei Salden- und Umsatzabfrage über ein spezielles Antwortsegment oder nur über eine kurze Rückmeldung.

Die Variable objDialog.HBCIResultCode enthält die Rückmeldung der letzten Sendung. Ist alles in Ordnung, kommt vom Rechenzentrum entweder die Zeichenfolge ‘10’ oder ‘20’. Höhere Nummern bedeuten meist nichts Gutes: Warnungen beginnen mit ‘3’, Fehler mit ‘9’. Zu dieser Nummer gesellt sich noch erklärender Text, der mit objDialog.HBCIResult abgefragt wird.

In Aktion

Ist die Verbindung aufgebaut, geht ProcessContact jede einzelne Aktion durch. Über Action legt man fest, welches Segment gefüllt wird. Auch hier lässt sich das Programm leicht erweitern. Die Zeile

```
Set objSegment = objBanking.NewSegment
("HKSAL", objCustomer("HBCIVersion"))
```

liefert ein Segment vom Typ HKSAL. Damit fragt der Kunde seinen Kontostand ab. Bevor es aber ans Verschicken geht, muss das leere Segment noch gefüllt werden. Das kann so aussehen:

```
objSegment("AuftraggeberKontoverbindung1",
"Kontonummer1") = qryTransmit!Account-
number
```

DDBAC liefert eine HTML-Dokumentation der einzelnen Segmente mit. Über die verschiedenen HBCI-Versionen reihen sich nicht nur neue Segmenttypen ein, bereits bestehende ändern bisweilen auch ihr Gesicht. Mitunter muss das Kundensystem dies berücksichtigen, zum Beispiel durch:

```
If objSegment.Version >= 3 Then objSegment
("Kontowaehrung1") = qryTransmit!Currency
```

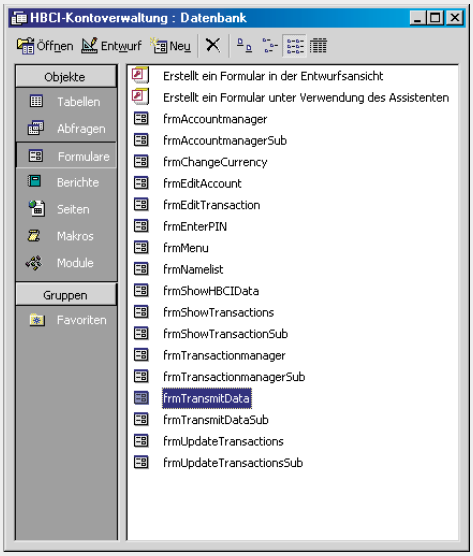
Erst ab Segmentversion 3 gibt es das Datenelement ‘Kontowaehrung’. Die Segmentversionen entsprechen nicht den HBCI-Versionen: HBCI 2.01 versteht zum Beispiel das HKSAL-Segment Version 3.

Ist das Segment in der entsprechenden Unterroutine gefüllt, schickt es das Programm mit

```
Set objMessage = objDialog.ExecuteSegment
(objSegment)
```

auf die Reise. Anschließend muss die Antwort bearbeitet werden. Bei anderen Aktionen als der Überweisung kommt nur eine kurze Meldung über objDialog.HBCIResultCode. Nach Abschluss der Aktion ruft ProcessContact mit NextContact den nächsten Kontakt auf.

Mit PrepareHKUEB wird ein Überweisungssegment gefüllt. Die Banken akzeptieren jeweils unterschiedliche Textschlüssel. Über ein eigenes Segment kann sich der Kunde über die Gepflogenheiten seiner Bank informieren. Wir haben es uns einfach gemacht und verwenden ausschließlich die Kodierung ‘51’ für ‘Überweisung’. Die müsste jedes Kreditinstitut akzeptieren. Außerdem beschneidet die Routine die Verwendungszweckzeilen auf 27 Zeichen. Den Abruf der Umsätze bereitet PrepareHKKAZ vor, es ist für die Saldenabfrage zuständig.



Die Hauptarbeit im selbst gestrickten HBCI-Banking leistet das Formular frmTransmitData.

ProcessHIKAZ füllt die Tabelle tReceivedTransactions mit den empfangenen Umsätzen. Hier benutzt HBCI das Fremdformat MT-940. Das Antwortsegment besteht nur aus einem großen Eintrag. Um ihn zu lesen, stellt DDBAC das Object BACSwiftStatement bereit:

```
Set objStatement =
objResponseSegment("UmsaetzeGebucht1")
```

Die einzelnen Datensätze werden über die Instanz objLine abgefragt, die über alle Einträge in objStatement läuft. Die Banken geben sich beim Füllen der einzelnen Felder unterschiedlich große Mühe. Davon hängt dann auch die Präsentation im Programm ab.

Das separate Formular frmUpdateTransactions übernimmt die in tReceivedTransations zwischengespeicherten Umsätze in das eigentliche Kontobuch. Eventuelle Duplikate lassen sich löschen, neue Einträge übernehmen. Die Tabelle tBusinessTransactionCodes enthält viele der üblichen Textschlüssel. Hieraus entnimmt das Formular die Langtexte der einzelnen Textschlüssel.