ALF-BanCo: Api-Schnittstelle für .Net Framework

Inhaltsverzeichnis

1 Grundlagen

Auf Funktionen von ALF-BanCo kann über eine Api-Schnittstelle zugriffen werden. Um diese Schnittstelle zu nutzen, wird ein Lizenz-Schlüssel benötigt.
Sie erhalten für private und viele kommerzielle Zwecke einen kostenlosen Lizenz-Schlüssel auf Nachfrage. Nutzen Sie hierfür das Support-Formular auf unserer Homepage.

Der Schlüssel muss vor Nutzung der Schnittstelle übergeben werden.
Der Schlüssel setzt voraus, dass die Business-Version freigeschaltet ist oder in der Testphase genutzt wird.

Hinweis für Software-Hersteller:
Der Schlüssel wird nur von Ihrer Software benötigt. Ihre Kunden benötigen dann nur eine passende Lizenz (in der Regel Business-Version).
Es besteht die Möglichkeit, ALF-BanCo gemeinsam mit Ihrer Software zu vertreiben. Bitte wenden Sie sich wegen Kooperationen an die ALF AG oder informieren Sie sich im "Partner-Bereich" auf www.alf-banco.de.
Im Rahmen solcher Kooperationen gibt es auch die Möglichkeit
- weitere Funktionen der Schnittstelle zu nutzen
- die Schnittstelle auch für andere Lizenz-Stufen (z. B. Profi-Version) freizuschalten

Zur Nutzung der Schnittstelle gibt es ein Beispiel-Projekt in c#, das von unserer Homepage heruntergeladen werden kann.

In diesem Dokument werden die Funktionen und die Nutzung der Schnittstelle beschrieben.

1.1 Einbinden der Schnittstelle

Um die Schnittstelle in Ihrem Programm zu verwenden, müssen Sie einen Verweis auf folgende Assembly einbinden:

AlfBancoNetApi.dll

Diese DLL wird mit ALF-BanCo ausgeliefert. Sie liegt im Programmverzeichnis von ALF-BanCo und wird im GAC (Globaler Assembly Cache) registriert.

In c# kann die Schnittstelle dann z. B. über

```
Alf.Banco.NetApi.AlfApi mBanCo = new AlfApi();
```

genutzt werden.

Der Namespace der Api-Schittstelle ist also

```
using Alf.Banco.NetApi;
```

Die Assembly verwendet die .Net Framework Version 4.7.2.

1.2 Prüfen, ob ALF-BanCo installiert ist

Wenn Sie die Schnittstelle initialisieren, aber ALF-Banco ist auf dem PC nicht installiert, dann wird eine Exception ausgelöst.

1.3 Schnittstellen-Version

Die Schnittstelle ist so angelegt, dass diese bei den verschiedenen ALF-BanCo-Versionen weitgehend unverändert bleibt.

Um festzustellen, auf welche ALF-BanCo-Version die Schnittstelle zugreift, gibt es deshalb die Property "Version". Diese liefert die Version der Schnittstelle als String zurück - z.B. "8.0.0".
Über die erste Zahl der Versionsnummer kann festgestellt werden, welcher Lizenzschlüssel (z. B. für Version 7 oder 8) benötigt wird.

1.4 Anwendungsmöglichkeiten

Folgende Möglichkeiten bietet u. a. die Schnittstelle:

Auslesen von Daten aus der Datenbank:
- Umsätze eines Kontos (optional mit Zeitbegrenzung)
- Liste aller eingerichteter Konten

Anlegen und Ausführen von Online-Aufträgen:
- SEPA-Überweisungen und -Lastschriften
- Umsatz-Abfrage (im MT940- oder CAMT-Format)
- Saldo-Abfrage
- Auslandsüberweisungen
- SEPA-Terminüberweisungen
- SEPA-Sammelüberweisungen
- SEPA-Sammellastschriften

Bei ausgeführten Online-Aufträgen werden die Ergebnisse (z. B. neue Umsätze) stets in die Homebanking-Datenbank geschrieben.

Sie können danach die Daten mit den entsprechenden Funktionen aus der Datenbank auslesen.
Bei Fehlern können die Rückmeldungen des Instituts ausgelesen werden.

