

DDBAC DataDesign Banking Application Components

Callbackschnittstelle

Übersicht

Titel	Thema	Datei	Autor
DDBAC DataDesign Banking Application Components	Klasse BACStorage	BACStorage.doc	Eugen Schabenberger

Seitenumfang	Version	Status	Datum
104 Seiten	4.1.7	Release	13.04.2007

Änderungsverzeichnis

Version	Datum	Autor	Änderungen / Kommentare
0.2.0	02.09.2005	Schabenberger, Eugen	Erste Version des Dokuments
4.1.7	22.03.2006	Schabenberger, Eugen	Erweiterungen

Abhängige Dokumente

Dateiname	Beschreibung
-----------	--------------

Inhaltsverzeichnis

1 Einführung	6
1.1 Allgemeines	6
1.2 Bankingkontakte	6
1.3 Kompatibilität zu bestehenden Applikationen	7
1.4 Implementierung	7
1.5 Erstellen eines BACCustomer Objects	9
2 BACContact	9
2.1 Create	10
2.2 Load	10
2.3 Save	10
2.4 Delete	11
2.5 NewWizard	11
2.6 Synchronize	11
2.7 ChangePin	12
2.8 Edit	12
2.9 LockKeys	12
3 BACContacts	12
3.1 Populate	12
3.2 FindContact	13
4 Befehle	14
4.1 ListCustomersRq	14
4.2 ReadCustomerRq	16
4.3 CreateCustomerRq	16
4.4 UpdateCustomerRq	18
4.5 DeleteCustomerRq	19
4.6 IsValidCustomerRq	20
4.7 NewContactWizardRq	21
4.8 CreateDialogRq	22
4.9 GetBankInfoRq	24
4.10 SyncCustomerWizardRq	25
4.11 ChangePinRq	27
4.12 LoadStringTableRq	28
4.13 OpenTanDialogRq	29
4.14 QueryPinRq	32
4.15 RequestTANListRq	33
4.16 ActivateTANListRq	34
4.17 LockTANListRq	35
4.18 ShowTANListRq	36
4.19 LockPINRq	37
4.20 UnlockPINRq	38
4.21 ChangeTANMethodRq	39
4.22 GenerateIniLetterRq	40
4.23 ChangeConnectionRq	41
4.24 EditContactRq	42
4.25 LockKeysRq	43
4.26 ChangeUserIdRq	44
4.27 ChangeHbciVersionRq	45
4.28 FilelistRq	46
4.29 OpenAboutDialogRq	46
4.30 ChangeKeysRq	48
4.31 UpgradeKeysRq	49

5	XAML Dialog Format	50
5.1	<i>Canvas</i>	50
5.2	<i>Button</i>	50
5.3	<i>TextBox</i>	51
5.4	<i>TextBlock</i>	51
5.5	<i>Image</i>	52
5.6	<i>ListBox</i>	52
5.7	<i>ComboBox</i>	53
5.8	<i>CheckBox</i>	53
5.9	<i>RadioButton</i>	53
5.10	<i>Label</i>	54
5.11	<i>Canvas</i>	54
6	XAML Dialoge erstellen	54
6.1	<i>Wizards</i>	55
6.1.1	WizardFrame	55
6.1.2	EnterBlz	58
6.1.3	GetBankInfo	60
6.1.4	SelectConnect	62
6.1.5	EnterID	63
6.1.6	EnterPin2	65
6.1.7	EnterPin2 (Zweischritt TAN Verfahren)	67
6.1.8	EnterPin2 mit Klasse 1	68
6.1.9	EnterPin2 mit Klasse 2	69
6.1.10	SelectContact	70
6.1.11	Synchronize	71
6.1.12	BankNotice	72
6.1.13	EnterNewPin	73
6.1.14	EnterRetypePIN	74
6.1.15	ChangePin	75
6.1.16	VerifyBankkey	76
6.1.17	SendIniBrief	77
6.1.18	SelectHBCIVersion	78
6.1.19	EnterCommAddress (TCP/IP)	79
6.1.20	EnterCommAddress (HTTP)	80
6.1.21	EnterSocks5	81
6.1.22	EnterKeyfile	82
6.1.23	EnterKeySize	83
6.1.24	EnterKonto	84
6.1.25	SaveKeyFile	86
6.1.26	ChangeKeys	87
6.1.27	UpgradeKeys	88
6.1.28	Finish	89
6.1.29	Error	91
6.1.30	EnterTANListNr	92
6.1.31	EnterOldTAN	93
6.1.32	ChangeTANMethod	94
6.1.33	EditContact	95
6.1.34	LockKeys	96
6.2	<i>Dialoge</i>	97
6.2.1	dlgEnterTan	97
6.2.2	ErrorReport	99
6.2.3	dlgAbout	100
7	Implementierung	101
7.1	<i>XAML Dialoge verändern</i>	101
7.2	<i>Zweischritt (Mehrfach) TAN Eingabe</i>	102
7.2.1	Synchrone ein- oder mehrfach TAN Eingabe	102
7.2.2	Asynchrone Mehrfach TAN Eingabe	103

1 Einführung

1.1 Allgemeines

Die Callbackschnittstelle der DDBAC erlaubt es der Applikation in bestimmte Vorgänge innerhalb der DDBAC einzugreifen. Die Schnittstelle kann dazu verwendet werden um das Speichern der Kontaktdaten in die eigene Applikation zu verlegen oder um alle PIN oder TAN Eingaben innerhalb der Applikation zu beantworten, damit der Anwender seine PIN hinterlegen kann und nicht mehr danach gefragt wird.

Die DDBAC bietet zudem eine Reihe von Wizards an um administrative Vorgänge wie das Neuanlegen eines Kontakts oder eine PIN – Änderung für einen bestehenden Kontakt. Die Callbackschnittstelle kann dazu verwendet werden, um die dabei gezeigten Dialoge an das Layout der jeweiligen Applikation anzupassen.

1.2 Bankingkontakte

Die Daten aller angelegten Kontakte (Customers) in der DDBAC werden in einer Datei ddusers.dat gespeichert und im Verzeichnis für Anwendungsdaten gespeichert. Zusätzlich werden für jeden Kontakt auch noch die BPD und UPD Informationen der Bank in diesem Verzeichnis abgelegt. Darin sind alle relevanten Informationen zu allen Konten eines Anwenders im Klartext enthalten, was sicherheitstechnisch bedenklich ist.

Bisher gab es nur die Möglichkeit den Speicherort mittels eines RegistryKeys zu verändern, die Daten wurden aber auf jeden Fall im Klartext abgelegt. Mit den neuen Funktionen der DDBAC soll es möglich sein, dass die Anwendungen die Benutzerdaten in ihren eigenen Datenbanken speichern, bzw. dass die Daten in dem Verzeichnis verschlüsselt abgelegt werden.

Folgende Anforderungen werden von DDBACStorage erfüllt:

- Die Daten verschlüsselt speichern
- Die Daten optional direkt in der Anwendung zu speichern, wo es oft schon einen Datentresor gibt, der diese Informationen aufnehmen kann
- Die Daten in Dialogen zu bearbeiten
-

Diese Funktionen werden von der neuen BACStorage Klasse übernommen. Für bestehende Anwendungen bedeutet das, dass ohne eine Änderung der Applikation die Daten in Zukunft verschlüsselt abgelegt werden. Das wird optional abschaltbar sein und es wird dann wieder das bisherige Format verwendet. Im Moment ist diese Funktionalität aus Kompatibilitätsgründen noch nicht aktiviert.

Um zu bestimmen, ob die Daten verschlüsselt in XML Form oder wie bisher in der ddusers.dat gespeichert werden, wird der RegistrySchlüssel: „USE_DDUSERS_DAT“ REG_SZ unter Software/DataDesign/DDBAC verwendet. Wenn dieser eine „1“ enthält, werden die Daten wie bisher aus der ddusers.dat gelesen und gespeichert.

Ansonsten wird die ddusers.xml verwendet. Wenn diese Datei noch nicht vorhanden ist, werden alle Kontakte aus der ddusers.dat automatisch importiert und stehen dann zur Verfügung. Alle Änderungen an den Kontaktdaten werden dann in die ddusers.xml Datei geschrieben. Wann die ddusers.dat dann endgültig gelöscht wird ist noch nicht geklärt.

Zum Verschlüsseln der ddusers.xml wird ein fester Schlüssel verwendet, der im Code hinterlegt ist und jeder der die DDBAC API verwendet kann die Daten auslesen ohne dass nach einem Passwort gefragt wird. Es ist für die DDBAC nicht mehr möglich zum jetzigen Zeitpunkt hier eine Passwortabfrage einzubauen, da dies alle darauf basierenden Applikationen stören würde. Es bietet insofern mehr Sicherheit, als dass ein einfacher Trojaner der die DDBAC nicht kennt, die Daten nicht einfach unverschlüsselt auslesen kann. Der Trojaner muss zumindest auf die DDBAC angestimmt sein.

Eine Speicherung der Daten mit einem variablen Schlüssel in einem Datentresor kann zukünftig die Applikation selbst übernehmen. Dazu müssen der DDBAC einige Funktionen per Callback zur Verfügung gestellt werden, über diese die DDBAC dann die Daten Lesen und Schreiben kann. Zusätzlich gibt es für alle Applikationen über diese Schnittstelle die Möglichkeit die Dialoge oder Wizards der DDBAC zum Bearbeiten der Daten aufzurufen. Dazu gehören:

- Anlegen von neuen Kontakten
- Bearbeiten der Kontaktdaten
- Synchronisierung
- Passwortänderung

Und als letztes neues Feature, ist es nun auch möglich die Dialoge der DDBAC selbst zu implementieren und diese anstatt den Dialogen der DDBAC zu verwenden.

1.3 Kompatibilität zu bestehenden Applikationen

BACStorage dient grundsätzlich als Erweiterungsmöglichkeit für ihre Applikation und ist keinesfalls verpflichtend. Bereits bestehende Applikationen funktionieren ohne Änderung weiter. Es gibt jedoch einen wichtigen Unterschied zu früher, der möglicherweise zu Problemen führen kann.

Früher wurde bei BACCustomer alle Werte die mit PutField gesetzt wurden direkt in die ddusers.dat geschrieben. Es gibt leider keine Befehle in der bestehenden Schnittstelle, mit welchem man einen BACCustomer explizit abspeichern kann. Einen solchen Befehl hinzuzufügen hätte nichts genützt, da alle bereits bestehenden Applikationen diesen Befehl ja nicht implementiert hätten.

Würde man jedes Property sofort speichern, dann würde mit der neuen Schnittstelle bei jedem setzen eines Wertes ein UpdateCustomerRq gemacht. Das würde bei zehn zu setzenden Werten 10 Aufrufe von UpdateCustomer bedeuten und ist so nicht realisierbar.

Von daher werden die gesetzten Werte erst einmal nur zwischengespeichert und erst wenn das BACCustomer Objekt endgültig freigegeben wird, dann wird erkannt, dass sich die Werte geändert haben und dann wird ein UpdateCustomerRq erzeugt, der die Daten endgültig abspeichert. Dies kann und ungünstigen Umständen zu Problemen führen, die nicht komplett für alle Applikationen durchgetestet werden konnte.

Aufgrund der bisherigen Implementierung des Homebanking Administrators kommt es auch zu redundanten Aufrufen von Update, List und Readcustomer. Das war schon immer so und wird durch die neue Schnittstelle erst sichtbar. Diese Aufrufe wurden jedoch vorerst noch nicht optimiert damit die Kompatibilität der einzelnen Komponenten untereinander erhalten bleibt.

1.4 Implementierung

In der DDBAC gibt es eine neue, statische Klasse BACStorage, die ähnlich wie die BACMonitor Klasse angelegt ist. Diese Klasse enthält folgende Funktionen:

```
[id(1)] HRESULT List ([in] String sFilter, [out] String *psData);
[id(2)] HRESULT Read ([in] String sID, [out] String *psData);
[id(3)] HRESULT Create ([in] String sData, [out] String *psResult);
[id(4)] HRESULT Update ([in] String sData, [out] String *psResult);
[id(5)] HRESULT Delete ([in] String sData, [out] String *psResult);
[id(6)] HRESULT Execute ([in] long hWnd, [in] String sData, [out] String *psResult);
[id(7)] HRESULT CallDefaultImpl ([in] long hWnd, [in] String sData, [out] String
*psResult);
```

Diese Funktionen sind sehr einfach gehalten, damit auch zukünftige Anforderungen umgesetzt werden können, ohne dass das Interface immer wieder verändert werden muss, was dazu führt, dass die Anwendungen inkompatibel werden. Als Parameter werden hier wie in der DDBACXML XML Dokumente übergeben, die dann endgültig definieren, was passieren soll.

Die DDBAC verwendet diese Klasse intern um mit beispielsweise mit Read die Liste der Customer zu füllen, diese mit Update wieder zu speichern, mit Create anzulegen und mit Delete zu löschen. Dies kann für Anwendungsprogramme interessant sein, es ist aber nicht notwendig diese Funktionen direkt aufzurufen.

Interessant ist hier vor allem die Funktion Execute, die verwendet werden kann um die Daten eines Customers zu bearbeiten, eine Synchronisation durchzuführen oder das Passwort zu ändern. Die DDBAC verwendet zukünftig an allen Stellen in welchen Dialoge benötigt werden, diese Funktion. Die Executefunktion ist mit dem Execute() in der DDBACXML nahezu identisch.

Die Dialoge selbst wurden in eine neue DLL verlagert, welche DDBACDLG.DLL heißt. Diese .DLL wird von DataDesign signiert und diese Signatur wird vor dem Verwenden eines Dialogs überprüft um sicherzustellen, dass die Datei nicht verändert oder ausgetauscht wurde.

Damit wird die DDBAC resistenter gegen Viren die sich an Dateien anhängen und sich damit in Prozesse eingliedern und gegen einen kompletten Austausch von Dlls. Es ist allerdings nicht sicher gegen Viren die sich schon im Prozess eingemischt haben und Aufrufe an das Betriebssystem umleiten.

Die BACStorage Klasse enthält einen ConnectionPoint für um diese Funktionen in der jeweiligen Applikation auszuführen:

DBACStorageEvents:

```
[id(1)] void OnCallback ([in] long hWnd, [in] String sData, [out] String *psResult);
```

Dies funktioniert in VB analog zur BACMonitor Klasse. An einer Stelle im Programm wird die Klasse mit new und „with events“ erzeugt und eine Funktion Storage_OnCallback wird, sofern vorhanden, immer aufgerufen, wenn die DDBAC Daten eines Customers benötigt, wenn Daten gespeichert werden müssen, usw.

Dabei werden in den Parametern XML Daten ausgetauscht, welche im Laufe der Zeit an neue Erfordernisse angepasst werden können, ohne dass das Interface dabei verändert werden muss. Der restliche Teil der Dokumentation beschäftigt sich nun mit dem Aufbau dieser XML Daten. Mit dem Registryeintrag "TraceStorage" unter Software/Datadesign/DDBAC können die Xml Aufrufe in der Hbcilog.txt Datei mitprotokolliert werden.

Diese Events können dazu verwendet werden um die Funktionen der DDBAC komplett zu ersetzen. Wenn die DDBAC im Homebankingdialog nur Kontakte aus einer Applikation anzeigen soll, dann

kann die Applikation einfach alle Aufrufe die mit <ListCustomersRq> beginnen abfangen und die Antwort selbst erstellen.

Es ist aber auch möglich, das Kommando abzufangen und CallDefaultImpl() aufzurufen. Als Antwort erhält die Applikation dann das XML das die DDBAC normalerweise zurückliefern würde. Die Applikation kann dann die Daten bei Bedarf nachbearbeiten und dann modifiziert zurückliefern. Dies ist in ddBACTest beispielhaft implementiert.

1.5 Erstellen eines BACCustomer Objects

Bisher konnte das BACCustomer Objekt nur über das BACBanking Interface erzeugt werden. Mit der neuen DDBAC kann das BACCustomer Objekt auch mit CreateObject erzeugt werden. Anschließend kann man das Property "XML" mit einem <Customer> XML füllen, wie es in den Beispielen hier verwendet wird. Dieser Customer existiert dann nur im Speicher und kann für alle Funktionen in der DDBAC verwendet werden.

2 BACContact

Das BACContact Objekt ist eine übergeordnete Klasse über dem BACCustomer, es beinhaltet alle Funktionen des BACCustomers und einige neue Funktionen. Das BACCustomer Objekt ist eher als ein temporäres Objekt zu sehen mit welchen die Transaktionen durchgeführt werden und das anschließend verworfen wird, während das BACContact Objekt einen dauerhaft gespeicherten Kontakt repräsentiert.

Alle Funktionen die mit den BACCustomer ausgeführt werden, welche Dialoge der DDBAC erzeugen wie AuthenticateUI, CreateDialogUI erzeugen weiterhin die bisher verwendeten Dialoge, damit Anwendungen, welche mit einer älteren Version der DDBAC entwickelt wurden unverändert funktionieren.

Alle neuen Funktionen im BACContact verwenden jedoch neue Dialoge und Funktionen, welche hier im Weiteren vorgestellt werden. Die ganzen dort beschriebenen XML Requests und Responses sind im BACContact Objekt gekapselt. Wenn sie nur die neuen Funktionen verwenden möchten, dann genügt es völlig, die Funktionen des BACContact Objektes aufzurufen. Diese erzeugen dann die jeweils notwendigen Aufrufe.

Das bedeutet, es genügt völlig wenn ihre Anwendung zum anlegen eines neuen Kontakts folgendes macht:

```
BACContact aContact = new BACContact();  
aContact.NewWizard (this.Handle);
```

Mit diesem Aufruf kann der Anwender mit einem Wizard einen neuen Kontakt anlegen, welcher dann dauerhaft gespeichert ist und im Weiteren zur Durchführung von Transaktionen mit der Bank verwendet werden kann.

Wenn Sie jedoch kontrollieren möchten, wo die Daten dieses BACContact gespeichert werden und wie die Dialoge genau aussehen und wenn Sie eventuell schon Informationen haben, welche Daten in dem Kontakt vorkommen, zum Beispiel die Bankleitzahl, dann sind die folgenden Kapitel wichtig,

weil dort beschrieben wird, wie diese Funktionen im einzelnen ablaufen und an welchen Stellen sie in diesen Ablauf eingreifen können.

Ansonsten sind Sie nach diesem Kapitel fertig mit lesen.

Die Sourcecode Beispiele für Create, Load, Save und Delete finden sie in der Applikation ddBACTest unter dem Menüpunkt Funktionen / Contact Load/Save. Wenn Sie das Monitor Fenster öffnen und anschließend den Menüpunkt aufrufen, wird sichtbar welche Transaktionen dabei ausgeführt werden.

2.1 Create

Mit diesem Aufruf kann ein neuer Kontakt dauerhaft gespeichert werden.

```
BACContact aContact = new BACContact();
aContact["CountryCode"] = "280";
aContact["BankCode"] = "12345678";
aContact["UserID"] = "12345";
aContact["Contact"] = "MyContactName";
aContact.Create();
```

Damit wird der Kontakt angelegt. Mit Sicherheit wird er so nicht funktioniert um mit einer Bank zu kommunizieren, aber die grundlegenden Informationen sind schon hinterlegt. Dieser Kontakt erscheint im Homebankingadministrator und kann dort weiterbearbeitet, korrigiert oder auch wieder gelöscht werden.

Im Normalfall ist dieser Kontakt anschließend in der Datei <eigeneDateien>/<Anwendungsdaten>/DDBAC/ddUsers.dat gespeichert. Im Normalfall bedeutet, dass die Funktionen der DDBAC nicht überschrieben wurden und die Daten des Kontakts an einer anderen Stelle gespeichert wurden.

2.2 Load

Mit diesem Aufruf kann ein bereits gespeicherter Kontakt wieder geladen werden. Dabei muss die eindeutige ID des Kontakts bekannt sein. Diese ID wird beim Anlegen eines Kontakts neu und eindeutig vergeben.

In obigen Fall ist die ID "280:12345678:12345".

```
BACContact aContact = new BACContact();
aContact.Load ("280:12345678:12345");
if (aContact["BankCode"] != "12345678")
    throw new Exception("Dokumentation ist fehlerhaft!");
```

2.3 Save

Damit kann ein geladener, geänderter Kontakt wieder gespeichert werden.

```
BACContact aContact = new BACContact();
aContact.Load ("280:12345678:12345");
aContact["Contact"] = "MyAndererContactName";
aContact.Save ();
```

2.4 Delete

Und schließlich kann man mit diesem Aufruf einen dauerhaft gespeicherten Kontakt wieder löschen.

```
BACContact aContact = new BACContact();
aContact.Load ("280:12345678:12345");
aContact.Delete ();
```

2.5 NewWizard

Mit diesem Befehl wird der Wizard gestartet um einen neuen BACContact anzulegen. Dabei wird der Anwender nach allen notwendigen Informationen gefragt die zur Einrichtung des Kontakts notwendig sind.

Wenn der Kontakt erfolgreich eingerichtet wurde, wird BACDialogResults.OK zurückgeliefert. Falls ein Fehler aufgetreten ist oder der Dialog abgebrochen wurde, wird BACDialogResults.Cancel geliefert.

Falls OK gemeldet wird, wurde der Kontakt bereits dauerhaft gespeichert es kann jedoch sein, dass der Kontakt noch nicht synchronisiert wurde. Dies kann über das Feld "NeedSynchronisation" abgefragt werden. Sollte darin eine "1" enthalten sein, ist der Kontakt noch nicht synchronisiert und kann nicht verwendet werden.

Falls vor dem Anlegen eines neuen Kontakts bereits bestimmte Daten bekannt sind, können diese in den Feldern des Kontakts bereits vorbelegt werden. Diese Werte werden dann in den einzelnen Wizardseiten in den Eingabefeldern vorgeschlagen. Siehe dazu die Beschreibung der einzelnen Wizardseiten in den folgenden Kapiteln.

2.6 Synchronize

Alle Kontakte müssen vor ihrer Verwendung mit der Bank synchronisiert werden. Dabei werden die BPD (Bankparameterdaten) und die UPD (Userparameterdaten/Konten) für einen bestimmten Kontakt ausgetauscht und je nach Zugangsart noch einige andere Werte abgeglichen. Erst nach einer erfolgreichen Synchronisation kann der Kontakt verwendet werden.

Mit diesem Aufruf wird eine solche Synchronisation in einem Wizard durchgeführt, dazu muss das BACContact Objekt einen existierenden Kontakt enthalten, der entweder über Load oder von der BACContacts Liste geladen wurde.

Wenn der Kontakt erfolgreich synchronisiert wurde, wird BACDialogResults.OK zurückgeliefert. Falls ein Fehler aufgetreten ist oder der Dialog abgebrochen wurde, wird BACDialogResults.Cancel geliefert.

2.7 ChangePin

Für die meisten Zugangsarten kann der Anwender die PIN selbst vergeben, je nach Zugangsart wird die PIN auf die Chipkarte geschrieben oder an die Bank übermittelt.

Mit diesem Aufruf wird eine solche Pin Änderung in einem Wizard durchgeführt, dazu muss das BACContact Objekt einen existierenden Kontakt enthalten, der entweder über Load oder von der BACContacts Liste geladen wurde.

Wenn die PIN des Kontakts erfolgreich geändert wurde, wird BACDialogResults.OK zurückgeliefert. Falls ein Fehler aufgetreten ist oder der Dialog abgebrochen wurde, wird BACDialogResults.Cancel geliefert.

Die Prüfung ob die Passphrase geändert werden kann, kann über das Feld "CanChangePassphrase" geprüft werden. Wenn eine "1" zurückgeliefert wird, ist eine Änderung möglich.

2.8 Edit

Noch nicht implementiert, steht als Platzhalter für eine zukünftige Version in welcher mit diesem Aufruf die Kontaktdaten manuell geändert werden können. Derzeit kann dafür der Homebanking-Administrator verwendet werden.

2.9 LockKeys

Noch nicht implementiert, steht als Platzhalter für eine zukünftige Version in welcher mit diesem Aufruf die Schlüssel gesperrt werden können. Derzeit kann dafür der Homebanking-Administrator verwendet werden.

3 BACContacts

Diese Collection liefert eine Liste aller dauerhaft gespeicherten Kontakte. Diese Liste ist anfangs leer und muss mit Populate gefüllt werden.

3.1 Populate

Liest alle Kontakte die der Filterbedingung entsprechen und stellt kopiert diese im Speicher. In der Standardimplementierung ist keine Filterbedingung implementiert, es werden also immer alle Kontakte zurückgeliefert, es ist jedoch möglich für eigene Implementierungen diese Filterbedingung auszuwerten.

Die Liste der Kontakte ist ab diesem Zeitpunkt nur eine Kopie im Speicher und enthält alle Kontakte die zum Zeitpunkt von Populate vorhanden waren. Es können mehrere BACContacts Listen gleichzeitig existieren. Werden Kontakte neu angelegt oder gelöscht, so bleiben diese dennoch in der BACContacts Collection erhalten. Diese muss mit Populate immer neu geladen werden um den aktuellen Stand der Kontakte zu enthalten.

3.2 FindContact

Mit diesem Aufruf kann der Index eines bestimmten Kontakts innerhalb der Liste gesucht werden. Diese muss zuvor mit Populate gefüllt worden sein.

Als Parameter wird die eindeutige "ID" des Kontakts erwartet und zurückgeliefert wird die Position innerhalb der Liste oder -1 wenn der Kontakt nicht gefunden wird. (0 = erstes Element).

4 Befehle

Diese Kapitel enthält die komplette Übersicht über alle XML Request und Response Aufrufe die von der DDBAC verwendet werden.

4.1 ListCustomersRq

Liste aller Customer.

Wird von BACContacts verwendet wenn Populate aufgerufen wird. Der Parameter "Filter" wird im XML im Attribut Filter übergeben.

```
<ListCustomersRq Filter=""></ListCustomersRq>
```

Es wird die komplette Liste aller Customer im folgenden Format erwartet:

```
<ListCustomersRs>
  <Customers>
    <CustomerHeader>
      <ID>280:333xxxxx:87654321</ID>
      <Contact>Cust1</Contact>
      <CountryCode>280</CountryCode>
      <BankCode>33355555</BankCode>
      <UserID>87654321</UserID>
      <SecurityMediaType>4</SecurityMediaType>
      <NeedSynchronisation>1</NeedSynchronisation>
    </CustomerHeader>
    <CustomerHeader>
      <ID>280:79900010:123456</ID>
      <Contact>Cust2</Contact>
      <CountryCode>280</CountryCode>
      <BankCode>79900010</BankCode>
      <UserID>123456</UserID>
      <NeedSynchronisation>1</NeedSynchronisation>
    </CustomerHeader>
  </Customers>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</ListCustomersRs>
```

Unter Status wird immer der Fehlercode zurückgeliefert. 0 bedeutet generell "kein Fehler".

Ansonsten können hier alle Fehlercodes von Windows auftreten, wobei der Statuscode das HRESULT und die Beschreibung in Statusdesc enthalten sind. Bei logischen Fehlern enthält der StatusCode eine 1 und in der Beschreibung ist die Begründung für den Fehler enthalten. Diese Annahmen gelten für alle XML Aufrufe.

ID ist eine eindeutige ID die bei späteren Aufrufen dazu dienen soll, den Customer eindeutig zu identifizieren.

CustomerHeader bedeutet, dass nicht alle Daten eines Customers geliefert werden, sondern nur die Teile die zum Beispiel zum Anzeigen in einer Liste notwendig sind. Mit Customer.Field() kann dann auf die Felder zugegriffen wird, die hier zurückgeliefert wurden. Die hier angegebenen Felder werden vom Homebanking Administrator verwendet um die Liste zu füllen.

Wenn auf ein Feld zugegriffen wird, das nicht vorhanden ist, dann wird mit ReadCustomer der komplette Customer angefordert. Das ist der folgende Befehl.

Es ist auch möglich bei diesem Aufruf gleich den kompletten Customer zurückzuliefern. Dazu wird im XML nicht CustomerHeader sondern gleich der komplette Customer geliefert. In diesem Fall wird ReadCustomer nicht mehr aufgerufen.

Bei Customer.Field gilt weiterhin, dass beliebig benannte Felder hinzugefügt werden können. Diese werden dann an das XML angefügt, leere Felder werden wieder gelöscht.

4.2 ReadCustomerRq

```
<ReadCustomerRq ID="280:70000999:35081"></ReadCustomerRq>
```

Damit werden alle Daten eines Customers erwartet. Die umfasst alle Felder aus ddusers.dat, die bpd's, die upd's und die zusätzlichen Informationen aus FinTS 4.0. In ID wird der Inhalt des Feldes mitgeliefert, der bei <ListCustomersRq> im Customerheader übergeben wurde.

Das Xml der Antwort Customers ist wie folgt aufgebaut:

```
<ReadCustomerRs>
  <Customer>
    <ID>280:70000999:35081</ID>
    <UserID>35081</UserID>
    <BankCode>70000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Cust2</Contact>
    <SecurityProgID>DataDesign.BACSecurityTAN</SecurityProgID>
    <SecurityMediaID></SecurityMediaID>
    <SecurityMediaType>4</SecurityMediaType>
    <CustomerSystemStatus>1</CustomerSystemStatus>
    <BankName>DataDesign DemoBank</BankName>
    <BankUserID></BankUserID>
    <HBCIVersion>220</HBCIVersion>
    <CommunicationsService>3</CommunicationsService>
    <CommunicationsAddress>http://bankcomputer.de</CommunicationsAddress>
    <CommunicationsAddressSuffix>64</CommunicationsAddressSuffix>
    <CustomerSystemID>9RRPRGBVPGM31OISHBCI02</CustomerSystemID>
    <SignatureID>25</SignatureID>
    <UPD>48495550413A313.../UPD>
    <BPD>48495550413A313.../BPD>
  </Customer>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</ReadCustomerRs>
```

Alle Elemente unterhalb Customer sind dann über Customer[„Bankname“] wie bisher zu erreichen und können auch verändert werden. Es ist auch wie bisher möglich, eigene Elemente hinzuzufügen, die dann im XML angelegt und gespeichert werden.

Die UPD und BPD Daten sind ebenfalls enthalten, allerdings als HexString formatiert, wenn in diesen Daten binäre Werte vorkommen, ansonsten im Klartext. Mit Customer.GetBankData werden diese Daten dann wie gewohnt in BACSegmente umgewandelt und stehen wie bisher zur Verfügung.

4.3 CreateCustomerRq

```
<CreateCustomerRq ID="{41C3730E-780A-48ED-BE4B-E7566984A6DB}">
```

```

<Customer>
  <CountryCode>280</CountryCode>
  <BankCode>23423423</BankCode>
  <UserID>234</UserID>
  <Contact>234</Contact>
  <SecurityProgID>DataDesign.BACSecurityTAN</SecurityProgID>
  <SecurityMediaType>4</SecurityMediaType>
  <CustomerSystemStatus>1</CustomerSystemStatus>
  <HBCIVersion>220</HBCIVersion>
  <CommunicationsService>2</CommunicationsService>
  <CommunicationsAddress>http://bankcomputer.de</CommunicationsAddress>
</Customer>
</CreateCustomerRq>

```

Mit dieser Funktion wird ein neuer Customer angelegt. Der neue Customer wird in sData übergeben und ist, wie in OnRead beschrieben aufgebaut.

Die ID im Attribut des CreatCustomerRq ist nur ein optimaler Wert und enthält eine neu generierte GUID die mit CoCreateGuid(). Dies ist nur als Vorschlag für eine eindeutige ID zu sehen und erleichtert das Vergeben einer neuen ID in Sprachen, in welchen CoCreateGuid() nicht so einfach aufgerufen werden kann.

Nach dem Speichern wird in psResult der komplette <Customer> erwartet mit einer ID die beim Anlegen vergeben wurde. Eventuell können auch zusätzlich Felder hinzugefügt werden, es sollte sich um die gleichen Daten handeln, die bei einem späteren OnRead mit der gleichen ID zurückgeliefert werden.

```

<CreateCustomerRs>
  <Customer>
    <CountryCode>280</CountryCode>
    <BankCode>23423423</BankCode>
    <UserID>234</UserID>
    <Contact>234</Contact>
    <SecurityProgID>DataDesign.BACSecurityTAN</SecurityProgID>
    <SecurityMediaType>4</SecurityMediaType>
    <CustomerSystemStatus>1</CustomerSystemStatus>
    <HBCIVersion>220</HBCIVersion>
    <CommunicationsService>2</CommunicationsService>
    <CommunicationsAddress>http://bankcomputer.de</CommunicationsAddress>
  </Customer>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</CreateCustomerRs>

```

4.4 UpdateCustomerRq

Mit dieser Funktion wird ein Customer verändert. Der Customer wird in sData übergeben und ist, wie in OnRead beschrieben aufgebaut. Nach dem Speichern wird in psResult der komplette <Customer> erwartet. Eventuell können auch zusätzlich Felder hinzugefügt oder verändert werden, es sollte sich um die gleichen Daten handeln, die bei einem späteren OnRead mit der gleichen ID zurückgeliefert werden.

```
<UpdateCustomerRq>
  <Customer>
    <ID>280:23423423:234</ID>
    <UserID>234</UserID>
    <BankCode>23423423</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>234</Contact>
    <SecurityProgID>DataDesign.BACSecurityTAN</SecurityProgID>
    <SecurityMediaType>4</SecurityMediaType>
    <CustomerSystemStatus>1</CustomerSystemStatus>
    <HBCIVersion>220</HBCIVersion>
    <CommunicationsService>2</CommunicationsService>
    <CommunicationsAddress>http://bankcomputer.de</CommunicationsAddress>
    <UPD></UPD>
    <BPD></BPD>
    <CustomerID>99</CustomerID>
  </Customer>
</UpdateCustomerRq>
```

Die Antwort sieht wie folgt aus:

```
<UpdateCustomerRs>
  <Customer>
    <ID>280:23423423:234</ID>
    <UserID>234</UserID>
    <BankCode>23423423</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>234</Contact>
    <SecurityProgID>DataDesign.BACSecurityTAN</SecurityProgID>
    <SecurityMediaType>4</SecurityMediaType>
    <CustomerSystemStatus>1</CustomerSystemStatus>
    <HBCIVersion>220</HBCIVersion>
    <CommunicationsService>2</CommunicationsService>
    <CommunicationsAddress>http://bankcomputer.de</CommunicationsAddress>
    <UPD></UPD>
    <BPD></BPD>
    <CustomerID>99</CustomerID>
  </Customer>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</UpdateCustomerRs>
```

Es sollte in Customer der Customer zurückgeliefert werden, wie er tatsächlich gespeichert wurde falls beispielsweise noch neue Felder hinzugefügt werden, dann sollten diese auch zurückgeliefert werden.

4.5 DeleteCustomerRq

Mit dieser Funktion wird ein Customer gelöscht. Der Customer wird in sData übergeben und ist, wie in OnRead beschrieben aufgebaut. Nach dem Löschen psResult nur eine Erfolgsmeldung erwartet.

```
<DeleteCustomerRq>
  <Customer>
    <ID>280:23423423:234</ID>
    <UserID>234</UserID>
    <BankCode>23423423</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>234</Contact>
    <SecurityProgID>DataDesign.BACSecurityTAN</SecurityProgID>
    <SecurityMediaType>4</SecurityMediaType>
    <CustomerSystemStatus>1</CustomerSystemStatus>
    <HBCIVersion>220</HBCIVersion>
    <CommunicationsService>2</CommunicationsService>
    <CommunicationsAddress>243</CommunicationsAddress>
    <CustomerID>99</CustomerID>
    <UPD></UPD>
    <BPD></BPD>
  </Customer>
</DeleteCustomerRq>
```

Und die Antwort:

```
<DeleteCustomerRs>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</DeleteCustomerRs>
```

4.6 IsValidCustomerRq

Hiermit wird vor dem Erstellen eines Contacts geprüft, ob dieser schon vorhanden ist. In der Standardimplementierung muss die Bankleitzahl, UserID und der Kontakname eindeutig sein. Für andere Implementierungen können hier aber eigene Regeln festgelegt werden, allerdings muss dann auch das Laden und Speichern implementiert werden, da sich der eindeutige Schlüssel in der Standardimplementierung aus diesen Werten zusammensetzt und die Kontakte sonst nicht eindeutig identifizierbar sind.

```
<IsValidCustomerRq>
  <Customer>
    <CountryCode>280</CountryCode>
    <BankCode>12345678</BankCode>
    <UserID>12345</UserID>
    <Contact>MyContactName</Contact>
  </Customer>
</IsValidCustomerRq>
```

Im Node Customer sind alle Felder des zu prüfenden Customers enthalten.

Als Antwort wird nur ein Status erwartet, wobei 0 bedeutet, dass der Customer angelegt werden darf. Alle anderen Werte in StatusCode bedeuten, dass der Customer nicht gespeichert werden darf und in StatusDesc wird eine lesbare Begründung erwartet, die dem Anwender den Grund für den Fehler begreiflich machen.

```
<IsValidCustomerRs>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</IsValidCustomerRs>
```

4.7 NewContactWizardRq

Damit wird der neue Wizard zum Anlegen eines neuen Homebankingkontakts gestartet. Ird von BACContact.NewWizard ausgeführt.

```
<NewContactWizardRq></NewContactWizardRq>
```

Als Antwort werden die Daten des neuen Customers erwartet:

```
<NewContactWizardRs>
  <Customer>
    <BankCode>70000999</BankCode>
    <CountryCode>280</CountryCode>
    <BankName>DataDesign DemoBank</BankName>
    <HBCIVersion>220</HBCIVersion>
    <CommunicationsService>3</CommunicationsService>

    <CommunicationsAddress>http://hbcio2.datadesign.de/</CommunicationsAddress>
    <SecurityProgID>DataDesign.BACSecurityTAN</SecurityProgID>
    <SecurityMediaType>4</SecurityMediaType>
    <CustomerID></CustomerID>
    <UserID>45</UserID>
    <Contact>DataDesign DemoBank</Contact>
  </Customer>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</NewContactWizardRs>
```

Oder im Falle eines Abbruchs folgendes XML:

```
<NewContactWizardRs>
  <Status>
    <StatusCode>1</StatusCode>
    <StatusDesc>CANCELED</StatusDesc>
  </Status>
</NewContactWizardRs>
```

Während dieser Aufruf aktiv ist, wird die Callbackfunktion weitere Male aufgerufen. Zum Beispiel CreateCustomerRq um den Customer zu speichern und CreateDialogRq um die einzelnen Dialoge abzurufen. Die Callbackfunktion muss in der Lage sein, nochmal aufgerufen zu werden, noch bevor der erste Aufruf beendet wurde. (reentrant).

Beim Neuanlegen eines Kontakts wird nach einer neuen PIN (Passwort) gefragt, das der Anwender im Verlauf des Assistenten eingeben muss. Da die DDBAC solche Passwörter möglichst kurz im Speicher hält und auch nicht an die Applikation zurückgibt, muss der Anwender die vergebene PIN (Passwort) noch mal in der Applikation hinterlegen.

4.8 CreateDialogRq

```
<CreateDialogRq Name="WizardFrame">
    <NewContactWizardRq></NewContactWizardRq>
</CreateDialogRq>
```

Mit CreateDialog werden die Daten zum Aufbau des Dialogs in dem Attribut "Name" erwartet. Hier ist es möglich eigene Dialoge zu entwerfen und diese dann zurückzuliefern. Für die Funktionalität ist es nur wichtig, dass für die jeweiligen Dialoge bestimmte Schlüsselfelder gleiche Namen haben. Das restliche Design der Dialog kann frei gestaltet werden.

Die Antwort auf diesen Request sieht wie folgt aus:

```
<CreateDialogRs>
    <DialogResource Font="Arial" FontSize="8">
        <Canvas Name="ROOT" Width="440" Height="314"
            xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
            xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">

            <TextBox Name="txtBlz" Width="260" Height="20" Canvas.Left="20"
                Canvas.Top="90" Text="60050101" IsReadOnly="True">
            </TextBox>

            <Label Name="Label1" Width="410" Height="25" FontWeight="Bold"
                Canvas.Left="20" Canvas.Top="20" Content="Bankleitzahl:">
            </Label>

            <Label Name="Label2" Width="160" Height="24" Canvas.Left="20"
                Canvas.Top="76" Content="Bankleitzahl (BLZ):">
            </Label>

            <TextBox Name="lblBankname" Width="260" Height="20"
                Canvas.Left="20" Canvas.Top="131">
            </TextBox>

            <Label Name="Label3" Width="165" Height="24" Canvas.Left="20"
                Canvas.Top="117" Content="Kreditinstituts:">
            </Label>

            <Label Name="Label4" Width="400" Height="24" Canvas.Left="20"
                Canvas.Top="290" Content="Fahren Sie fort, indem Sie auf
                [Weiter] klicken.">
            </Label>

        </Canvas>
    </DialogResource>
    <Status>
        <StatusCode>0</StatusCode>
        <StatusDesc>OK</StatusDesc>
    </Status>
</CreateDialogRs>
```

DialogResource enthält einen XML Dialog der durch XAML definiert ist. XAML ist ein neuer Standard von Microsoft zur Darstellung von Dialogen im neuen Windows Vista.

Die DDBAC kann nicht den kompletten Umfang von XAML darstellen. Dies würde eine Installation von WinFX erfordern, was aus Kompatibilitätsgründen kaum möglich ist. Das XAML Format wird verwendet, weil das C++ Format von MFC nicht genügend Möglichkeiten zur Formatierung bietet und auch kaum für eine Übermittlung in XML geeignet ist. Die DDBAC hätte hier im Prinzip auch ein eigenes Format erfinden können, jedoch ist die Anlehnung an XAML von Vorteil, da es dafür externe Dialog-Editoren gibt, die in Zukunft noch wachsen werden.

Für die Dialoge die derzeit enthalten sind wurde der XAML Dialogeditor von Mobiform verwendet welcher unter <http://www.mobiform.com/> heruntergeladen werden kann. Dies setzt dann jedoch tatsächlich die Installation von WinFX Beta und einigen anderen Dingen voraus.

An dieser Stelle wird es sicher noch zu einigen Anpassungen kommen, bis die Dialogeditoren das Betastadium verlassen haben. Je nach Bedarf werden in der DDBAC dann auch weitere Controls und Optionen von XAML implementiert. Wobei aber spektakuläre 3D Effekte eher nicht dazugehören werden. Vielmehr geht es um die Darstellung von Bitmaps, Auswahl, Farbe und Größe des Zeichensatzes für die Datenfelder. Zusätzlich ist es möglich den Dialoghintergrund mit einem Farbverlauf zu versehen.

Im Canvas eines XAML ist es derzeit nicht vorgesehen, dass der Zeichensatz für das Fenster angegeben werden kann. Dies wird in den Attributen von DialogResource übergeben. In Font wird das Typeface des Fonts festgelegt und in Fontsize die Fontgröße. Dieser Font wird für alle Controls verwendet, bei welchen nicht explizit ein anderes Typeface oder eine andere Größe verwendet wurde.

Sollte ihre Applikation also ausschließlich Tahoma als Font verwenden, dann würde es genügen in allen CreateDialogRq, die an die Callbackfunktion gehen, das Attribut "Font" durch den gebräuchlichen Font in ihrer Applikation auszutauschen und damit würden alle Dialoge der DDBAC die aus ihrer Applikation aufgerufen werden, mit diesem Font dargestellt.

Die Größe der Fenster und die Positionierung der Controls ist abhängig von diesem Font und der angegebenen Größe. Bei einem kleinen Font sind die Fenster kleiner, bei einem großen entsprechend größer. Eine pixelgenaue Positionierung von Bitmaps ist damit derzeit nicht möglich.

Kurzfristig wurde der Dialogname auch in einem Tag "DialogName" übergeben. Dies ist weiterhin aus Kompatibilitätsgründen möglich, allerdings verwendet die DDBAC.Net nur das Attribut "Name".

4.9 GetBankInfoRq

```
<GetBankInfoRq BankCode="70000999"></GetBankInfoRq>
```

Diese Funktion wird vom Wizard verwendet um Informationen über die Zugangsarten zu einer Bank zu ermitteln. Die Antwort sieht so aus:

```
<GetBankInfoRs>
  <BankInfo>
    <BankName>DataDesign DemoBank</BankName>
    <BankConnect>
      <Name>PTAN</Name>
      <HBCIVersion>220</HBCIVersion>
      <CommunicationsService>3</CommunicationsService>
      <CommunicationsAddress>http://bank.de/</CommunicationsAddress>
      <CommunicationsAddressSuffix>64</CommunicationsAddressSuffix>
      <Security SecurityProgID="DataDesign.BACSecurityTAN"
        SecurityMediaType="4" Code="TAN" Version="1">
      </Security>
    </BankConnect>
    <BankConnect>
      <Name>HBCI</Name>
      <HBCIVersion>220</HBCIVersion>
      <CommunicationsService>2</CommunicationsService>
      <CommunicationsAddress>bank.de</CommunicationsAddress>
      <Security Code="RDH" Version="1"></Security>
    </BankConnect>
  </BankInfo>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</GetBankInfoRs>
```

Diese Funktion verwendet die Daten aus der DDBACBLZ und versucht einen anonymen Dialog mit der Bank zu führen und prüft die verfügbaren Zugangsarten ab. In dem Tool ddBACTest gibt es einen Dialog, mit welchem die Zugangsdaten zu allen bekannten Bankleitzahlen überprüft werden können.

Der weitere Ablauf des Wizards wird über diese Daten gesteuert. Wenn kein HBI Zugang vorhanden ist, wird der Anwender beispielsweise gar nicht danach gefragt.

Diese Funktion wurde implementiert, damit Applikationen selbst auf diese Informationen zugreifen können und damit die Daten der DDBAC gegebenenfalls von der Applikation angepasst oder ergänzt werden können.