Für die Grundfunktionen wie z. B. Konto einrichten oder Kartenleser einrichten, verwenden Sie bitte weiterhin ALF-BanCo direkt.

1.5 Lizenz-Schlüssel

Um die Schnittstelle zu nutzen, muss über die Funktion mBanCo.SetLicenseKey() ein gültiger Lizenz-Schlüssel übergeben werden.

Folgende Funktionen sind aktuell mit einem kostenlosen Lizenzschlüssel verfügbar:

Datenbank-Funktionen
- Konten aus Datenbank auslesen (inkl. Salden)
- Umsätze aus Datenbank auslesen

Online-Funktionen
- Umsätze online abrufen
- Saldo online abrufen
- Einzelüberweisung

Folgende Funktionen sind auf Nachfrage über einen speziellen Key verfügbar, der je nach Nutzung (z.B. für private Nutzung) kostenlos ist:

- Terminüberweisung
- Auslandsüberweisung
- SEPA-Einzel-Lastschrift vom Typ Basis, Firmen oder Cor1 (automatische Sammel-Lastschrift mit 1 Auftrag, wenn Institut keine Einzel-Lastschrift unterstützt)
- SEPA-Sammelüberweisung
- SEPA-Sammellastschrift

1.6 Nutzung

Folgende Reihenfolge ist bei jeder Nutzung nötig:

1. Instanz von `AlfApi` erzeugen
2. Lizenz-Schlüssel setzen
3. Einloggen über .User.Login
4. Falls KontoID des gewünschten Kontos nicht bekannt ist: Konten auslesen über .Konten.Read(), um Konto auszuwählen
5. Daten aus Datenbank lesen (Umsätze) oder Online-Auftrag anlegen und versenden
6. Daten auswerten / anzeigen
7. Ausloggen
8. Instanz von `AlfApi` wieder freigeben

Im Folgenden finden sich für die meisten dieser Schritte Beispiele.

1.7 Beispiele

1.7.1 Lizenz-Schlüssel setzen und User einloggen

```
        int UserID = 0;
        string UserName = "";

        if (mBanCo.SetLicenseKey(txtKey.Text))
        {
            UserName = txtUserName.Text;
            if (mBanCo.User.Login(ref UserID, ref UserName, txtPasswort.Text))
            {
                //User eingeloggt
                txtUserName.Text = UserName;
            }
            else
            {
                //Feler beim Einloggen
                MessageBox.Show(mBanCo.Error.Text);
            }
        }
        else
        {
            //Fehler beim Setzen des Lizenz-Schlüssels
            MessageBox.Show(mBanCo.Error.Text);
        }
```

1.7.2 Konten auslesen

```
// Konten (neu) einlesen
if (mBanCo.Konten.Read())
{

    if (mBanCo.Konten.Count == 0)
    {
        //Keine Konten vorhanden
    }
    else
        for (int i = 0; i < mBanCo.Konten.Count; i++)
        {
            KontoDaten kto = mBanCo.Konten.Item(i);

            switch (kto.KontoTyp)
            {
                case alfKontoTyp.Depot:        sKontoTyp = "Depot-Konto";   break;
                case alfKontoTyp.DepotOffline: sKontoTyp = "Offline-Depot"; break;
                case alfKontoTyp.Normal:       sKontoTyp = "Online-Konto";  break;
                case alfKontoTyp.Offline:      sKontoTyp = "Offline-Konto"; break;
                case alfKontoTyp.PayPal:       sKontoTyp = "PayPal-Konto";  break;
                default:              sKontoTyp = kto.KontoTyp.ToString();  break;
            }


            // weitere Daten auslesen
        }
}
else
{
    //Fehler beim Auslesen der Konten
    Out(mBanCo.Error.Text);
}
```

1.7.3 Alle Umsätze auslesen

1.7.4 Auftrag "Umsätze holen" senden

1.7.5 Auftrag "SEPA-Überweisung" erzeugen und senden

1.7.6 Auftrag "Auslandsüberweisung" erzeugen und senden

1.7.7 Fehlerhaftes Element bei Online-Auftrag bestimmen

1.8 HBCI / FinTS

Für die Online-Übertragung der Daten wird in den meisten Fällen der HBCI/FinTS-Standard verwendet.
Die Dokumentation zu HBCI bzw. FinTS findet sich unter http://www.fints.org .