4.10 SyncCustomerWizardRq

Mit diesem Aufruf kann ein Kontakt synchronisiert werden. Dazu werden einfach die Daten des Customers wie folgt übergeben:

```
<SyncCustomerWizardRq>
  <Customer>
    <ID>280:70000999:35081</ID>
    <UserID>35081</UserID>
    <BankCode>70000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>TestHKTAN</Contact>
    <SecurityProgID>DataDesign.BACSecurityTAN</SecurityProgID>
    <SecurityMediaID></SecurityMediaID>
    <SecurityMediaType>4</SecurityMediaType>
    <CustomerSystemStatus>1</CustomerSystemStatus>
    <BankName>DataDesign DemoBank</BankName>
    <BankUserID></BankUserID>
    <HBCIVersion>220</HBCIVersion>
    <CommunicationsService>3</CommunicationsService>
    <CommunicationsAddress>http://bank.de</CommunicationsAddress>
    <CommunicationsAddressSuffix>64</CommunicationsAddressSuffix>
  </Customer>
</SyncCustomerWizardRq>
```

Anschließend erfolgen die Pinabfrage und die Synchronisierung in einem Wizard. Wenn die Synchronisation erfolgreich ist, wird der Daten des Customer gespeichert und folgende Antwort zurückgeliefert:

```
<SyncCustomerWizardRs>
  <Customer>
    <ID>280:70000999:35081</ID>
    <UserID>35081</UserID>
    <BankCode>70000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Cust1</Contact>
    <SecurityProgID>DataDesign.BACSecurityTAN</SecurityProgID>
    <SecurityMediaID></SecurityMediaID>
    <SecurityMediaType>4</SecurityMediaType>
    <CustomerSystemStatus>1</CustomerSystemStatus>
    <BankName>DataDesign DemoBank</BankName>
    <BankUserID></BankUserID>
    <HBCIVersion>220</HBCIVersion>
    <CommunicationsService>3</CommunicationsService>
    <CommunicationsAddress>http://bank.de</CommunicationsAddress>
    <CommunicationsAddressSuffix>64</CommunicationsAddressSuffix>
    <CustomerSystemID>9RRPRGBVPGM310ISHBCI02</CustomerSystemID>
    <SignatureID>26</SignatureID>
    <AskForEmptyPassphrase>NO</AskForEmptyPassphrase>
    <UPD>484955...</UPD>
    <BPD>48494250...</BPD>
  </Customer>
```

```
<Status>
  <StatusCode>0</StatusCode>
  <StatusDesc>OK</StatusDesc>
</Status>
</SyncCustomerWizardRs>
```

4.11 ChangePinRq

Damit kann die PIN eines Kontaktes verändert werden. Diese Funktion wird von BACContact.ChangePin aufgerufen.

```
<ChangePinRq>
  <Customer>
    <ID>280:70000999:35081</ID>
    <UserID>35081</UserID>
    <BankCode>70000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>TestHKTAN</Contact>
    <SecurityProgID>DataDesign.BACSecurityTAN</SecurityProgID>
    <SecurityMediaID></SecurityMediaID>
    <SecurityMediaType>4</SecurityMediaType>
    <CustomerSystemStatus>1</CustomerSystemStatus>
    <BankName>DataDesign DemoBank</BankName>
    <BankUserID></BankUserID>
    <HBCIVersion>220</HBCIVersion>
    <CommunicationsService>3</CommunicationsService>
    <CommunicationsAddress>http://bank.de</CommunicationsAddress>
    <CommunicationsAddressSuffix>64</CommunicationsAddressSuffix>
  </Customer>
</ ChangePinRq >
```

Falls die PIN erfolgreich geändert wurde:

```
<ChangePinRs>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</ChangePinRs>
```

Beim ChangePinRq wird die bisherige PIN für das Sicherheitsmedium benötigt. Dies wird mit dem Aufruf <QueryPinRq> versucht, dort kann die Applikation eine möglicherweise bereits hinterlegte PIN übergeben. An dieser Stelle kann bereits die neue PIN übergeben werden, damit der Anwender damit nicht noch einmal im Assistenten danach gefragt wird und damit die Applikation die neue PIN zu dem Kontakt hinterlegen kann.

4.12 LoadStringTableRq

Lädt eine Stringtabelle aus Texten die innerhalb der neuen Dialoge verwendet werden. Damit wird es möglich alle Ausgaben zu lokalisieren, ohne dass die Ressourcen der DDBAC verändert werden. In den jeweiligen Dialogen wird beschrieben, welche Texte an welchen Stellen erscheinen.

```
<LoadStringTableRq></LoadStringTableRq>
```

Und als Antwort wird eine Stringtabelle erwartet:

```
<LoadStringTableRs>  
  <StringTable>  
    <DialogTitleErrorReport>HBCI+ Fehlerbericht</DialogTitleErrorReport>  
    <DialogTitleNewContact>HBCI+ Sicherheit Only</DialogTitleNewContact>  
    <DialogHeaderNewContact>Einrichten eines... </DialogHeaderNewContact>  
    <DialogFinalNewContact>Das Einrichten eines...</DialogFinalNewContact>  
  </StringTable>  
</LoadStringTableRq>
```

Die DDBAC enthält intern ebenfalls eine Stringtabelle, welche immer geladen wird. Deshalb muss die zurückgelieferte StringTabelle auch nicht vollständig sein, es genügt, wenn die Strings enthalten sind, die tatsächlich verändert werden sollen. Werden Strings nicht in dieser Tabelle gefunden, werden die Texte aus der Defaulttabelle verwendet.

4.13 OpenTanDialogRq

Dieser Request wird für alle Kontakte ausgeführt die ein Zwei-Schritt-TAN Verfahren verwenden. In diesem Fall kann die TAN nicht vor dem Auftragübergeben werden, sondern die TAN muss nach der Auftragseinreichung passend zu einer Challenge (oder einer anderen Methode) eingegeben werden.

Ob ein Kontakt ein Zwei-Schritt-TAN Verfahren verwendet und deshalb keine TAN vorher übergeben werden kann, ist über das Feld Contact["ITanSupported"] abfragbar. Falls eine "1" zurückgeliefert wird, handelt es sich um einen Kontakt, bei welchem ein Zwei-Schritt-Verfahren ausgewählt wird.

```
<OpenTanDialogRq>
  <Bankparameter>
    <EinSchrittErlaubt>0</EinSchrittErlaubt>
    <MehrereTANpflichtigeGVErlaubt>0</MehrereTANpflichtigeGVErlaubt>
    <HashwertVerfahren>1</HashwertVerfahren>
    <BankenSignatur>0</BankenSignatur>
    <Sicherheitsfunktion>990</Sicherheitsfunktion>
  </Bankparameter>
  <Verfahrensparameter>
    <TANProzess>4</TANProzess>
    <TANVerfahrenID>1</TANVerfahrenID>
    <TANVerfahrenName>Indiziertes TAN-Verfahren</TANVerfahrenName>
    <TANLen>30</TANLen>
    <TANFormat>1</TANFormat>
    <MaxLen>30</MaxLen>
    <RueckgabewertText>TAN-Nummer</RueckgabewertText>
    <AnzahlTANListen>1</AnzahlTANListen>
    <MehrfachTANerlaubt>0</MehrfachTANerlaubt>
    <ZeitversetzteTANerlaubt>0</ZeitversetzteTANerlaubt>
  </Verfahrensparameter>
  <ITAN>
    <TANProcess>1</TANProcess>
    <Hashwert>3132333435363738393031323334353637383930</Hashwert>
    <AuftragsReferenz>Ref</AuftragsReferenz>
    <Challenge>Die TAN wurde an ihr Handy 1234567 verschickt</Challenge>
    <ChallengeGueltigkeitDatum>16.08.2005</ChallengeGueltigkeitDatum>
    <ChallengeGueltigkeitZeit>235959</ChallengeGueltigkeitZeit>
    <TANListNr>69</TANListNr>
  </ITAN>
  <Kontoinformation>
    <Kundenid>1234</Kundenid>
    <Kontowaehrung>EUR</Kontowaehrung>
    <NameEins>Name1</NameEins>
    <NameZwei>Name2</NameZwei>
    <Kontoproduktbezeichnung>Girokonto</Kontoproduktbezeichnung>
    <Kontonummer>12345678</Kontonummer>
  </Kontoinformation>
  <Customer>
    <ID>280:70000999:3508</ID>
    <UserID>3508</UserID>
    <BankCode>70000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>TestHKTAN</Contact>
```

```
<SecurityProgID>DataDesign.BACSecurityTAN</SecurityProgID>
<SecurityMediaID></SecurityMediaID>

<SecurityMediaType>4</SecurityMediaType>
<CustomerSystemStatus>1</CustomerSystemStatus>
<BankName>DataDesign DemoBank</BankName>
<CustomerID></CustomerID>
<BankUserID></BankUserID>
<HBCIVersion>220</HBCIVersion>
<CommunicationsService>3</CommunicationsService>
<CommunicationsAddress>http://server</CommunicationsAddress>
<CommunicationsAddressSuffix>64</CommunicationsAddressSuffix>
<CustomerSystemID>JRUIJQPVKSEM3HNMSHBCI02</CustomerSystemID>
<SignatureID>355</SignatureID>

</Customer>
</OpenTanDialogRq>
```

In diesem Request werden möglichst alle notwendigen Daten mitgeliefert um einen Dialog darzustellen, in welchem der Anwender die TAN eingeben kann oder die (indizierte) TAN aus einer vorher eingegebenen Liste auszuwählen.

Als Antwort auf diesen Request wird eine gültige TAN erwartet:

```
<OpenTanDialogRs>
  <TAN>2111</TAN>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</OpenTanDialogRs>
```

Oder, falls der Anwender keine TAN eingibt und den Dialog beendet, einen entsprechenden Hinweise im Statuscode. Hier ist zu beachten, dass es bankseitig einen "TAN abgefragt aber keine TAN empfangen" Zähler gibt. Wenn zu viele Anfragen nicht beantwortet werden, kann das Konto gesperrt werden.

Die einzelnen Abschnitte im Request werden wie folgt belegt:

Bankparameter (Optional)

Darin sind die Felder des HITANS enthalten, die für alle TAN Verfahren gelten.

Verfahrensparameter (Optional)

Beim Anmelden kann der Anwender möglicherweise aus mehreren TAN Verfahren auswählen, in diesem Abschnitt sind die Werte für das ausgewählte TAN Verfahren aus dem HITANS enthalten. Darin findet man die Bezeichnung für die Challenge und andere Texte, die sich nicht ändern.

ITAN (Optional)

Darin sind alle Felder enthalten, die im HITAN Segment von der Bank zurückgeliefert wurden. Darin ist die Challenge enthalten, die nur für diesen Auftrag gilt.

Kontoinformation (Optional)

Enthält die Werte aus den UPDs des jeweiligen Kontakts passend zur CustomerID mit welcher der Dialog geführt wird.

Customer

Darin sind die Felder des Customers enthalten, für den diese TAN Abfrage gesendet wird.

Die Abschnitte Bankparameter, Verfahrensparameter und ITAN sind optional. Wenn diese fehlen, wird eine TAN für das Ein-Schritt-Verfahren erwartet. (Für DDBAC geplant, in DDBAC.Net realisiert).

Die Standardimplementierung lädt zur Eingabe der TAN einen Dialog mit folgendem Request:

```
<CreateDialogRq Name="dlgEnterTan">  
  <OpenTanDialogRq>... </OpenTanDialogRq>  
</CreateDialogRq>
```

Dabei wird der OpenTanDialogRq zur Information mitgegeben, falls die Dialogressource in Abhängigkeit vom TAN Verfahren abhängig sein soll. Die Standardimplementierung verwendet den dlgEnterTan welcher im Kapitel XAML Dialoge näher beschrieben ist.

4.14 QueryPinRq

Immer wenn eine Pin oder Passphrase benötigt wird und keine Pin übergeben wurde, sendet die DDBAC diesen Request. Darin sind die Daten des Kontakts enthalten, für den eine Pin angefordert wird. Wenn keine Pin zurückgeliefert wird, erscheint ein Fenster in welchem nach der Pin gefragt wird.

Dieser Request erfolgt nicht für Kontakte mit Chipkarten wenn eine Eingabe der Pin am Kartenleser gewünscht wird. In diesem Fall erfolgt immer in Hinweis, dass die Pin am Kartenleser eingegeben werden muss.

```
<QueryPinRq>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    .....
  </Customer>
</QueryPinRq>
```

In der Antwort darauf muss die Pin im Tag <PIN></PIN> zurückgeliefert werden:

```
<QueryPinRs>
  <PIN>123456</PIN>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</QueryPinRs>
```

Diese Implementierung macht die Übergabe der Pin in den Feldern des BACCcustomers überflüssig, diese wird jedoch noch aus Kompatibilitätsgründen noch beibehalten.

Die Standardimplementierung liefert nie eine Pin zurück.

Optional kann bei der PIN-Änderung zusätzlich das Tag <NewPIN> zurückgegeben werden. Diese PIN wird dann als neue PIN verwendet. In allen anderen Fälle wird das Tag nicht beachtet.

4.15 RequestTANListRq

Mit diesem Request kann für einen Kontakt eine neue TAN Liste angefordert werden. Dazu muss es sich um einen Kontakt handeln, der das Pin/Tan Verfahren unterstützt.

Es wird je nach Kontakt entweder ein HKTLA oder ein DKTLA gesendet. Je nach Kreditinstitut wird hier eine TAN Listenummer benötigt.

Siehe auch `BACContact.RequestTANList` und `CanRequestTANList`.

```
<RequestTANListRq>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    . . . . .
  </Customer>
</RequestTANListRq>
```

```
<RequestTANListRs>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    . . . . .
  </Customer>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</RequestTANListRs>
```

4.16 ActivateTANListRq

Mit diesem Request kann für einen Kontakt eine neue TAN Liste aktiviert werden. Dazu muss es sich um einen Kontakt handeln, der das Pin/Tan Verfahren unterstützt.

Es wird je nach Kontakt entweder ein HKTLF oder ein DKTLF gesendet. Je nach Kreditinstitut wird hier eine TAN Listenummer benötigt, eine TAN aus der alten Liste und eine TAN aus der neuen Liste, wobei hier im Zwei-Schritt-Verfahren eine Eingabe einer indizierten TAN notwendig sein kann, also ein OpenTanDialogRq auftritt in welchem der Anwender eine TAN aus der neuen Liste eingeben muss.

Siehe auch BACContact.ActivateTANList und CanActivateTANList.

```
<ActivateTANListRq>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    .....
  </Customer>
</ActivateTANListRq>
```

```
<ActivateTANListRs>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    .....
  </Customer>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</ActivateTANListRs>
```

4.17 LockTANListRq

Mit diesem Request kann für einen Kontakt eine neue TAN Liste gesperrt werden. Dazu muss es sich um einen Kontakt handeln, der das Pin/Tan Verfahren unterstützt.

Es wird je nach Kontakt entweder ein HKTSP oder ein DKTSP gesendet. Je nach Kreditinstitut wird hier eine TAN Listenummer benötigt.

Siehe auch BACContact.LockTANList und CanLockTANList.

```
<LockTANListRq>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    .....
  </Customer>
</LockTANListRq>
```

```
<LockTANListRs>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    .....
  </Customer>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</LockTANListRs>
```

4.18 ShowTANListRq

Mit diesem Request kann für einen Kontakt die Liste bereits verbrauchter TAN Nummer anzeigen. Dazu muss es sich um einen Kontakt handeln, der das Pin/Tan Verfahren unterstützt.

Es wird je nach Kontakt entweder ein HKTAZ oder ein DKTAZ gesendet. Siehe auch BACContact.ShowTANList und CanShowTANList.

```
<ShowTANListRq>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    .....
  </Customer>
</ShowTANListRq>
```

```
<ShowTANListRs>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    .....
  </Customer>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</ShowTANListRs>
```

4.19 LockPINRq

Mit diesem Request kann für einen Kontakt die PIN gesperrt werden. Dazu muss es sich um einen Kontakt handeln, der das Pin/Tan Verfahren unterstützt.

Es wird je nach Kontakt entweder ein HKPSP oder ein DKPSP gesendet.
Siehe auch BACContact.LockPin und CanLockPin.

```
<LockPINRq>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    .....
  </Customer>
</LockPINRq>
```

```
<LockPINRs>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    .....
  </Customer>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</LockPINRs>
```

4.20 UnlockPINRq

Mit diesem Request kann für einen Kontakt die PIN entsperrt werden. Dazu muss es sich um einen Kontakt handeln, der das Pin/Tan Verfahren unterstützt.

Es wird je nach Kontakt entweder ein HKPSA oder ein DKPSA gesendet, in der Regel wird eine TAN benötigt.

Siehe auch BACContact.UnlockPin und CanUnlockPin.

```
<UnlockPINRq>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    . . . . .
  </Customer>
</UnlockPINRq>
```

```
<UnlockPINRs>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    . . . . .
  </Customer>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</UnlockPINRs>
```

4.21 ChangeTANMethodRq

Mit diesem Request kann für einen Kontakt das Zwei-Schritt-TAN-Verfahren ausgewählt werden. Dazu muss es sich um einen Kontakt handeln, der mehr als ein Zwei-Schritt-Tan Verfahren anbietet.

Nach der Umstellung wird versucht mit dem ausgewählten Verfahren eine Synchronisation durchzuführen. Sollte das erfolgreich sein, wurde das TAN Verfahren erfolgreich umgestellt und der Kontakt gespeichert.

Siehe auch BACContact.ChangeTANMethod und CanChangeTANMethod.

```
<ChangeTANMethodRq>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    .....
  </Customer>
</ChangeTANMethodRq>
```

```
<ChangeTANMethodRs>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    .....
  </Customer>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</ChangeTANMethodRs>
```

4.22 GenerateIniLetterRq

Mit diesem Request kann für einen Kontakt der Ini-Brief angezeigt werden. Dazu muss es sich um einen Kontakt handeln, der das RDH Verfahren unterstützt. (Schlüsseldiskette oder G+D Chipkarte)

Siehe auch BACContact.GenerateIniLetter und CanGenerateIniLetter

```
<GenerateIniLetterRq>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    .....
  </Customer>
</GenerateIniLetterRq>
```

```
<GenerateIniLetterRs>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    .....
  </Customer>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</GenerateIniLetterRs>
```

4.23 ChangeConnectionRq

Mit diesem Request kann für einen Kontakt die Verbindung zum Bankserver bearbeitet werden.

Siehe auch BACContact.ChangeConnection

```
<ChangeConnectionRq>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    .....
  </Customer>
</ChangeConnectionRq>
```

Im Tag <Customer> sind anschließend die Verbindungsdaten enthalten, zusätzlich wird der Kontakt gespeichert, wenn er bereits existiert hat.

```
<ChangeConnectionRs>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    .....
  </Customer>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</ChangeConnectionRs>
```

4.24 EditContactRq

Mit diesem Request können für einen Kontakt die Konten (UPD) bearbeitet werden. Der Anwender kann Konten hinzufügen, ändern und löschen.

Siehe auch BACContact.EditContact

```
<EditContactRq>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    . . . . .
  </Customer>
</EditContactRq>
```

Im Tag <Customer/UPD> sind anschließend die Konten enthalten, zusätzlich wird der Kontakt gespeichert, wenn er bereits existiert hat.

```
<EditContactRs>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    . . . . .
  </Customer>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</EditContactRs>
```

4.25 LockKeysRq

Mit diesem Request können für einen Kontakt die Schlüssel gesperrt werden.

Siehe auch BACContact.LockKeys und BACContact.CanLockKeys

```
<LockKeysRq>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    .....
  </Customer>
</LockKeysRq>
```

Der Kontakt wird gelöscht, wenn er bereits existiert hat.

```
<LockKeysRs>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</LockKeysRs>
```

4.26 ChangeUserIdRq

Mit diesem Request kann für einen Kontakt die UserID, CustomerID und der Kontaktname umgestellt werden. Nach der Auswahl der neuen HBCI Version wird eine Synchronisation durchgeführt.

Siehe auch BACContact. ChangeUserId

```
< ChangeUserIdRq>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    .....
  </Customer>
</ ChangeUserIdRq>
```

Der Kontakt enthält anschließend die gewählte HBCI Version.

```
< ChangeUserIds>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    .....
  </Customer>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</ ChangeUserIds>
```

4.27 ChangeHbciVersionRq

Mit diesem Request kann für einen Kontakt die HBCI Version umgestellt werden. Nach der Auswahl der neuen HBCI Version wird eine Synchronisation durchgeführt.

Siehe auch BACContact. ChangeHbciVersion

```
<ChangeHbciVersionRq>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    .....
  </Customer>
</ChangeHbciVersionRq>
```

Der Kontakt enthält anschließend die gewählte HBCI Version.

```
<ChangeHbciVersionRs>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    <HbciVersion>220</HbciVersion>
    .....
  </Customer>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</ChangeHbciVersionRs>
```

4.28 FilelistRq

Mit diesem Request wird eine Liste aller DDBAC Komponenten und zusätzlich einiger ausgewählter Systemkomponenten erstellt. Diese Liste kann jederzeit abgerufen werden um den Status der DDBAC zu überprüfen.

Für die DDBAC Komponenten wird die Signatur der DataDesign AG überprüft. Die Komponenten müssen digital signiert sein und es muss sich um eine Signatur der DataDesign AG handeln. Nur in diesem Fall wird der Tag <Signatur> OK enthalten. Dieser Aufruf kann dazu verwendet werden um die Integrität der DDBAC zu überprüfen.

Der Aufruf kann ja nach System einige Sekunden benötigen, da die Signaturprüfung der DLLs zeitaufwendig ist.

```
<FilelistRq></FilelistRq>
```

```
<FilelistRs>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
  <DDBACFiles>
    <File>
      <Version>4, 0, 13, 0</Version>
      <Description>DataDesign DDBAC Hom... Administrator</Description>
      <Signatur>OK</Signatur>
      <Filepath>C:\WINDOWS\system32\DDBACCPL.CPL</Filepath>
      <Modul>DDBACCPL</Modul>
    </File>
    <File>
      <Version>4, 0, 13, 0</Version>
      <Description>DataDesign DDBAC Card Terminal Manager</Description>
      <Signatur>OK</Signatur>
      <Filepath>C:\WINDOWS\system32\DDBACCTM.CPL</Filepath>
      <Modul>DDBACCTM</Modul>
    </File>
    ...
  </DDBACFiles>
  <SystemFiles>
    <File>
      <Version>8.00.7002.0</Version>
      <Description>XML OM for Win32</Description>
      <Filepath>C:\WINDOWS\system32\MSXML.DLL</Filepath>
      <Modul>MSXML</Modul>
    </File>
    ...
</FilelistRs>
```

4.29 OpenAboutDialogRq

Mit diesem Request wird eine Liste aller DDBAC Komponenten und zusätzlich einiger ausgewählter Systemkomponenten in einem Dialog angezeigt.

Dabei wird im Wesentlichen der Inhalte aus dem FilelistRq in einer Liste angezeigt. Auf dieser Dialogseite kann zusätzlich das Logging in die HBCILog.txt eingestellt werden.

Der Aufruf kann je nach System einige Sekunden benötigen, da die Signaturprüfung der DLLs zeitaufwendig ist.

```
<OpenAboutDialogRq></ OpenAboutDialogRq>
```

```
<OpenAboutDialogRs>  
  <Status>  
    <StatusCode>0</StatusCode>  
    <StatusDesc>OK</StatusDesc>  
  </Status>  
</OpenAboutDialogRs>
```

4.30 ChangeKeysRq

Mit diesem Request kann für einen Kontakt der Wizard zur Schlüsseländerung gestartet werden. Ob eine Schlüsseländerung möglich ist kann derzeit über das Feld „CanChangeKey“ in BACContact.Fields abgefragt werden. Falls eine Schlüsseländerung möglich ist, wird „1“ zurückgeliefert. Eine Schlüsseländerung kann auch zum Sicherheitsprofilwechsel verwendet werden, wenn der Anwender bei der Auswahl des Schlüsselformats ein anderes verfügbares Format auswählt.

```
<ChangeKeysRq>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    .....
  </Customer>
</ChangeKeysRq>
```

Der Kontakt enthält anschließend die geänderten Daten.

```
<ChangeKeysRs>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    .....
  </Customer>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</ChangeHbciVersionRs>
```

4.31 UpgradeKeysRq

Mit diesem Request kann für einen Kontakt ein Sicherheitsprofilwechsel gestartet werden. Dieser Wechsel ist derzeit nur für Schlüsseldisketten im RDH-1 Format auf RDH-2 möglich. Ob ein Sicherheitsprofilwechsel möglich ist kann derzeit über das Feld „CanUpgradeKey“ in BACContact.Fields abgefragt werden. Falls ein Sicherheitsprofilwechsel möglich ist, wird „1“ zurückgeliefert.

Der Assistent für den Sicherheitsprofilwechsel legt eine neue Sicherheitsdatei im RDH-2 Format an und synchronisiert den Kontakt. Anschließend wird der neue Kontakt zurückgeliefert.

Beim Neuanlegen der Schlüsseldatei wird nach einem neuen Passwort gefragt, das der Anwender im Verlauf des Assistenten eingeben muss. Da die DDBAC solche Passwörter möglichst kurz im Speicher hält und auch nicht an die Applikation zurückgibt, muss der Anwender unter Umständen darauf hingewiesen werden, dass das in der Applikation hinterlegte Passwort ebenfalls geändert werden muss.

Dies gilt insbesondere beim Wechsel von RDH-1 auf RDH-2 da bei RDH-2 höhere Anforderungen an das zu verwendende Passwort gestellt werden.

```
<UpgradeKeysRq>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    . . . . .
  </Customer>
</UpgradeKeysRq>
```

Der Kontakt enthält anschließend die gewählte HBCI Version.

```
<UpgradeKeysRs>
  <Customer>
    <UserID>123456</UserID>
    <BankCode>7000999</BankCode>
    <CountryCode>280</CountryCode>
    <Contact>Kontaktname</Contact>
    . . . . .
  </Customer>
  <Status>
    <StatusCode>0</StatusCode>
    <StatusDesc>OK</StatusDesc>
  </Status>
</UpgradeKeysRs>
```

5 XAML Dialog Format

Alle Dialogressourcen liegen in XAML Dateien vor, werden derzeit aber nicht mittels Windowsfunktionen dargestellt sondern werden interpretiert und dann mit Standardcontrols dargestellt. Das bedeutet, dass nur ein kleiner Teil der XAML Features verfügbar ist, die jedoch für die benötigte Funktionalität ausreichen sollten.

5.1 Canvas

XAML Dialoge sind reine XML Dateien die von jedem XML Parser interpretiert werden können. Die DDBAC verwendet MSXML um diese Dateien zu öffnen. Die Dialoge müssen wie folgt als Canvas gespeichert werden und die angegebenen Namespaces enthalten:

```
<Canvas Name="ROOT" Width="440" Height="314"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
    ....
</Canvas>
```

Alle Elemente innerhalb dieses Canvas werden von der DDBAC ausgewertet.

Attribut	Bedeutung	Defaultwert
Width	Breite des Dialogs	
Height	Höhe des Dialogs	
Font	Name des Fonts für den gesamten Dialog	Arial
FontSize	Größe des Fonts	10
Title	Überschrift des Dialogs	
HwndParent	Optionales Handle des gewünschten Parentfensters	
DialogUnits	0 = Positionen in Pixel 1 = Postionen in DialogUnits	0
AlwaysOnTop	1 = Dialog wird "AlwaysOnTop" angezeigt.	0
Background	Hintergrundfarbe (RGB in Form von "0044FF")	

Der Fontname und die FontSize werden an CreateFont übergeben, wobei die Fontgröße in logischen Einheiten erwartet wird und dann in DeviceUnits mit folgender Formel umgerechnet wird:

```
lf.lfHeight = -MulDiv(lf.lfHeight, GetDeviceCaps(GetDC()->m_hDC, LOGPIXELSY), 72);
```

Das Canvas definiert den Rahmen in welchen die folgenden Controls eingefügt werden. Die TAB Reihenfolge entspricht der Reihenfolge der Controls im XML.

5.2 Button

Einfache Taste mit folgenden Attributen:

Attribut	Bedeutung	Defaultwert
Name	Name des Controls	
Canvas.Left	Linker Abstand	
Canvas.Top	Oberer Abstand	
Width	Breite des Controls	
Height	Höhe des Controls	
Font	Name des Fonts für den gesamten Dialog	Arial
FontSize	Größe des Fonts	10
Content	Text des Controls	
Foreground	Textfarbe in RGB	
FontFamily	Art des Fonts z.B. Tahoma, MS Sans Serif	
FontWeight	"Normal" oder "Bold" für Fett	Normal
FontSize	Größe des Fonts	

Der Button mit dem Namen "btnHelp" wird gesondert behandelt und bekommt immer das vordefinierte Bitmap für die Hilfefunktion.

5.3 TextBox

Standard Eingabefeld für Text.

Attribut	Bedeutung	Defaultwert
Name	Name des Controls	
Canvas.Left	Linker Abstand	
Canvas.Top	Oberer Abstand	
Width	Breite des Controls	
Height	Höhe des Controls	
Font	Name des Fonts für den gesamten Dialog	Arial
FontSize	Größe des Fonts	10
Text	Text des Controls	
TextWrapping	Falls "Wrap" wird ein Multilinecontrol erzeugt	
IsReadOnly	Falls "True" wird Readonly gesetzt	
Foreground	Textfarbe in RGB	
Background	Hintergrundfarbe (RGB in Form von "0044FF")	
FontFamily	Art des Fonts z.B. Tahoma, MS Sans Serif	
FontWeight	"Normal" oder "Bold" für Fett	Normal
FontSize	Größe des Fonts	

Alle Eingabefelder die "txtPassword" heißen werden als Passwortfeld deklariert.

5.4 TextBlock

Mehrzeiliges Feld zur Anzeige von Text. Der Inhalt wird dabei über mehrere Zeilen umgebrochen, wenn der Inhalt zu lang ist.

Attribut	Bedeutung	Defaultwert
Name	Name des Controls	
Canvas.Left	Linker Abstand	

Canvas.Top	Oberer Abstand	
Width	Breite des Controls	
Height	Höhe des Controls	
Font	Name des Fonts für den gesamten Dialog	Arial
FontSize	Größe des Fonts	10
Text	Text des Controls	
TextAlignment	Falls "Right" rechtsbündig Falls "Center" zentriert, sonst linksbündig	Linksbündig
Foreground	Textfarbe in RGB	
Background	Hintergrundfarbe (RGB in Form von "0044FF")	Transparent
FontFamily	Art des Fonts z.B. Tahoma, MS Sans Serif	
FontWeight	"Normal" oder "Bold" für Fett	Normal
FontSize	Größe des Fonts	

5.5 Image

Darstellung von Bildern.

Attribut	Bedeutung	Defaultwert
Name	Name des Controls	
Canvas.Left	Linker Abstand	
Canvas.Top	Oberer Abstand	
Width	Breite des Controls	
Height	Höhe des Controls	
Source	Dateiname (siehe unten)	
Background	Hintergrundfarbe (RGB in Form von "0044FF")	Transparent

Also Source wird immer ein kompletter Dateiname erwartet, welcher mit file:// beginnen kann. Die Bilder werden mit der Win32 Funktion "OleLoadPicture" geladen, von daher werden alle Formate unterstützt die von dieser Win32 Funktion unterstützt werden. Die wesentlichen Formate wie .Gif und .Jpg sind unter allen Windowsversionen verfügbar.

Zusätzlich werden animierte Gif Dateien unterstützt. Transparente Images zeigen den Dialoghintergrund, außer wenn expliziert eine Hintergrundfarbe "Background" angegeben wird.

Ein Image mit dem Namen "SecurePad" wird immer durch das SecurePad ersetzt und interagiert in den dafür vorgesehen Dialog mit dem "txtPassword".

5.6 ListBox

Eine ListBox mit mehreren Elementen.

Attribut	Bedeutung	Defaultwert
Name	Name des Controls	
Canvas.Left	Linker Abstand	
Canvas.Top	Oberer Abstand	
Width	Breite des Controls	
Height	Höhe des Controls	

Kann derzeit nur für die Liste von Kontakten auf der Chipkarte verwendet werden, da der jeweilige Dialog auf diese Liste abgestimmt ist.

5.7 ComboBox

Eine AuswahlBox mit mehreren Elementen.

Attribut	Bedeutung	Defaultwert
Name	Name des Controls	
Canvas.Left	Linker Abstand	
Canvas.Top	Oberer Abstand	
Width	Breite des Controls	
Height	Höhe des Controls	

Kann derzeit nur für die Liste von TAN Verfahren bei der Authentifizierung in EnterPin2 verwendet werden, da der jeweilige Dialog auf diese Liste abgestimmt ist.

5.8 CheckBox

Einfache Auswahl einer Ja/Nein Option.

Attribut	Bedeutung	Defaultwert
Name	Name des Controls	
Canvas.Left	Linker Abstand	
Canvas.Top	Oberer Abstand	
Width	Breite des Controls	
Height	Höhe des Controls	
Font	Name des Fonts für den gesamten Dialog	Arial
FontSize	Größe des Fonts	10
Content	Text des Controls	
IsChecked	Falls "True" ist die CheckBos angewählt	False
Foreground	Textfarbe in RGB	
Background	Hintergrundfarbe (RGB in Form von "0044FF")	Transparent
FontFamily	Art des Fonts z.B. Tahoma, MS Sans Serif	
FontWeight	"Normal" oder "Bold" für Fett	Normal
FontSize	Größe des Fonts	

5.9 RadioButton

Auswahl aus mehreren Optionen.

Attribut	Bedeutung	Defaultwert
Name	Name des Controls	
Canvas.Left	Linker Abstand	
Canvas.Top	Oberer Abstand	
Width	Breite des Controls	
Height	Höhe des Controls	
Font	Name des Fonts für den gesamten Dialog	Arial

FontSize	Größe des Fonts	10
Content	Text des Controls	
IsChecked	Falls "True" ist die RadioBos angewählt	False
Foreground	Textfarbe in RGB	
Background	Hintergrundfarbe (RGB in Form von "0044FF")	Transparent
FontFamily	Art des Fonts z.B. Tahoma, MS Sans Serif	
FontWeight	"Normal" oder "Bold" für Fett	Normal
FontSize	Größe des Fonts	

5.10 Label

Einzeiliges Feld zur Anzeige von Text. Der Inhalt wird immer abgeschnitten wenn der Text zu lang ist.

Attribut	Bedeutung	Defaultwert
Name	Name des Controls	
Canvas.Left	Linker Abstand	
Canvas.Top	Oberer Abstand	
Width	Breite des Controls	
Height	Höhe des Controls	
Font	Name des Fonts für den gesamten Dialog	Arial
FontSize	Größe des Fonts	10
Text	Text des Controls	
TextAlignment	Falls "Right" rechtsbündig Falls "Center" zentriert, sonst linksbündig	Linksbündig
Foreground	Textfarbe in RGB	
Background	Hintergrundfarbe (RGB in Form von "0044FF")	Transparent
FontFamily	Art des Fonts z.B. Tahoma, MS Sans Serif	
FontWeight	"Normal" oder "Bold" für Fett	Normal
FontSize	Größe des Fonts	

5.11 Canvas

Ein Control innerhalb eines Dialogs, das einen weiteren Dialog aufnimmt. Dieses Control wird innerhalb des Wizards verwendet um in einem Frame zustandsabhängig andere Dialoge einzublenden. Ist nur in Zusammenhang mit dem Wizardframe sinnvoll und dient nur der Positionierung des Unterdialogs.

Attribut	Bedeutung	Defaultwert
Name	Name des Controls	
Canvas.Left	Linker Abstand	
Canvas.Top	Oberer Abstand	
Width	Breite des Controls	
Height	Höhe des Controls	

6 XAML Dialoge erstellen

Die Dialoge liegen als Textdateien vor und können mit jedem Texteditor bearbeitet werden. Dabei muss beachtet werden, dass die Dateien als UTF-8 codiert gespeichert werden müssen. Notepad.exe kodiert solche Dateien zuweilen mit einer 3 Byte langen Kennung am anfang, was dazu führt, dass die Dateien nicht gelesen werden können.

Mit den Tools aus Windows Vista können die XAML Dateien auch im WYSIWYG Modus bearbeitet werden. Ein weiteres Tool ist Aurora, ein XAML Editor von www.mobiform.de.

Um ihre eigenen Dialoge in die DDBAC einzubinden, ist es notwendig die Callback-Schnittstelle der DDBAC zu implementieren und die CreateDialogRq zu bearbeiten. Hier können sie ihre eigenen XAML Ressourcen zurückliefern und damit die Darstellung der Dialoge beeinflussen. Diese in in ddBACTest beispielhaft implementiert.

Um beispielsweise nur den Font zu ändern, genügt es in allen CreateDialogRq Antworten das Attribut "Font" und "FontSize" hinzuzufügen um in allen Dialogen die Fonts zu ändern. Sie können auch sehr einfach den kompletten Wizard ändern, indem Sie beispielsweise nur den Rahmen außen ändern, welche um alle Wizardseiten herum verwendet wird.

Sie können in den Dialogen jederzeit Textelemente, Grafiken oder zusätzliche Hinweise hinzufügen, die beispielsweise nur für bestimmte Banken wichtig sind. Sie können auch funktionale Elemente weglassen, wenn diese Auswahl nicht erlaubt sein soll. Das einzige was nicht möglich ist, ist funktionale Ergänzungen zu machen. Es ist zwar möglich einen <Button> hinzuzufügen, aber es wird nichts passieren wenn dieser ausgewählt wird, weil keine Funktion hinterlegt ist.

Daher ist es nur möglich das Layout der Dialoge zu beeinflussen, nicht aber deren Funktionsweise. Diese Funktionen sind an die jeweiligen Controls gebunden und werden über den Namen des Controls zugeordnet. So ist es für die Funktionalität unerheblich wo das Feld "txtUserID" steht und welche Farbe und Größe es hat, es ist nur wichtig, dass dieses Feld vorhanden ist.

6.1 Wizards

Wizards (Assistenten) sind alle Dialoge die zwei oder mehr Schritte des Anwenders erfordern um die Aktion durchzuführen. Dabei gibt es immer einen WizardFrame, einen Rahmen der jeweils eine Wizardseite enthält und je nach Aufgabe weitere Wizardseiten anzeigt. Der Rahmen bleibt immer der gleiche, weshalb alle Wizardseiten immer gleich groß sein müssen.

6.1.1 WizardFrame

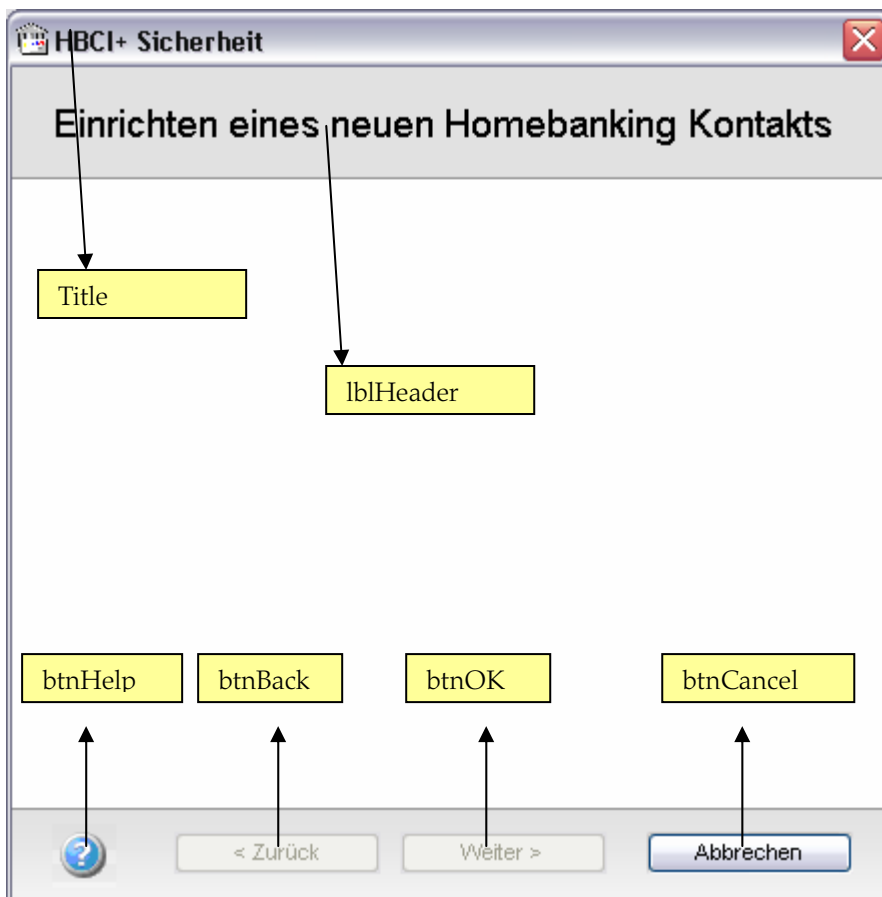
Das ist der Rahmen um alle Wizardseiten. In der Standardimplementierung sieht dieser so aus:

```
<Canvas Name="ROOT" Width="440" Height="417"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
  <Canvas.Background>
    <ImageBrush
ImageSource="file:///D:/Products/DDBAC/4.0/Source/DDBACDLG/BackgroundOB.gif" />
  </Canvas.Background>
  <Button Name="btnBack" InputMethod.IsInputMethodEnabled="False" Width="103"
Height="21" Canvas.Left="81" Canvas.Top="382" Content="&lt; Zurück">
</Button>
```

```

<Button Name="btnOK" InputMethod.IsInputMethodEnabled="False" Width="103"
      Height="21" Canvas.Left="194" Canvas.Top="382" Content="Weiter &gt;">
</Button>
<Canvas Name="WizardPage" Width="440" Height="314" Background="#FFF5F5DC"
      Canvas.Top="56" />
<Label Name="lblHeader" Width="411" Height="30" FontSize="14" Canvas.Left="20"
      Canvas.Top="17" Content="[Einrichten eines neuen Homebanking Kontakts]">
</Label>
<Button Name="btnCancel" InputMethod.IsInputMethodEnabled="False" Width="103"
      Height="21" Canvas.Left="317" Canvas.Top="382" Content="Abbrechen">
</Button>
<Button Name="btnHelp" Width="32" Height="32" Canvas.Left="20" Canvas.Top="378"
      Content="Help">
</Button>
</Canvas>

```



Background: ImageBrush enthält den Standardhintergrund.

Title

Wird mit dem Text "DialogTitleNewContact" aus der Stringtabelle gefüllt. Wobei hier je nach Wizard unterschiedliche Texte verwendet werden.

lblHeader

Wird mit dem Text " DialogHeaderNewContact" aus der Stringtabelle gefüllt. Wobei hier je nach Wizard unterschiedliche Texte verwendet werden.

btnBack

Die Zurücktaste mit der man zur vorherigen Wizardseite zurückgehen kann.

btnOK

Damit wird zur nächsten Seite gewechselt oder der Wizard beendet.

btnCancel

Dient zum Abbrechen des Wizards.

btnHelp

Hilfetaste die zur jeweiligen Wizardseite die Beschreibung aus der Hilfe aufruft.

Canvas WizardPage: Definiert die Position und Größe in welche die Wizardseiten dargestellt werden.

Durch das Ändern des Layouts dieser Seite kann das Aussehen des kompletten Wizards beeinflusst werden. Hier kann beispielsweise das Hintergrundbild oder die Hintergrundfarbe ausgewählt werden. Da alle Wizardseiten soweit wie möglich transparent gehalten sind, wird der hier gewählte Hintergrund auf allen Seiten erscheinen.

Das gleiche gilt für den Zeichensatz und die Schriftgröße. Da alle Controls bis auf wenige Ausnahmen keinen eigenen Font definieren, wird der Font aus diesem Frame übernommen. Damit ist es ohne viel Aufwand möglich für alle Seiten den Font auch "Tahoma" oder "MS Sans Serif" umzustellen indem dieser einfach auf dieser Wizardframe Seite eingestellt werden.

6.1.2 EnterBlz

Dieser Dialog dient zur Eingabe der Bankleitzahl.

The screenshot shows a dialog window titled "HBCI+ Sicherheit" with a close button in the top right corner. The main heading is "Einrichten eines neuen Homebanking Kontakts". Below this, the instruction reads: "Bitte geben Sie die achtstellige Bankleitzahl Ihres Kreditinstituts ein." There are two text input fields: "Bankleitzahl (BLZ):" and "Name des Kreditinstituts:". Below these fields is a checkbox labeled "Zugangsdaten manuell eingeben (für Experten)". Three yellow boxes with arrows point to the controls: "txtBlz" points to the BLZ input field, "lblBankname" points to the bank name input field, and "chkManual" points to the checkbox. At the bottom, there is a help icon, a "< Zurück" button, a "Weiter >" button, and an "Abbrechen" button.

txtBlz

In txtBlz wird die Eingabe einer Bankleitzahl erwartet. Sobald diese 8 Stellen lang ist wird über die Bankleitzahleninformation (IBACBlzInfo) der Bankname ermittelt. Sofern dieser gefunden wird, wird der Name in lblBankname übertragen.

Das Feld txtBlz wird mit dem Inhalt des Feldes "BankCode" aus dem Kontakt vorbelegt. Wenn nichts übergeben wird, wird der Inhalt aus der Dialogressource voreingestellt ansonsten bleibt das Feld leer.

Diese Seite kann erst verlassen werden, wenn eine achtstellige, numerische Bankleitzahl eingegeben wurde.

lblBankname (optional)

In diesem Control wird der Bankname passend zu der Bankleitzahl angezeigt, sofern ein Bankname gefunden wird. Kann bis zu 58 Zeichen enthalten und enthält den Namen (Feld 3) der Bankleitzahlendatei der Bundesbank.

chkManual (optional)

Falls dieses Feld ausgewählt wird, erscheinen während der Kontakteinrichtung zusätzliche Seiten in welchen nach der HBCI Version und den Verbindungsdaten gefragt wird.

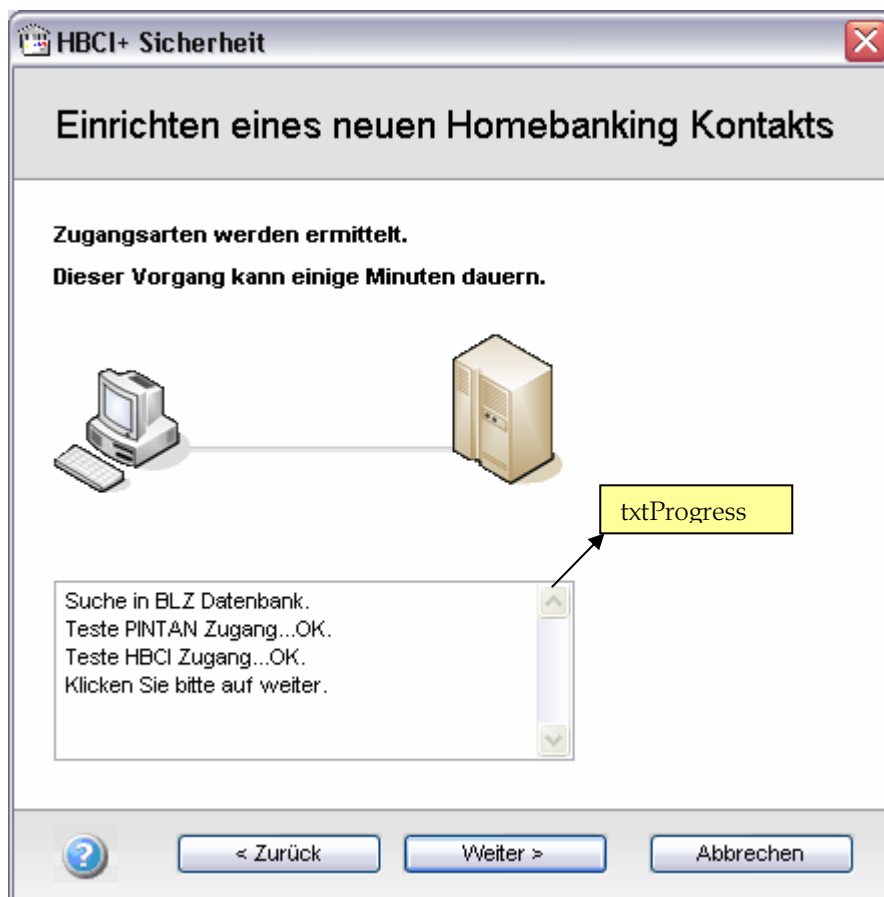
Weiter

Wenn die Bankleitzahl korrekt ist gelangt man anschließend auf die Seite "GetBankInfo".

6.1.3 GetBankInfo

Hier werden die Zugangsdaten zur Bank Online ermittelt. Dazu werden die Daten aus der Bankleitzahleninformation mit einem anonymen Dialog ermittelt. Wenn die Verbindung mit der Bank möglich ist, werden die verfügbaren Zugangsmethoden ermittelt. Falls keine Verbindung mit der Bank hergestellt werden kann, schält der wizard in den Expertenmodus in welchem die Zugangsdaten dann manuell eingetragen werden können.

Die Zugangsdaten zur Bank werden mit dem "GetBankInfoRq" ermittelt, siehe den dazugehörigen Aufruf. An dieser Stelle kann die Applikation zusätzliche oder eigene Informationen über die jeweilige Bank an den Wizard liefert und damit den weiteren Verlauf beeinflussen.



txtProgress (optional)

In diesem Feld wird der Prozessfortschritt durch Textausgaben dargestellt.

Weiter

Während der Verbindung mit der Bank ist die Weiter-Taste gesperrt. Sie wird freigegeben, wenn die Ermittlung der Daten abgeschlossen ist. Abhängig vom Ergebnis gelangt man anschließend auf eine der folgenden Seiten:

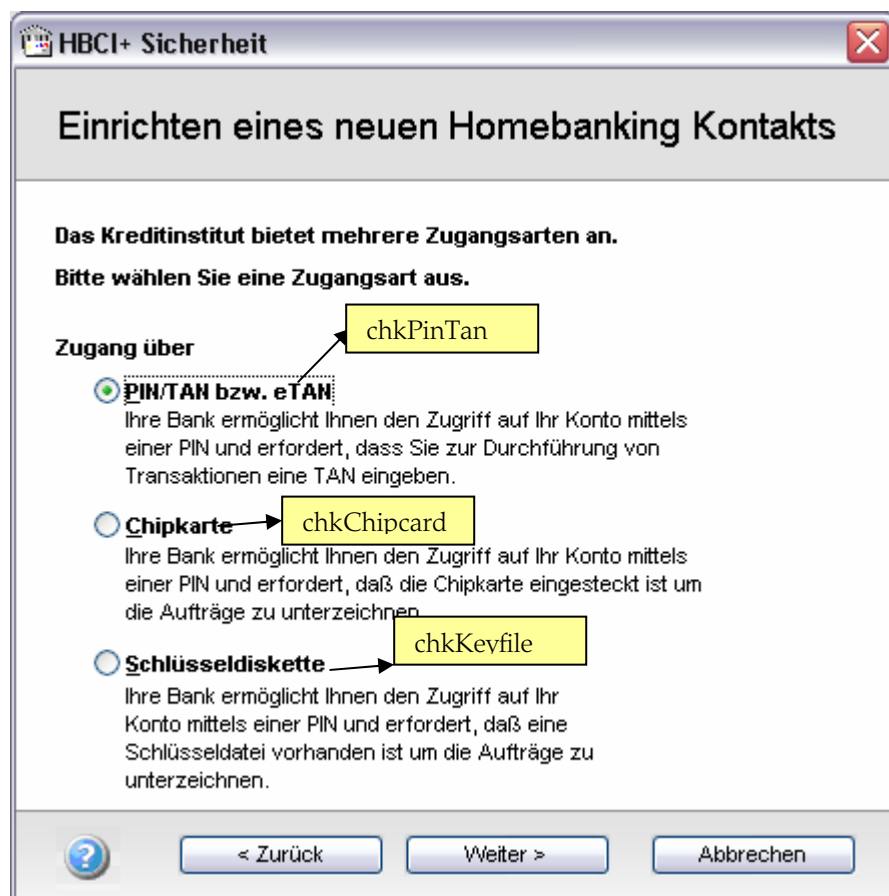
Falls eine Fehler aufgetreten ist, wird auf die Seite "Error" weitergeleitet.

Falls keine oder mehr als eine mögliche Zugangsart gefunden wird zur Seite "SelectConnect" weitergeleitet, auf welcher man die Zugangsart auswählen kann.

Wird genau eine mögliche Zugangsart gefunden, wird abhängig von dieser die nächste Seite ausgewählt. Falls es sich um eine Chipkarte mit DDV handelt ist die nächste Seite "EnterPin2" im Fall von PIN/TAN "EnterID" und ansonsten die Seite zur Auswahl der Zugangsart, auf welcher man zwischen Chipkarte und Schlüsseldiskette wählen kann.

6.1.4 SelectConnect

Dient zur Auswahl des verfügbaren Sicherheitsmediums. Derzeit stehen 3 Verfahren zur Auswahl: Chipkarte, Pin/TAN, Schlüsseldatei. Je nach Kreditinstitut werden verschiedene Verfahren angeboten:



Im Expertenmodus sind grundsätzlich alle Zugangsarten erlaubt. Ansonsten ist es abhängig von den verfügbaren Zugangsarten. Die Felder sind alle optional, das bedeutet, wenn für einen bestimmten Zugang die Option "Schlüsseldiskette" nicht angeboten werden soll, dann kann das Feld vollkommen entfernt werden.

chkPinTan (optional)

Ist verfügbar wenn die Bank Pin/Tan anbietet, ansonsten nicht anwählbar.

chkChipcard (optional)

Ist verfügbar wenn die Bank "RDH" oder "DDV" unterstützt.

chkKeyfile (optional)

Ist nur bei "RDH" verfügbar.

6.1.5 EnterID

Dient zur Eingabe der Benutzerkennung und des Kontaktnamens.

lblText (optional)

Darin wird der Text aus der Stringtabelle "EnterID_Text" gefolgt vom Banknamen angezeigt.

lblUserID (optional)

Je nach Kreditinstitut kann diese Bezeichnung variieren. Wenn im GetBankInfoRq unter <UserIDName> ein Text enthalten ist, wird dieser angezeigt, ansonsten der Inhalt welcher in der Dialogressource eingetragen ist.

txtUserID

Enthält die UserID für den jeweiligen Kontakt. Diese Feld muss ausgefüllt werden.

lblCustomerID (optional)

Die Bezeichnung der CustomerID wird wie bei der UserID aus dem Feld <CustomerIDName> gelesen. Sollte darin "(nicht belegt)" enthalten sein, wird das Feld lblCustomerID und das Feld txtCustomerID versteckt.

txtCustomerID (optional)

Optionales Feld zur Eingabe der CustomerID.

txtContactName

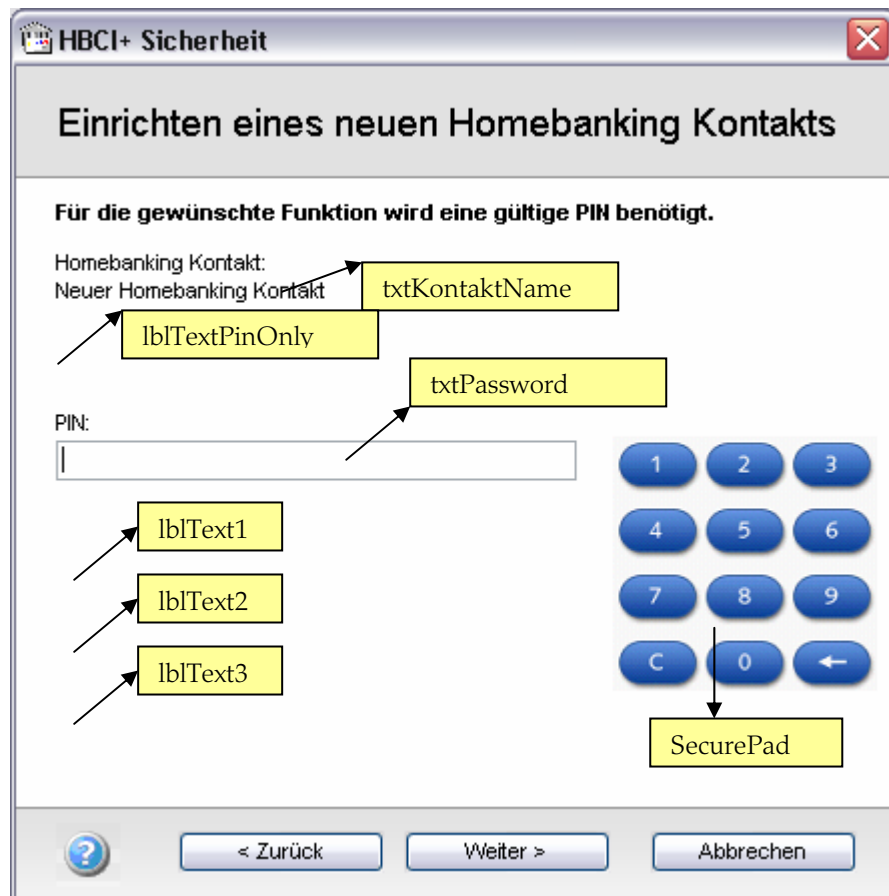
Name des Kontaktes wie er beispielsweise im Homebankingadministrator angezeigt wird. In der Standardkonfiguration muss dieses Feld ausgefüllt werden und es darf noch keinen Kontakt mit gleichem Inhalt geben.

chkDoNotSync (optional)

Nach dem die Benutzerdaten zu einem Kontakt angegeben wurden, muss diese Kontakt mit dem Kreditinstitut synchronisiert werden. Optional kann man diese Synchronisation unterlassen, dies kann jederzeit später mit Kontakt synchronisieren nachgeholt werden. Dieses Feld ist optional, wenn es nicht vorhanden ist, wird davon ausgegangen dass der Kontakt synchronisiert werden soll.

6.1.6 EnterPin2

Dieser Dialog wird zur Eingabe der PIN für alle Zugangsarten verwendet. Je nach Zugangsart gibt es unterschiedliche Ansichten, es handelt sich jedoch immer um die gleiche Ressource EnterPin2.xaml.



Die ist die Eingabe einer PIN für das klassische Ein-Schritt PIN/TAN Verfahren.

Felder die in allen Zugangsarten gleich verwendet werden:

txtKontaktName (optional)

In diesem Feld wird der momentan bearbeitete Kontaktname angezeigt. Falls es sich um einen neuen Kontakt handelt wird der der Text aus der Stingtabelle "EnterPin2_NewContact" angezeigt.

txtPassword

Enthält die einzugebende PIN.

SecurePad (optional)

Das Securepad wird in der xaml Datei nur als Image angegeben. Alle Felder mit dem Namen SecurePad werden durch das grafische Pin-Eingabe Feld ersetzt. Dieses korrespondiert mit txtPassword. Wenn die Tasten auf dem SecurePad gedrückt werden, erscheinen Sterne in txtPassword. Tatsächlich findet aber keine Übertragung der PIN statt, diese wird nur im Speicher gehalten. Damit ist es nicht möglich durch Überwachung der Eingaben über die Tastatur auf die PIN

zu schließen. Da in dem Textfeld die echte PIN nicht enthalten ist, kann diese auch nicht über Spionageprogramme ausgelesen werden.

lblText1, lblText2, lblText3 (optional)

Zusätzliche, optionale Textfelder die mit dem Texten aus der Stringtabelle "EnterPin2_TAN_Text1", "EnterPin2_TAN_Text2", "EnterPin2_TAN_Text3" gefüllt werden. In diesem Fall sind die Texte leer.

lblTextPinOnly (optional)

Dieses Textfeld wird nur angezeigt, wenn es sich um das PIN/TAN Verfahren handelt, in allen anderen Fällen wird es versteckt.

lblChipdrive, cmbChipdrives, txtChipdrives, btnChipdrives, chkUseClass2 (optional)

Diese Felder sind beim EinSchritt PIN/TAN Verfahren nicht sichtbar.

6.1.7 EnterPin2 (Zweischritt TAN Verfahren)

Falls ein Kontakt mit mehreren TAN Verfahren verwendet wird, sieht der EnterPin2 Dialog leicht anders aus:

The screenshot shows a dialog box titled "HBCI+ Sicherheit" with a close button in the top right corner. The main heading is "Synchronisieren eines Homebanking Kontakts". Below this, a message states: "Für die gewünschte Funktion wird eine gültige PIN benötigt." The form contains the following elements:

- Homebanking Kontakt: _J2HBCI_220
- TAN Verfahren: A dropdown menu showing "902 mTAN2". A yellow box containing the text "cmbChipdrives / txtChipdrives" has an arrow pointing to this dropdown.
- PIN: A text input field with five dots.
- A numeric keypad with buttons for digits 1-9, 0, a clear button (C), and a back arrow.
- Navigation buttons at the bottom: a help icon (?), "< Zurück", "Weiter >", and "Abbrechen".

cmbChipdrives

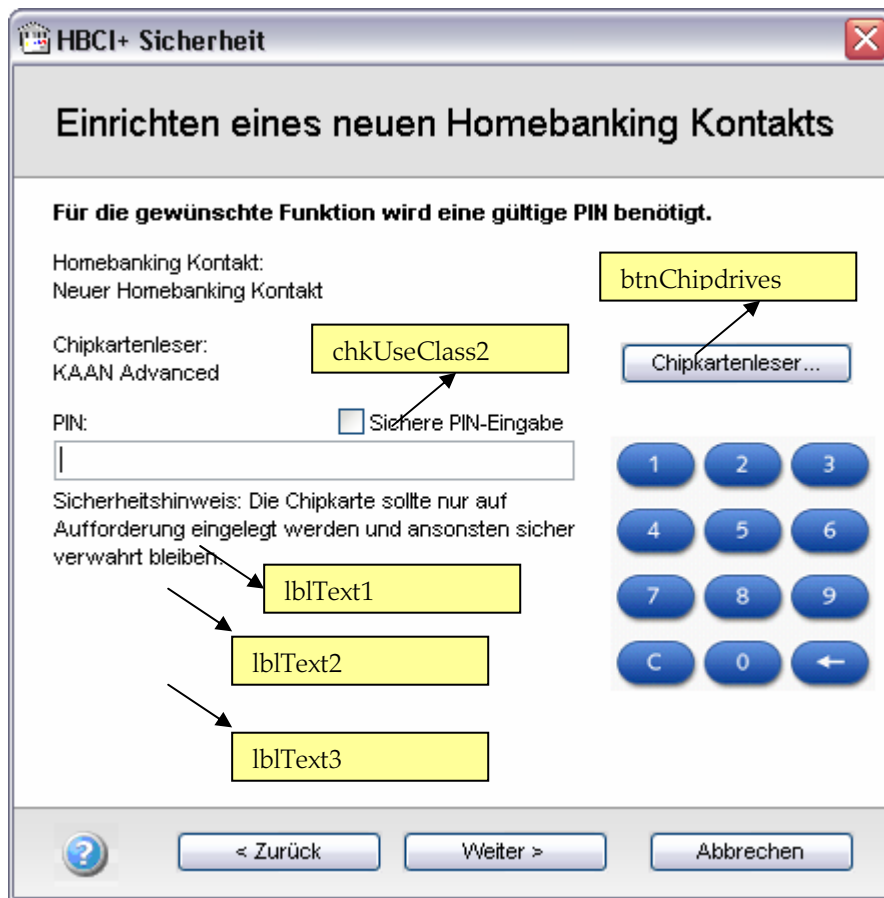
Enthält eine Liste der verfügbaren TAN Verfahren, der Anwender kann das konkret verwendete Verfahren auswählen. Das jeweils zuletzt verwendete Verfahren wird bei der nächsten Anmeldung vorausgewählt.

txtChipdrives

Falls nur ein TAN Verfahren verfügbar ist, wird die Auswahlbox "cmbChipdrives" versteckt und statt dessen Verfahren dieses Textfeld mit dem Namen des TAN Verfahrens angezeigt.

6.1.8 EnterPin2 mit Klasse 1

Wird zur Eingabe einer Pin mit einem Klasse1 Kartenleser verwendet.



btnChipdrives (Optional)

Damit wird das Chipkartenleser Control aus dem Controlpanel aufgerufen, mit welchem man den Kartenleser auswählen oder einrichten kann.

chkUseClass2 (Optional)

Falls ein Klasse 2 Leser angeschlossen ist, kann hier in die Sichere Klasse 2 Eingabe umgeschaltet werden. Anschließend muss die PIN am Lesegerät eingegeben werden. Die letzte Einstellung wird gespeichert.

lblText1 (Optional)

Hier wird für Klasse 1 Leser der Text "EnterPin2_Card1_Text1" aus der Sstringtabelle angezeigt.

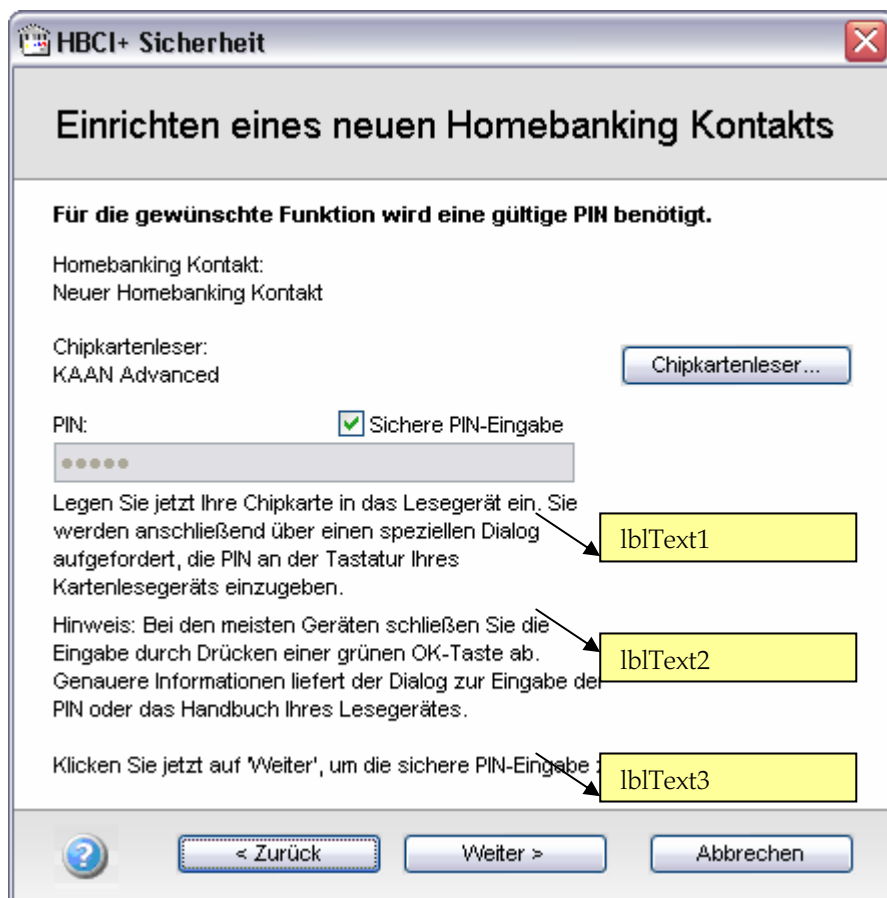
lblText2 (Optional)

Hier wird für Klasse 1 Leser der Text "EnterPin2_Card1_Text2" aus der Sstringtabelle angezeigt.

lblText3 (Optional)

Hier wird für Klasse 1 Leser der Text "EnterPin2_Card1_Text3" aus der Sstringtabelle angezeigt.

6.1.9 EnterPin2 mit Klasse 2



Falls die Pin auf dem Kartenlesegerät eingegeben werden soll (sichere PIN-Eingabe) erscheint der Dialog in dieser Form.

lblText1 (Optional)

Hier wird für Klasse 1 Leser der Text "EnterPin2_Card2_Text1" aus der Sstringtabelle angezeigt.

lblText2 (Optional)

Hier wird für Klasse 1 Leser der Text "EnterPin2_Card2_Text2" aus der Sstringtabelle angezeigt.

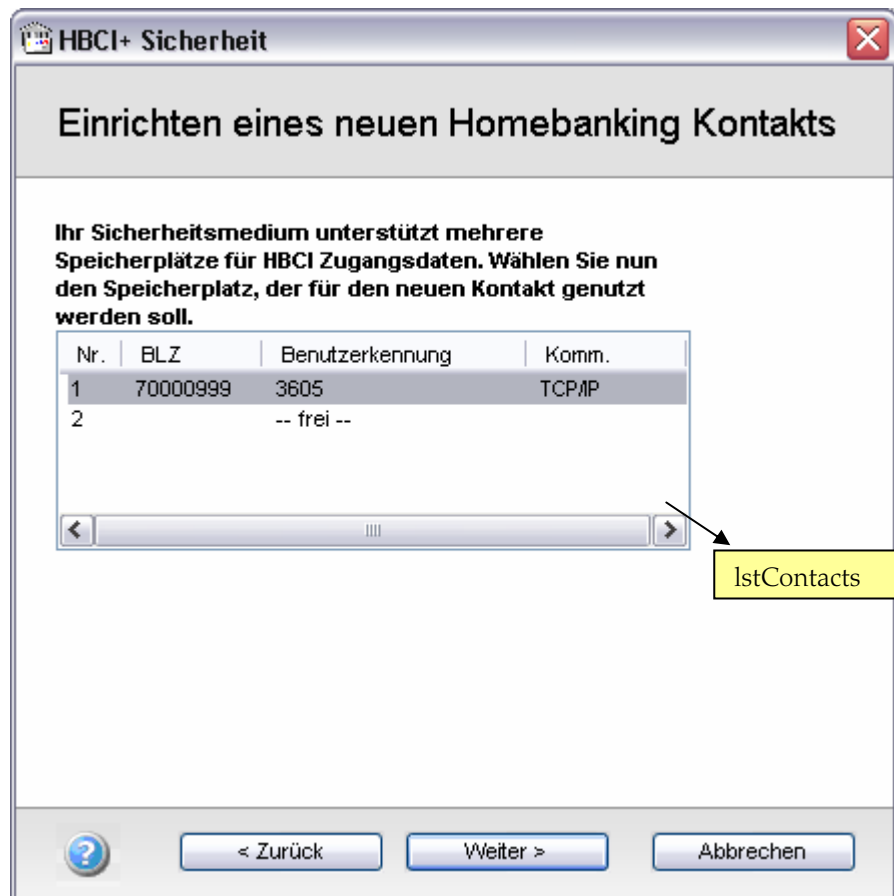
lblText3 (Optional)

Hier wird für Klasse 1 Leser der Text "EnterPin2_Card2_Text3" aus der Sstringtabelle angezeigt.

Mit Weiter wird die Eingabe der PIN auf dem Kartenleser aktiviert. Ja nach Gerät erscheinen unterschiedliche Dialoge und Eingabeaufforderungen, teils in einem neuen Fenster, teils auf dem Gerät selbst.

6.1.10 SelectContact

Falls auf einem Sicherheitsmedium mehrere Kontakte vorhanden sind, kann mit dieser Seite der gewünschte Kontakt ausgewählt werden.

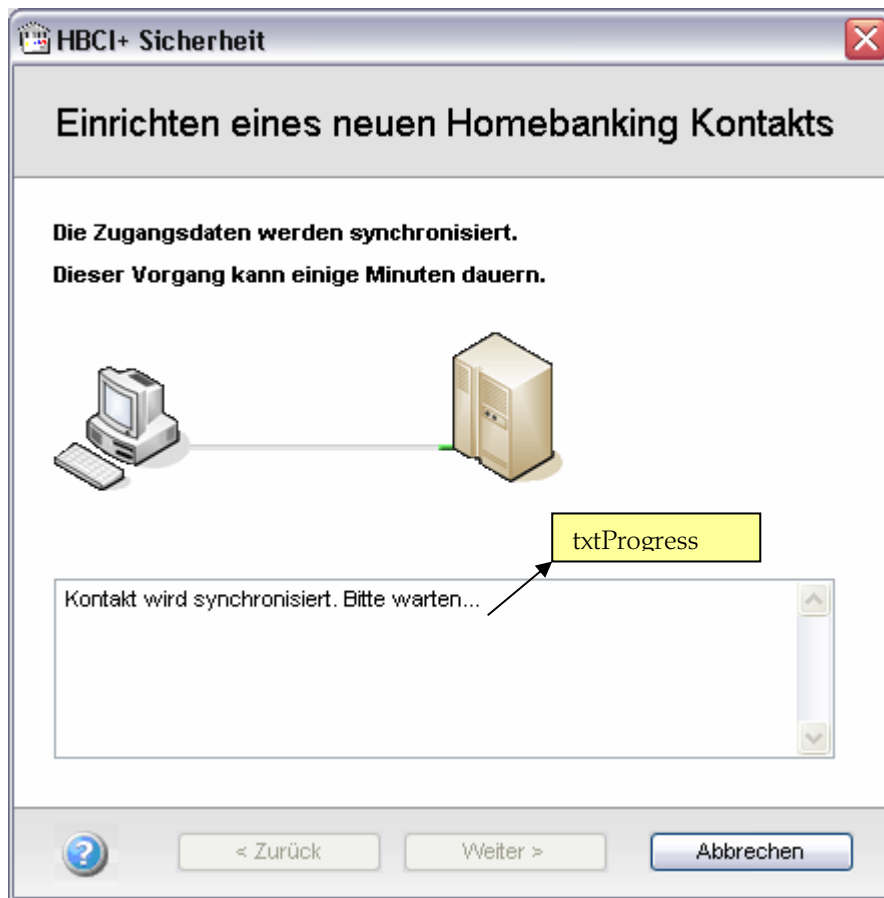


IstContacts

Enthält eine Liste aller verfügbaren Kontakte. Die Formatierung der Tabelle wird innerhalb des Dialogs vorgenommen und kann nicht modifiziert werden, die Spalten und der Beschriftung sind werden aus der Stringtabelle gelesen.

6.1.11 Synchronize

Diese Seite wird zum synchronisieren von neuen als auch von bestehenden Kontakten verwendet.



txtProgress (optional)

In diesem Feld wird der Prozessfortschritt durch Textausgaben dargestellt.

Weiter

Während der Verbindung mit der Bank ist die Weiter-Taste gesperrt. Sie wird freigegeben, wenn die Ermittlung der Daten abgeschlossen ist.

6.1.12 BankNotice

Wenn in der Antwort der Bank zusätzliche Warnungen oder Hinweise für den Benutzer enthalten sind, wird diese Seite angezeigt.



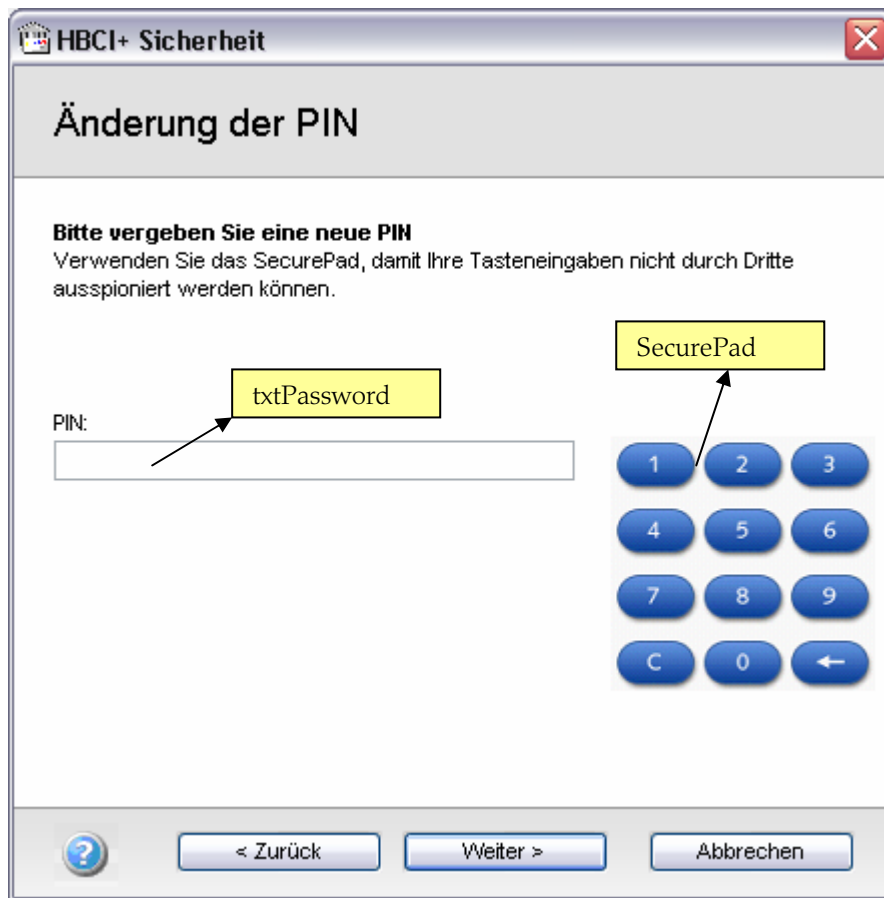
`txtText` (Optional)

In diesem Feld wird die Nachricht der Bank angezeigt.

Mit Weiter geht es zu der Seite, die in der jeweiligen Situation nach dieser Seite gekommen wäre, hier gibt es keine zusätzliche Logik.

6.1.13 EnterNewPin

Wird verwendet wenn eine neue Pin vergeben werden muss.



txtPassword

In diesem Feld wird die Eingabe einer neuen PIN erwartet. Falls eine leere PIN eingegeben wird, wird ein Hinweis ausgegeben ("AskForEmptyPin").

Das Feld übernimmt die Längenbegrenzung aus EinterPin2.

Weiter ist möglich sobald die minimale PinLänge eingegeben wurde.

6.1.14 EnterRetypePIN

Wird zur Bestätigung einer neu vergebenen PIN verwendet.

The screenshot shows a dialog box titled "HBCI+ Sicherheit" with a close button in the top right corner. The main heading is "Änderung der PIN". Below this, a message reads: "Bitte wiederholen Sie die neue PIN" followed by "Verwenden Sie das SecurePad, damit Ihre Tasteneingaben nicht durch Dritte ausspioniert werden können." The interface contains a text input field labeled "PIN:" with a yellow callout box "txtPassword" pointing to it. To the right is a "SecurePad" numeric keypad with buttons for digits 1-9, 0, a clear button (C), and a back arrow. At the bottom, there are three buttons: a help icon (?), "< Zurück", "Weiter >", and "Abbrechen".

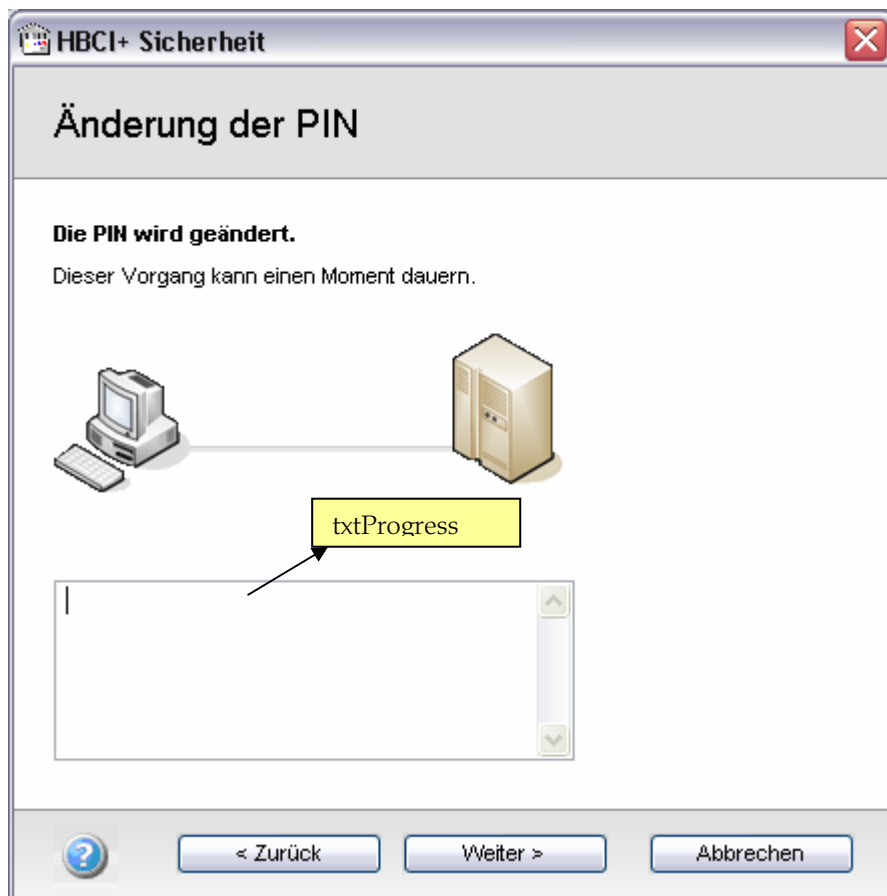
txtPassword

In diesem Feld wird die Wiederholung der neuen PIN erwartet.
Das Feld übernimmt die Längenbegrenzung aus EinterPin2.

Weiter ist möglich sobald die minimale PinLänge eingegeben wurde und die eingegebene PIN mit der neu vergebenen übereinstimmt.

6.1.15 ChangePin

Auf dieser Seite wird die eigentliche Änderung der PIN durchgeführt. Je nach Sicherheitsmedium muss dazu ein Geschäftsvorfall an die Bank gesendet werden (PIN), die PIN auf der Chipkarte geändert werden mittels Klasse 1 oder Klasse 2, je nach Leser oder die Verschlüsselung der Schlüsseldatei geändert werden. Je nach Sicherheitsmedium erscheinen hier die aktuellen Änderungen.



txtProgress (optional)

In diesem Feld wird der Prozessfortschritt durch Textausgaben dargestellt.

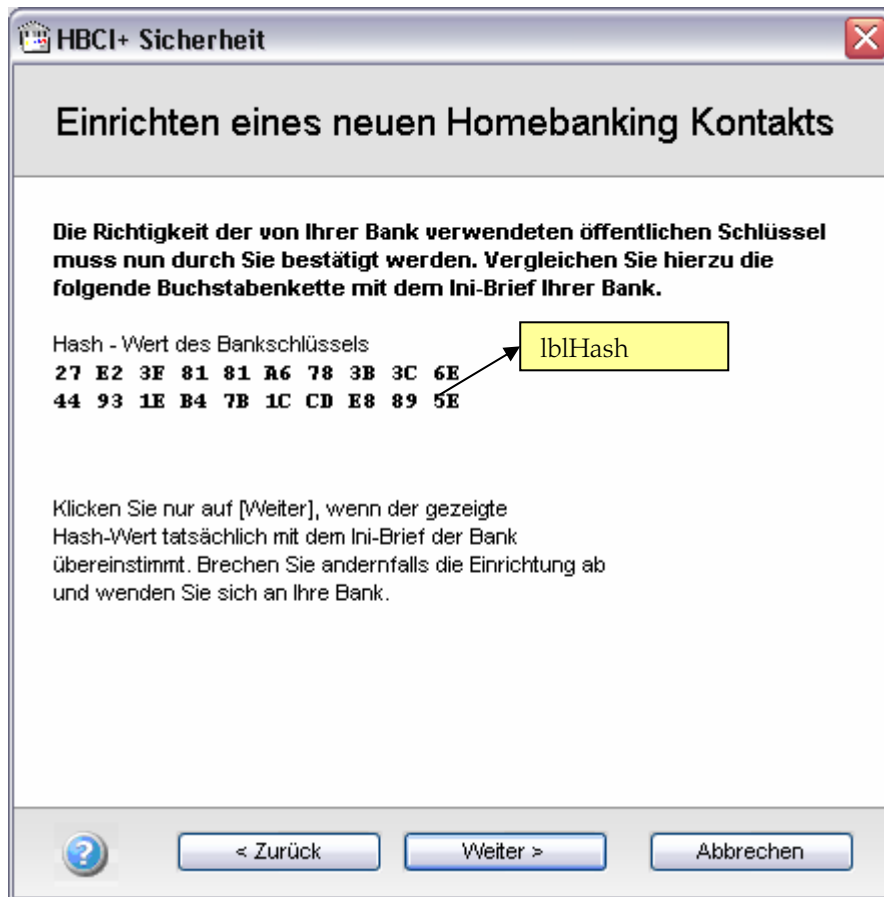
Weiter

Während der Verarbeitung der Daten ist die Weiter-Taste gesperrt. Sie wird freigegeben, wenn die Pin-Änderung abgeschlossen ist.

Diese Art von Seite wird auch für andere Aktionen verwendet die mit dem Kreditinstitut kommunizieren. Dazu gehören: ActivateTANList, RequestTANList, LockPinExec, UnlockPinExec, ShowTANListExec, LockKeysExec. Diese haben jeweils eigene Seiten die sich nur geringfügig unterscheiden.

6.1.16 VerifyBankkey

Wenn die Bankschlüssel von der Bank geholt werden und auf dem Sicherheitsmedium kein Hashwert hinterlegt ist, muss der Anwender diesen Wert mit dem Begleitschreiben der Bank vergleichen.

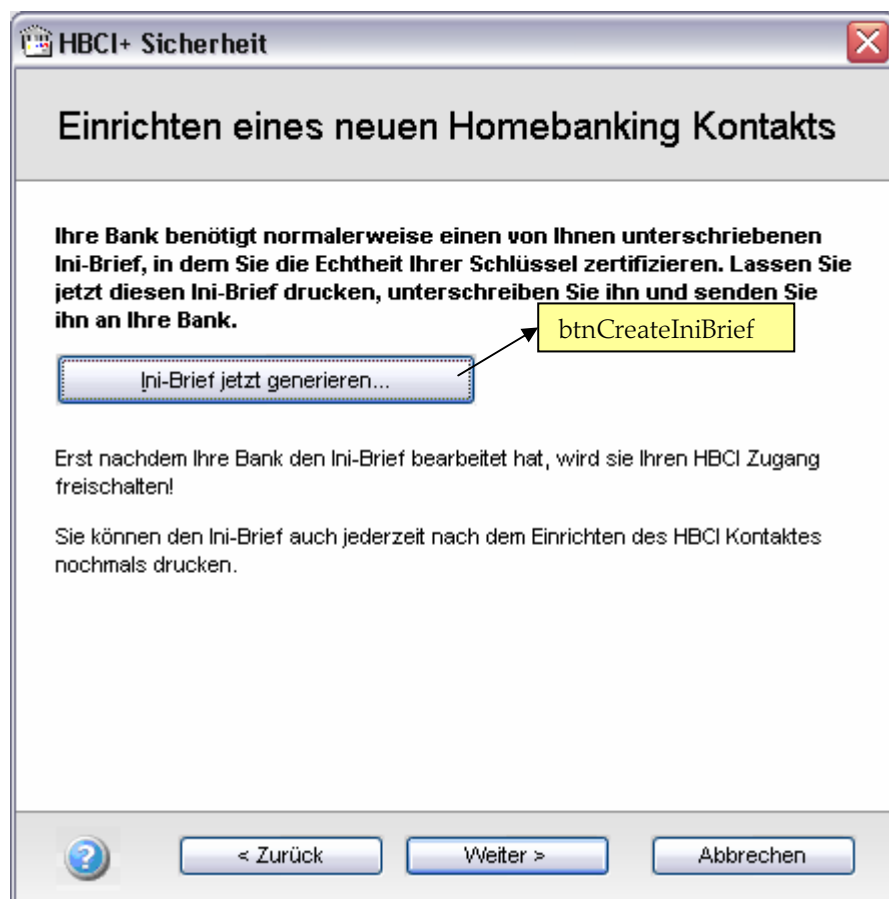


lblHash

Enthält den Hashwert des Bankschlüssels

6.1.17 SendIniBrief

Wenn die Schlüssel bei der Bank eingereicht wurden und auf dem Sicherheitsmedium keine Zertifikate vorhanden sind, muss ein Ini-Brief generiert werden, der einen Fingerprint des eingereichten Schlüssels enthält um die eingereichten Schlüssel eindeutig zu verifizieren.



btnCreateIniBrief

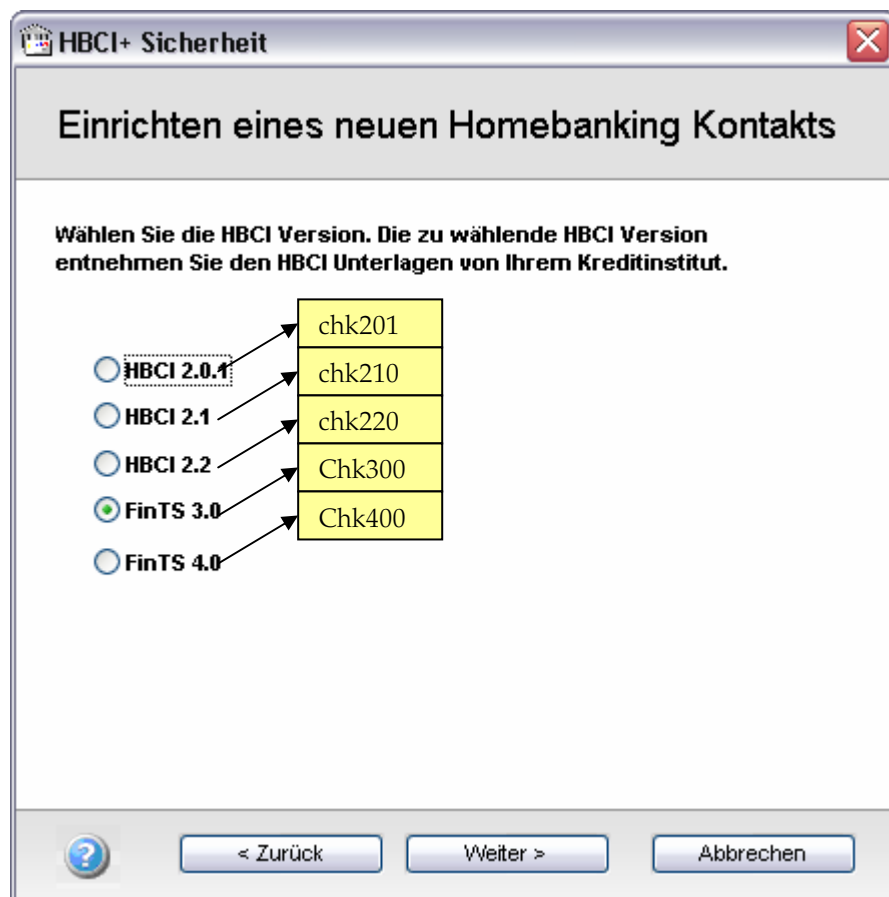
Damit wird ein Ini-Brief erzeugt, im Verzeichnis "Eigene Dateien" unter dem Namen "IniBrief.html" abgelegt und mit einem Aufruf der Shell angezeigt. Die dabei aufgerufene Applikation ist abhängig davon was für die Endung ".html" als Standard definiert wurde.

Weiter

Falls eine neue Schlüsseldatei angelegt wurde, wird diese nun gespeichert. Man endet immer auf der "Finish" Seite mit einem nicht-synchronisierten Kontakt. Die Synchronisation kann erst erfolgen wenn die Bank den Ini-Brief bearbeitet hat und der Kontakt freigeschalten wurde.

6.1.18 SelectHBCIVersion

Im Expertenmodus kann mit dieser Seite die Hbci-Version im Wizard bestimmt werden.



Chk201 - Chk400 (Optional)

Die einzelnen Radiobuttons sind optional, FinTS 4.0 könnte also weggelassen werden, wenn diese Option nicht zur Verfügung stehen soll.

6.1.19 EnterCommAddress (TCP/IP)

Im Expertenmodus (oder wenn bei GetBankInfo keine Verbindung zum Bankserver hergestellt werden konnte) kann hier die TCP/IP Verbindung zur Bank eingegeben werden.

HBCI+ Sicherheit

Einrichten eines neuen Homebanking Kontakts

Bitte geben Sie die Verbindungsdaten Ihres Kreditinstituts ein.

Um eine Verbindung mit dem HBCI-System Ihres Kreditinstituts aufbauen zu können, wird die Internet-Adresse des HBCI-Systems benötigt.
Die Internet-Adresse kann als numerische IP-Adresse (z.B. 123.123.123.123) oder als DNS-Adresse (z.B. hbc1.meinebank.de) angegeben werden.

Internet-Adresse:

Die Verbindung erfolgt über einen Socks5 Proxy.

txtCommAddress

Hier wird die Adresse des Bankservers erwartet. Der Port kann mit : abgetrennt angegeben werden. Wird kein Port angegeben wird Port 3000 verwendet.

chkSocks5 (Optional)

Falls die Verbindung ins Internet über einen Socks-5 Proxy Server geht, muss diese Schaltfläche angewählt werden. Die Details zu Socks5 können dann auf der Seite "EnterSocks5" eingegeben werden.

6.1.20 EnterCommAddress (HTTP)

Im Expertenmodus (oder wenn bei GetBankInfo keine Verbindung zum Bankserver hergestellt werden konnte) kann hier die HTTPS Verbindung zur Bank eingegeben werden.

HBCI+ Sicherheit

Einrichten eines neuen Homebanking Kontakts

Bitte geben Sie die Verbindungsdaten Ihres Kreditinstituts ein.

Um eine Verbindung mit dem HBCI-System Ihres Kreditinstituts aufbauen zu können, wird die Internet-Adresse des HBCI-Systems benötigt.
Die Internet-Adresse kann als numerische IP-Adresse (z.B. 123.123.123.123) oder als DNS-Adresse (z.B. hbc1.meinebank.de) angegeben

Internet-Adresse:

Base64 Encoding

txtCommAddress

Hier wird die Adresse des Bankservers als URL erwartet. Also Beispielsweise `https://server.de/xxx`.

Der Port kann mit `:` abgetrennt angegeben werden. Wird kein Port angegeben wird Port `80/443` verwendet.

chkBase64 (Optional)

Falls dies angewählt ist werden die Daten an den Server als base64 Encoded übertragen. Das ist bei den meisten Bankverbindungen derzeit der Fall.

6.1.21 EnterSocks5

Falls der Zugang zum Internet nur über einen socks5 Proxyserver möglich ist, können hier die Zugangsdaten für diesen Server eingegeben werden.

HBCI+ Sicherheit

Einrichten eines neuen Homebanking Kontakts

Bitte geben Sie die Verbindungsdaten Ihres Socks5 Proxyservers ein.

Ein Socks5 Proxyserver wird innerhalb von lokalen Netzwerken verwendet um den Zugang zum Internet für bestimmte Aufgaben freizuschalten.

Proxy-Server: **txtServer**

Port: **txtPort**

Benutzername: **txtUser**

Passwort: **txtPwd**

Wenn ein Passwort benötigt wird, und Sie es nicht speichern möchten, werden Sie vor jedem Verbindungsaufbau nach diesem Passwort gefragt. Das Passwort wird optional verschlüsselt gespeichert.

txtServer

IP Adresse oder Name des Socks5 Servers.

txtPort

Port, Standard ist 1080

txtUser (Optional)

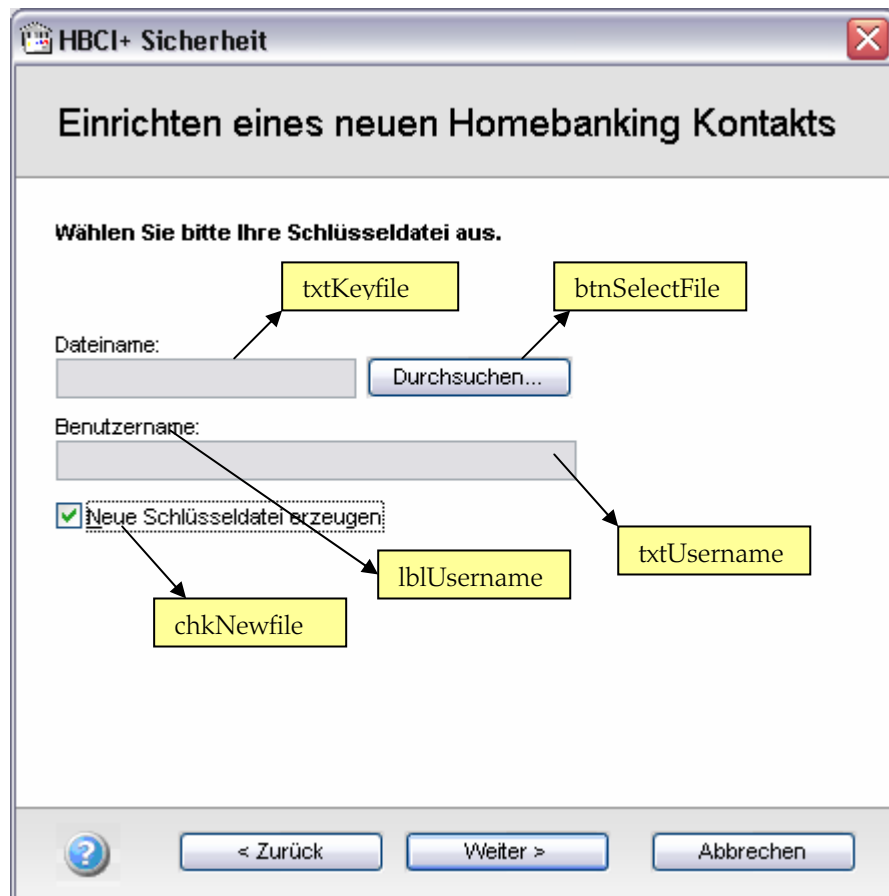
Optional der Benutzername für den Socks5 Server.

txtPwd (Optional)

Optional das Passwort. Dieses wird verschlüsselt abgelegt, allerdings ist der Schlüssel in den Komponenten hinterlegt.

6.1.22 EnterKeyfile

Diese Seite dient zur Auswahl einer bereits existierenden Schlüsseldatei oder der Anlage einer neuen Datei.



txtKeyFile

Dient zur Eingabe des Dateinamens einer Schlüsseldatei und kann mit der Datei die mit btnSelectFile ausgewählt wird befüllt.

btnSelectFile (Optional)

Dateiauswahldialog für die Schlüsseldatei

lblUsername (Optional)

Bezeichnung des Benutzernames, wird disabled wenn kein Benutzername erwartet wird.

txtUsername (Optional)

Wird für .Key Dateien erwartet, entspricht oft dem Dateinamen und wird mit diesem vorbelegt.

chkNewfile (Optional)

Wenn diese Schaltfläche angewählt wird, wird eine neue Schlüsseldatei angelegt. Eine Seite zum Speichern der Datei folgt im Anschluß.

6.1.23 EnterKeySize

Diese Seite wird immer aufgerufen, wenn eine neue Schlüsseldatei angelegt werden soll. Bei Format der Schlüsseldatei werden die möglichen Formate angeboten und in der anderen ComboBox werden die möglichen Längen angeboten.

Diese Seite erscheint beim Neuanlegen eines Kontaktes nur im Expertenmodus.

HBCI+ Sicherheit

Schlüsseländerung

Format der Schlüsseldatei.

Bitte wählen Sie das Format der Schlüsseldatei und die Länge des Schlüssels in Bit.

Format der Schlüsseldiskette:
RDH-1

Länge des Schlüssels in Bit:
768

Generell gilt: Je länger der Schlüssel, desto sicherer ist die Übertragung aber die Rechenzeit verlängert sich entsprechend.

< Zurück Weiter > Abbrechen

cmbFormat

Enthält alle verfügbaren Formate, abhängig von den angebotenen Sicherheitsverfahren der Bank im HISHV.

cmbSize

Enthält alle verfügbaren Schlüssellängen.

6.1.24 EnterKonto

Falls die Bank keine UPDs zurückliefert müssen die Konten auf dieser Seite von Hand eingegeben werden.

HBCI+ Sicherheit

Einrichten eines neuen Homebanking Kontakts

Kontendaten eingeben.
Ihre Bank hat keine Konten geliefert, Sie müssen ihre Konten manuell eingeben.

Kontonummer: **txtKontoNummer**

Kunden-ID: **lblCustomerID** **txtCustomerID**

Kontobezeichnung: **txtKontoName**

Kontowährung: **txtCurrency**

Kundenname: **txtKundenName**

Weitere Konten anlegen **chkEnterMore**

? < Zurück Weiter > Abbrechen

txtKontoNummer

Eingabe der Kontonummer

lblCustomerID (Optional)

Hier wird die Bezeichnung für die CustomerIDName des Kontakts eingetragen

txtCustomerID (Optional)

Eingabe der CustomerID

txtKontoName (Optional)

Names des Kontos

txtCurrency

Währung des Kontos

txtKundenName

Kundenname innerhalb des Kontos

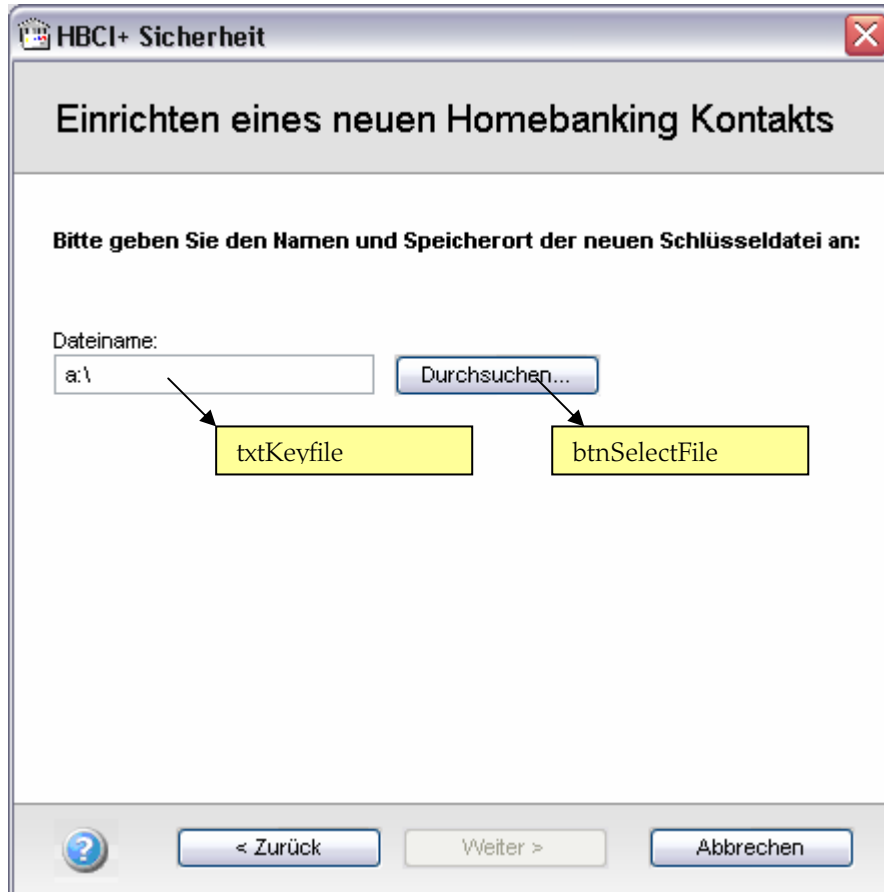
chkEnterMore (Optional)

Wenn ausgewählt können weitere Konten eingegeben werden.

Kontonummer, Währung und Kundenname sind Pflichtfelder die ausgefüllt werden müssen.

6.1.25 SaveKeyFile

Wenn ein Kontakt mit einer neuen Schlüsseldatei eingerichtet wird, dann wird am Ende wenn alles Erfolgreich war die Schlüsseldatei unter dem hier anzugebenden Speicherort abgespeichert.



txtKeyfile

Es wird ein gültiger Dateiname erwartet. Wenn keine Endung angegeben wird, wird automatisch "RDH" ergänzt.

btnSelectFile (Optional)

Standard SaveFileAs Dialog zur Auswahl des Dateinamens

6.1.26 ChangeKeys

Der Einleitungsdialog zur Schlüsseländerung.



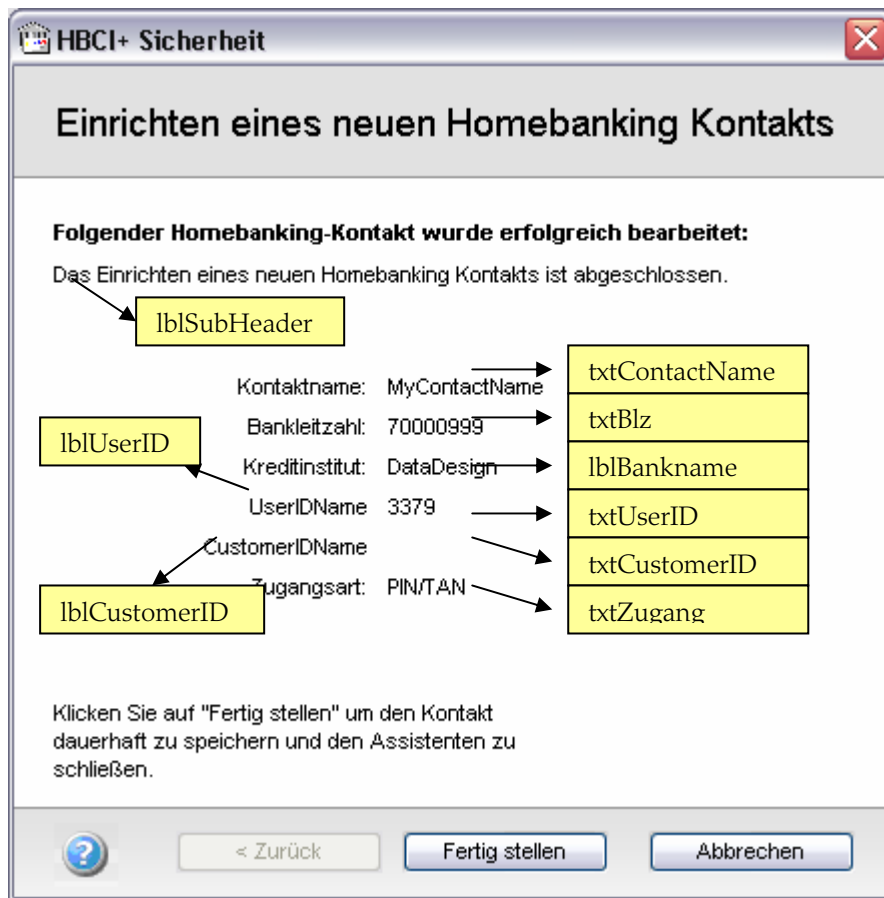
6.1.27 UpgradeKeys

Der Einleitungsdialog zum Sicherheitsprofilwechsel.



6.1.28 Finish

Der abschließende Dialog im Wizard wenn alle Aktionen fehlerfrei waren.



lblSubHeader (Optional)

Wird mit dem Text " DialogFinalNewContact " aus der Stringtabelle gefüllt. Wobei hier je nach Wizard unterschiedliche Texte verwendet werden wie DialogFinalSynchronise oder DialogFinalChangePin.

txtContactName (Optional)

Der Name des Kontakts

txtBlz (Optional)

Die Bankleitzahl

lblBankName (Optional)

Name der Bank wie in <GetBankInfoRs><BankInfo><BankName> enthalten.

lblUserID (Optional)

Bezeichnung der UserID für diesen Kontakt. Dabei wird optional der Wert <GetBankInfoRs><BankConnect><UserIDName> oder der " UserIDName" des Kontakts verwendet. Falls nichts für dieses Kreditinstitut hinterlegt ist und der Wert im Kontakt auch leer ist, wird hier der Wert aus der Dialogressource angezeigt, welcher standardmässig "Benutzerkennung" lautet.

txtUserID (Optional)

Wert der UserID die zu diesem Kontakt gehört.

lblCustomerID (Optional)

Bezeichnung der CustomerID für diesen Kontakt. Dabei wird optional der Wert <GetBankInfoRs><BankConnect><CustomerIDName> oder der " CustomerIDName" des Kontakts verwendet. Falls nichts für dieses Kreditinstitut hinterlegt ist und der Wert im Kontakt auch leer ist, wird hier der Wert aus der Dialogressource angezeigt, welcher standardmässig "Kunden-ID" lautet.

Falls " (nicht belegt)" enthalten ist, wird das Feld nicht angezeigt.

txtCustomerID (Optional)

Wert der CustomerID die zu diesem Kontakt gehört.

txtZugang (Optional)

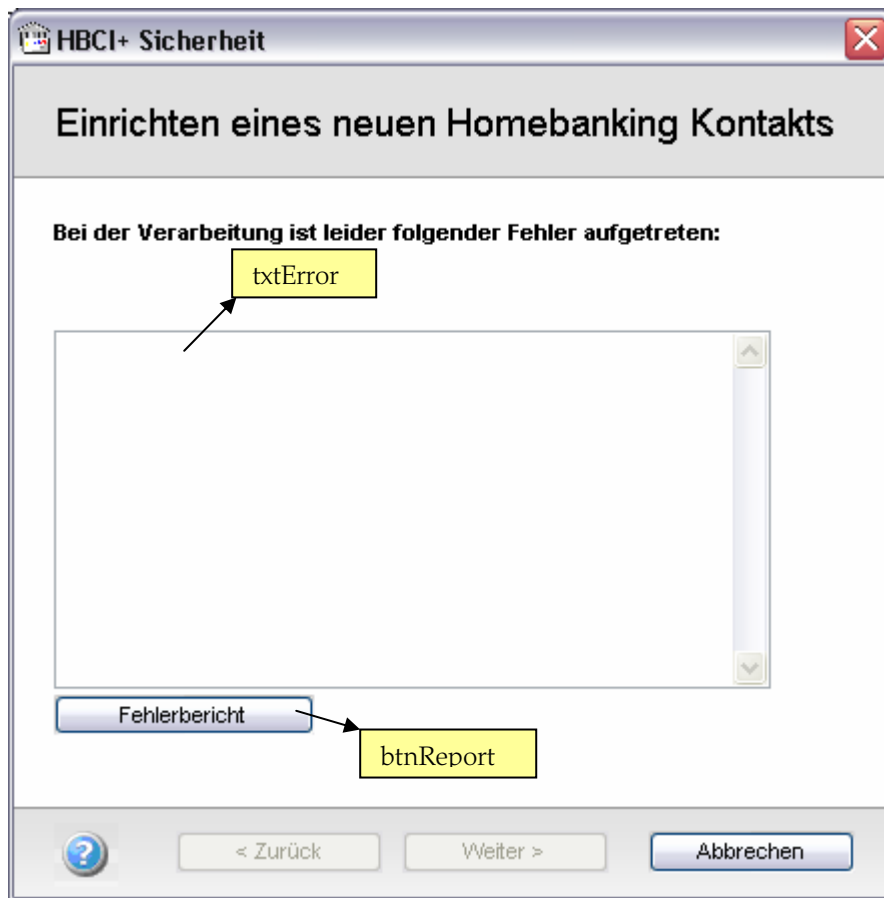
Bezeichnung der aktuellen Zugangsart. Dabei wird je nach Zugangsart folgender Text aus der Stringtable verwendet: SecurityKeyFileText, SecurityCardText oder SecurityPINTANText.

Mit OK wird der Wizard dann mit Erfolg abgeschlossen und die geänderten Daten des Kontakts werden gespeichert.

Wird der Wizard abgebrochen bleiben die Daten des Kontakts unverändert bzw werden verworfen.

6.1.29 Error

Diese Wizardseite erscheint immer wenn ein Fehler aufgetreten ist.



txtError

Dabei handelt es sich um ein mehrzeiliges Textfeld das die eigentlich Fehlerbeschreibung enthält.

btnReport

Öffnet ein Fenster mit einem detaillierten Fehlerbericht. Dabei handelt es sich um den Inhalt der Datei HbciLog.txt, welche alle Vorgänge in der DDBAC mitprotokolliert. Diese Protokollierung ist jedoch beim Endkunden in der Regel abgeschaltet, sodass die Datei HbciLog.txt keine Aufzeichnung der Vorgänge enthält.

Der Wizard protokolliert jedoch ab dem Öffnen des Fensters alle Vorgänge im Speicher mit, unabhängig davon ob die Protokollierung aktiviert ist. Dieses Protokoll wird dann in dem Fenster Fehlerbericht angezeigt und kann zur Analyse des Problems verwendet werden.

Weiter

Ist nicht möglich, der Wizard muss an dieser Stelle abgebrochen werden oder mit zurück nochmal versucht werden.

6.1.30 EnterTANListNr

Diese Seite erscheint, wenn für einen Vorgang eine TAN Listennummer benötigt wird.

HBCI+ Sicherheit

Aktivieren einer neuen TAN Liste

Bitte geben Sie die Nummer der TAN Liste ein.
Für die Bearbeitung dieses Auftrags wird die TAN Listennummer ihrer neuen TAN Liste benötigt.

Um eine TAN-Liste zu aktivieren, benötigen Sie je eine TAN der alten und der neuen zu aktivieren.

TAN Listennummer:

Optional labels: **lblTextActivate** and **lblTextLock**

Buttons: **< Zurück**, **Weiter >**, **Abbrechen**

lblTextActivate (Optional)

Wird nur angezeigt, wenn die TAN Listennummer zum Aktivieren der TAN Liste benötigt wird.

lblTextLock (Optional)

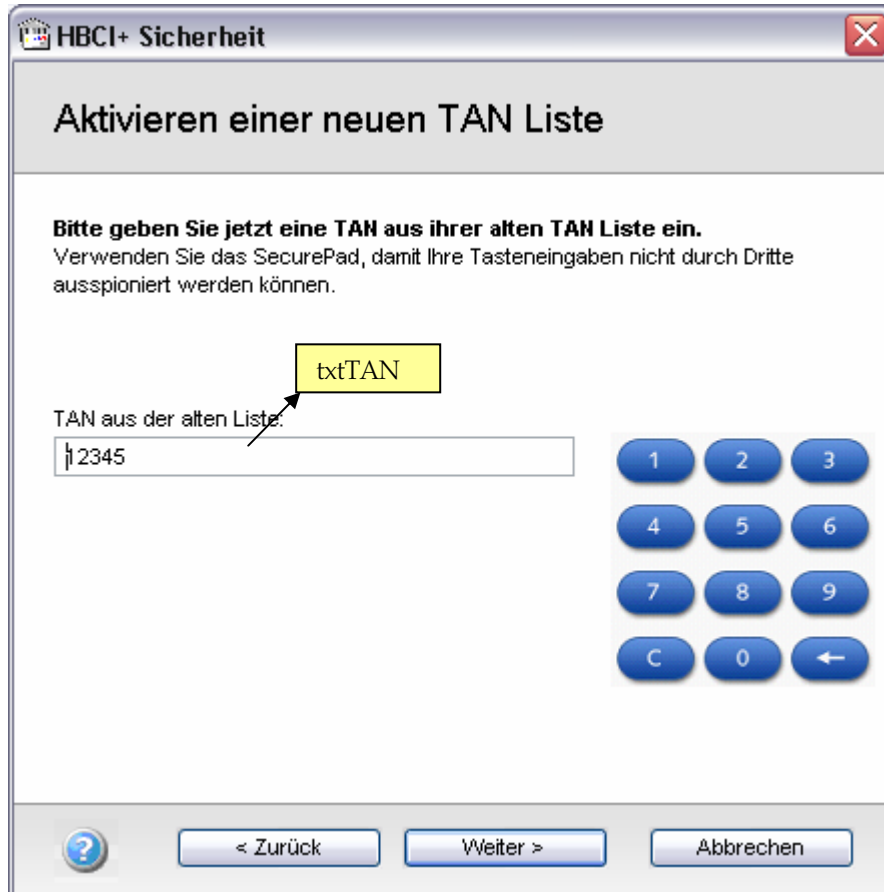
Wird nur angezeigt, wenn die TAN Listennummer zum Sperren der TAN Liste benötigt wird.

txtListNr

Hier wird die Eingabe der TAN Listennummer erwartet.

6.1.31 EnterOldTAN

Wir beim Aktivieren einer TAN Liste benötigt. Hier muss der Anwender ein TAN aus der alten TAN Liste eingeben.



txtTAN

Hier wird die Eingabe einerTAN erwartet.

Das SecurePAD ist optional.

Analog zu diesem Dialog gibt es auch einen EnterNewTAN Dialog der im Wesentlichen gleich aufgebaut ist.

6.1.32 ChangeTANMethod

Erlaubt die Auswahl aus mehreren Zwei-Schritt-TAN-Verfahren

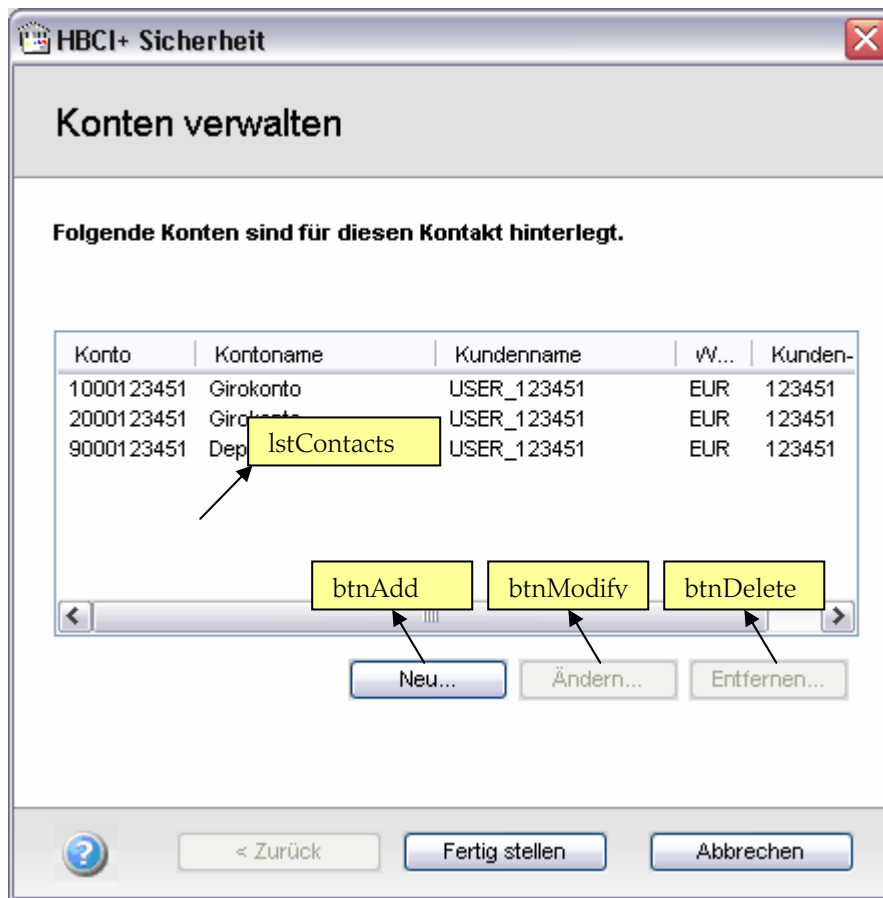


chkTan1 - chkTan11

Die Felder werden mit den verfügbaren TAN Verfahren und ihren Bezeichnungen aus den BPDs aufgefüllt und können vom Anwender ausgefüllt werden. Überflüssige Felder werden versteckt.

6.1.33 EditContact

Hier können die einzelnen Konten zu einem Kontakt bearbeitet werden.



IstContacts

Enthält alle Konten zu diesem Kontakt. Die Spaltenheader werden aus der Stringtabelle geladen:
<EditContact_Account>, <EditContact_AccountName>, <EditContact_UserName>,
<EditContact_Currency>, <EditContact_CustomerID>

btnAdd (Optional)

Taste zum Hinzufügen eines Kontakts, dabei wird die Seite EnterKonto aufgerufen, anschließend gelangt man wieder auf diese Seite.

btnModify (Optional)

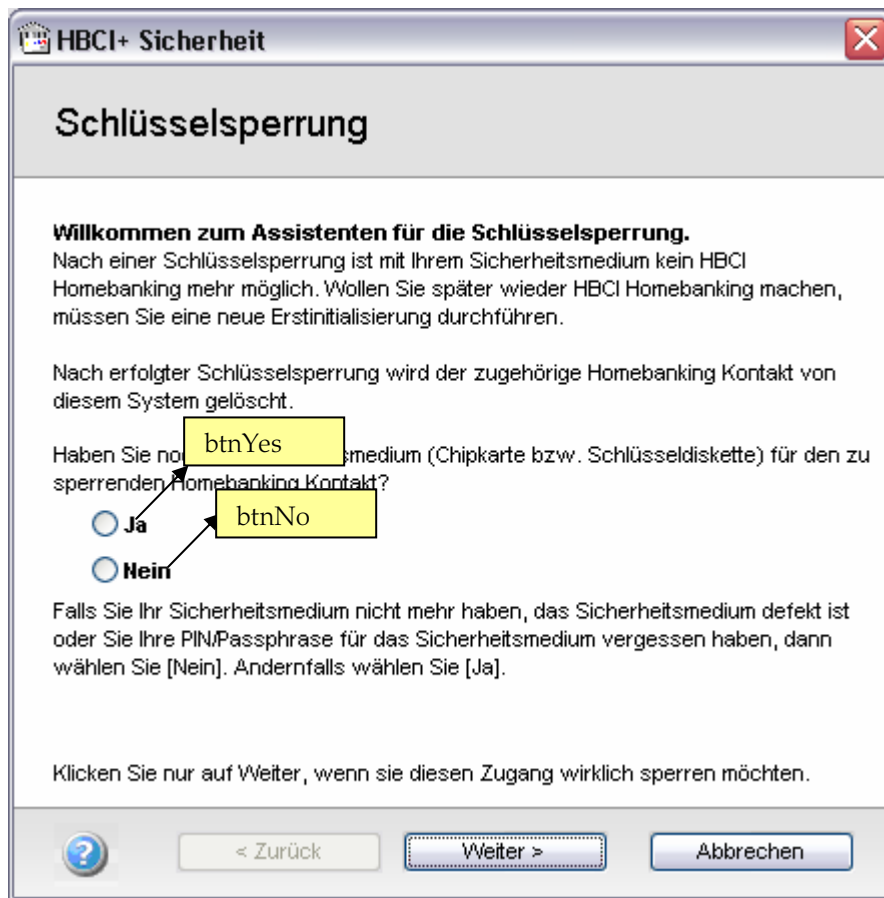
Damit kann das gewählte Konto geändert werden. Dies wird ebenfalls auf der Seite EnterKonto gemacht, anschließend gelangt man wieder auf diese Seite.

btnDelete (Optional)

Nach Rückfrage (EditContact_Delete) wird das Konto gelöscht.

6.1.34 LockKeys

Hier wird die Schlüsselsperrung eingeleitet



btnYes (Optional)

Die Schlüssel werden bei der Bank gesperrt und auf dem Medium deaktiviert.

btnNo (Optional)

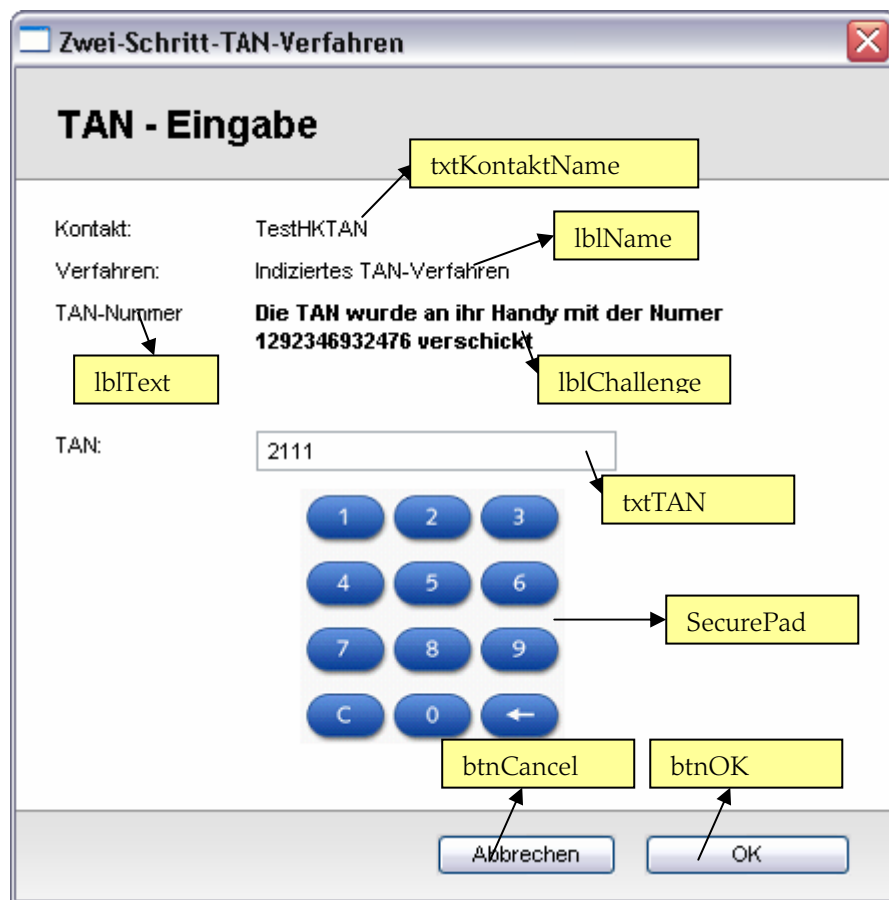
Wenn kein Medium vorhanden ist, werden die Schlüssel anonym bei der Bank gesperrt.

6.2 Dialoge

Neben den Wizards gibt es auch allein stehende Dialoge wie die TAN Eingabe, die in einem eigenen Fenster dargestellt werden. Hier kann der Wizardrahmen nicht verwendet werden, da die Dialoge unterschiedlich groß sein können. Hier folgt die Liste aller Dialoge in der DDBAC:

6.2.1 dlgEnterTan

Dieser Dialog wird zur Eingabe einer TAN für das Zwei- oder Ein-Schritt-TAN Verfahren verwendet.



Dialogtitel:

Falls es sich um ein Zwei-Schritt-Tan Verfahren handelt wird der Titel aus der StringTabelle "EnterTwoStepTanTitle" dort angezeigt.

txtKontaktName (Optional)

wird mit dem Inhalt von <Customer><Contact> gefüllt.

lblName (Optional)

Wird mit dem Inhalt des Feldes "Verfahrensparameter/RueckgabewertText" aus dem Request gefüllt.

IblText (Optional)

Wird mit dem Inhalt des Feldes "Verfahrensparameter/ TANVerfahrenName" aus dem Request gefüllt. Sollte der Name leer sein, wird der Text "EnterTwoStepTanChallenge" aus der Stringtabelle angezeigt.

IblChallenge (Optional)

Wird mit dem Inhalt des Feldes "ITAN/ Challenge" aus dem Request gefüllt.

txtTAN

Hier wird die Eingabe der TAN erwartet. Wenn im Request das Feld "Verfahrensparameter/ TANLen" einen numerischen Wert größer null enthält, wird die maximale Länge der Eingabe auf diesen Wert beschränkt.

SecurePad (Optional)

Erlaubt die Eingabe der TAN mittels des SecurePads. Da die Eingabe der TAN im Klartext erfolgt ist die Eingabe nicht wirklich sicherer.

btnCancel (Optional)

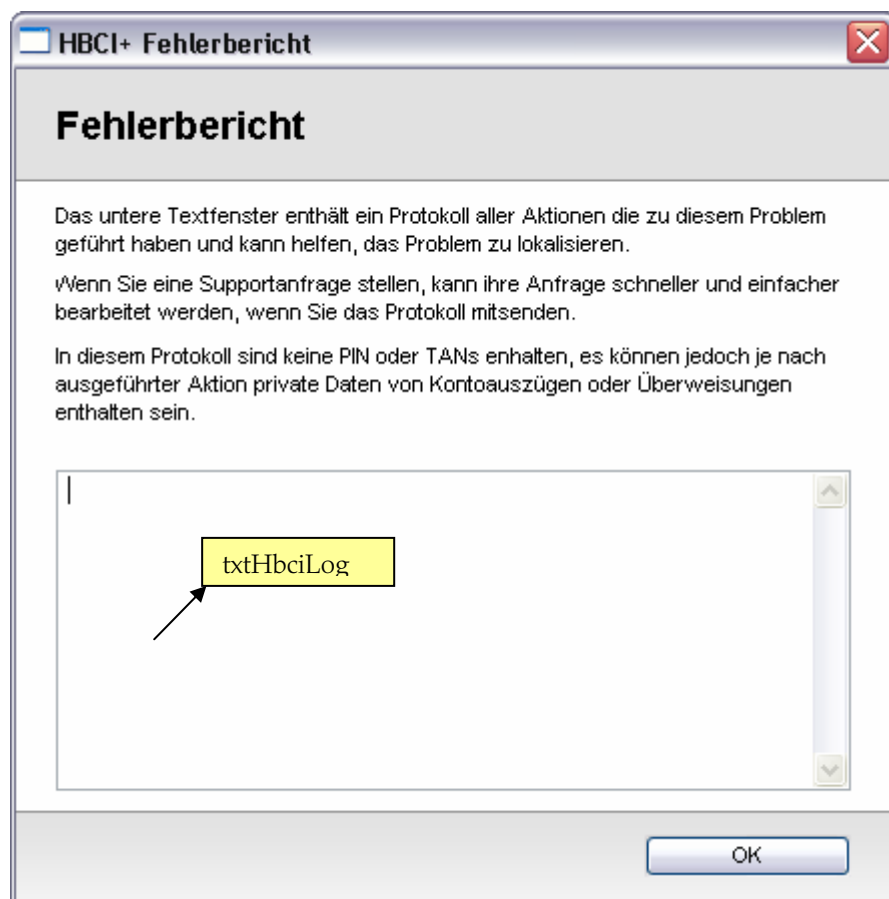
Damit wird die Eingabe der TAN abgebrochen

btnOK

Die Eingabe der TAN wird abgeschlossen. Diese Taste ist deaktiviert, solange das Feld txtTAN leer ist.

6.2.2 ErrorReport

Wird verwendet um detaillierte Fehlerberichte wie das HbciLog anzuzeigen um diese dann zur weiteren Analyse verwenden zu können.



txtHbciLog

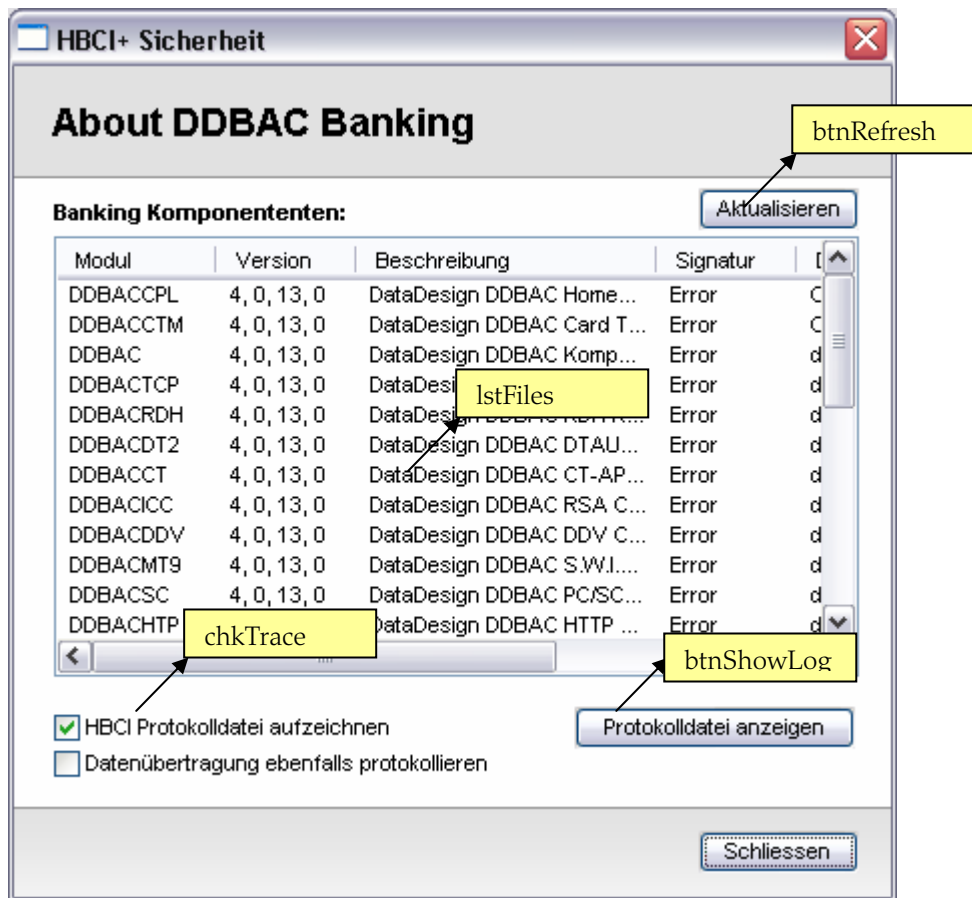
Mehrzeiliges Textfeld in welchen das Protokoll ausgegeben wird.

OK

Schließt den Dialog

6.2.3 dlgAbout

Zeigt den About Dialog mit der Liste aller Komponenten:



IstFiles (Optional)

Enthält die Liste alle Komponenten die von FilelistRq zurückgeliefert werden.

btnRefresh (Optional)

Liste aktualisieren

chkTrace (Optional)

Hiermit kann die Protokollierung in die HBCILOG.TXT Datei an- und abgeschaltet werden.

btnShowLog (Optional)

Hiermit wird die HBCILOG.TXT angezeigt, dabei wird das Programm aufgerufen, das auf dem jeweiligen System mit der Endung .txt verknüpft ist.

7 Implementierung

7.1 XAML Dialoge verändern

Um die XAML Dialoge zu verändern gibt es mehrere Möglichkeiten. Zum einen kann man die Callbackschnittstelle von BACStorage benutzen und die anfragen nach Dialogressourcen direkt beantworten.

Eine andere Möglichkeit ist, alle geänderten XAML Dateien und auch eigene Bilder in ein Verzeichnis zu stellen und die DDBAC anschließend darauf zu verweisen. Die DDBAC versucht dann als erstes die Ressourcen aus diesem Verzeichnis zu laden und verwendet die Callbackschnittstelle, wenn die Ressource nicht in dem Verzeichnis gefunden wird.

Das gewünschte Verzeichnis kann über die Klasse BACBanking eingestellt werden. Mit SetOptions ("ResourceDirectory") kann das gewünschte Verzeichnis gesetzt werden. Der Wert wird dabei nicht in die Registry geschrieben.

Zum Testen kann auf der Registryschlüssel \Software\DataDesign\DDBAC\ResourceDirectory verwendet werden. In diesem Fall werden aber alle Applikationen welche die DDBAC verwenden verändert. Dies kann zu unerwünschten Effekten führen und sollte von daher ausschließlich für Testzwecke verwendet werden.

Im HBCIPad und im HBCI.Net Sourcecode sind Codebeispiele hinterlegt, wie man mit dieser Schnittstelle den Zeichensatz in allen Dialogen ändert.

7.2 Zweisritt (Mehrfach) TAN Eingabe

7.2.1 Synchrone ein- oder mehrfach TAN Eingabe

Mit dem neuen Zwei-Schritt-TAN Verfahren wird die TAN mithilfe des <OpenTanDialogRq> von der Applikation abgefragt. Die Applikation kann daraufhin einen eigenen Dialog öffnen oder die passende TAN auf eine andere Art ermitteln und zurückliefert.

Mit dem Zwei-Schritt-TAN Verfahren wurde vom ZKA auch die neue Möglichkeit geschaffen, dass einzelne Aufträge mehrere Unterschriften (TAN) von unterschiedlichen Kunden (BACContact) erfordern. Dabei wird in den UPD zu dem Kunden zu jedem Geschäftsvorfall die erforderliche Anzahl an Unterschriften hinterlegt.

Wenn mehr als eine Unterschrift benötigt wird und die Bank mehrfache TANs in einem Auftrag erlaubt, wird die DDBAC automatisch nach den weiteren TANs Fragen. Im ersten <OpenTanDialogRq> kann man an den Feld <ErforderlicheSignaturen> erkennen, wie viele TANs erforderlich sind.

```
<OpenTanDialogRq>
...
    <MehrfachTANErlaubt>1</MehrfachTANErlaubt>
    <ZeitversetzteTANSErlaubt>0</ZeitversetzteTANSErlaubt>
</Verfahrensparameter>
<ITAN>
    <TANProcess>4</TANProcess>
    <AuftragsReferenz>N70718220070319152435</AuftragsReferenz>
    <Challenge>047</Challenge>
    <ErforderlicheSignaturen>2</ErforderlicheSignaturen>
    <VerbleibendeSignaturen>2</VerbleibendeSignaturen>
</ITAN>
...
</OpenTanDialogRq>
```

Die erste TAN wird dabei zu dem dialogführenden BACContact erwartet. Nachdem diese TAN abgesendet wurde, wird eine zweiter Kontakt benötigt, der die zweite Unterschrift leistet. Dazu gibt es mehrere Möglichkeiten.

Fall 1: Die Applikation macht nichts

In diesem Fall muss die DDBAC den zweiten Kontakt ermitteln. Dazu wird ein <SelectSignatureContactRq> gesendet in welchem die Kontaktdaten des zweiten Kontakts erwartet werden. Die DDBAC öffnet anschließend den Wizard mit dem Fenster SelectSigContact. Darin kann der Anwender den Kontakt auswählen und muss anschließend die PIN (Passwort) eingeben.

Dabei wird vor der Anzeige des Fensters ein <QueryPinRq> damit die Applikation die Möglichkeit hat, eine bereits hinterlegte PIN zu liefern.

Fall 2: Die Applikation kennt den Kontakt, der die 2. Unterschrift leisten muss

Falls es sich immer um den gleichen Kontakt handelt, kann im Kontakt des dialogführenden Kontakts im Feld „SecondSignature“ die ID des Kontakts angegeben werden, der die 2. Unterschrift leisten soll. Dies sieht beispielsweise so aus: CountyCode:BankCode:UserID. Die DDBAC wird versuchen diesen Kontakt zu laden (<ReadCustomerRq>) und benötigt anschließend die PIN. Dazu wird die DDBAC einen <QueryPinRq> senden um eine hinterlegte PIN abzufragen und wenn keine PIN hinterlegt ist, erscheint das Fenster (EnterPin2) mit der Aufforderung zur Eingabe der PIN.

Alternativ dazu kann die Applikation auch den <SelectSignatureContactRq> abfangen und dort selbst einen passenden Customer zurückliefern.

Sobald der Kontakt für die 2. Unterschrift bekannt ist, wird eine Nachricht an die Bank gesendet, welche daraufhin nach einer TAN für die 2. Unterschrift fragen wird. Hier sendet die DDBAC wiederum einen <OpenTanDialogRq>, jedoch für den Kontakt mit der 2. Unterschrift.

Mit einer 3. Unterschrift würde sich nun der Vorgang wiederholen, es muss ein Kontakt ausgewählt werden, PIN eingeben und die Anfrage wird abgesendet. Diese Schleife wird solange wiederholt bis <VerbleibendeSignaturen> den Wert 1 angenommen hat und damit die letzte TAN zu dem Auftrag gesendet wurde.

Anschließend sendet die Bank die Rückmeldung zum eigentlich Auftrag und die Transaktion wurde mit 2 oder mehr Unterschriften erfolgreich durchgeführt.

7.2.2 Asynchrone Mehrfach TAN Eingabe

Neben der synchronen TAN Eingabe, also der TAN Übermittlung im gleichen Dialog gibt es die Option die TAN später zu übertragen. Das ist die asynchrone TAN Eingabe. In diesem Fall wird im ersten Dialog nur die erste TAN übertragen. Die Antwort der Bank im HBCI Format sieht dann so aus:

```
HIRMG:2:2+0010::Nachricht entgegengenommen.'  
HIRMS:3:2:3+0020::Auftrag ausgefuehrt.'  
HITAN:4:2:3+2++N70718220070319152435'
```

In der zweiten Transaktion der Antwortnachricht ist das HITAN Segment enthalten und darin das Feld „Referenz“ in diesem Fall der Wert „N70718220070319152435“. Diese Referenz muss gespeichert werden.

Irgendwann später soll nun die 2. Unterschrift nachgereicht werden. Dazu muss ein Dialog mit dem ursprünglichen Kontakt geöffnet werden, also der Kontakt, welcher die Transaktion eingereicht hat. Anschließend muss mit ExecuteSegment eine HKTAN Segment ausgeführt werden. In diesem HKTAN Segment müssen 2 Felder gefüllt werden:

- Das Feld Prozess muss konstant 3 enthalten.
- Das Feld Referenz die gespeicherte Referenz.

In diesem erkennt die DDBAC, dass eine weitere Unterschrift nachgereicht werden muss. Dazu ermittelt die DDBAC wie im Kapitel vorher einen passenden 2. Kontakt (also der Kontakt, der nun die 2. Unterschrift leisten soll) und sendet den Auftrag ab.

Anschließend folgt wieder ein <OpenTanDialogRq> in welchem die TAN für die 2. Unterschrift eingegeben werden muss. Eine dritte Unterschrift kann ggf. auf dem gleichen Weg hinzugefügt werden.

Falls beide Methoden unterstützt werden, wird die DDBAC immer die synchrone Variante bevorzugen. Sollten jedoch das asynchrone Verfahren verwendet werden, dann kann beim ersten <OpenTanDialogRq> in der Antwort das Feld <AsynchronSignature> = 1 gesetzt werden. Damit wird verhindert, dass die DDBAC nach einer 2. Unterschrift fragt.

Die asynchrone Mehrfach TAN ist nur für TAN Prozessvariante 2 möglich.

Die synchrone Mehrfach TAN für TAN Prozessvariante 1 ist noch nicht implementiert.