Bei Online-Zahlungen gilt für Name und Verwendungszweck dabei folgender Zeichensatz für Auslandszahlungen:

Zeichencode
Zugelassen sind

- die numerischen Zeichen 0 bis 9 (X'30' - X'39')
- die Großbuchstaben A - Z (X'41' - X'5A')
- die Sonderzeichen
  Leerzeichen „ „ = X'20'
  Punkt „.“ = X'2E'
  Komma „,“ = X'2C'
  Kaufmännisch „und“ „&“ = X'26'
  Trennstrich „-„ = X'2D'
  Plus-Zeichen „+“ = X'2B'
  Stern „*“ = X'2A'
  Prozent-Zeichen „%“ = X'25'
  Schrägstrich „/“ = X'2F'
  Dollar „$“ = X'24'
- sowie die Umlaute Ä, Ö, Ü und das ß. Hierfür gelten die Codierungen „Ä“ = X'5B', „Ö“ = X'5C', „Ü“ = X'5D', „ß“ = X'7E'.

Die Umlaute werden von der Schnittstelle automatisch entsprechend kodiert.
Andere Zeichen als die obigen werden nicht übermittelt.

Außerdem gibt es bestimmte Längenbegrenzungen
bei klassischen Zahlungen:

Feld	**Begrenzung**
Name	2 Zeilen á 27 Zeichen
Verwendungszweck	14 Zeilen á 27 Zeichen - je nach Bank sind teilweise weniger Zeilen zulässig

bei SEPA-Zahlungen:

Feld	**Begrenzung**
Name	1 Zeile á 140 Zeichen
Verwendungszweck	1 Zeile á 140 Zeichen (wird oft als 4 Zeilen á 35 Zeichen dargestellt)

Alle Datums-Angaben sind entsprechend der HBCI-Dokumentation im Format „JJJJMMTT“. Also z.B. „20150607“ für den 7.6.2015.

2 Klassen

Die Schnittstelle ist soweit möglich mit ActiveX-Klassen realisiert.
Hier werden alle Klassen beschrieben.

2.1 Klasse AlfApi

Diese Grundklasse stellt den Zugriff auf alle anderen Klassen zur Verfügung.
Die Klasse selbst hat nur eine Funktion zum Setzen des Lizenz-Schlüssels.

2.1.1 Funktion SetLicenseKey

Parameter:

Name	**Typ**	**Beschreibung**
Key	String	Lizenz-Schlüssel zur Nutzung der ActiveX-Schnittstelle

Rückgabewerte:

False	Lizenz-Schlüssel ungültig
True	Lizenz-Schlüssel gültig

2.1.2 Property Version

Hiermit kann die Versionsnummer der Schnittstelle als String ausgelesen werden.
Mögliche Werte: "8.0.0", "8.0.1" oder höher (bei Änderungen der Schnittstelle).

2.1.3 Klassen

Folgende Klassen werden bereitgestellt:

Klasse	**Beschreibung**

Error	Liefert den letzten Fehler (Code und Text)
Konten	Zum Auslesen der vorhandenen Konten aus der Datenbank
Online	Online-Aufträge anlegen und senden
Umsatz	Umsätze aus Datenbank lesen
User	Benutzer einloggen / ausloggen

Diese Klassen werden in den folgenden Kapiteln beschrieben.

2.2 Klasse clsAuftrag

Die Klasse verwaltet jeweils einen Online-Auftrag.
Ein neuer Auftrag wird über `.Online.Add` hinzugefügt.
Auf die angelegten Aufträge kann dann über `.Online.Item()` zugegriffen werden.

2.2.1 Funktion Daten

Gibt die Auftragsdaten dieses Auftrags zurück.

Parameter: keine

Rückgabewert: clsAuftragsDaten

Über clsAuftragsDaten kann auf die einzelnen Werte des Auftrags zugegriffen werden.

2.2.2 Funktion Response

Gibt die Antwort des Instituts auf diesen Auftrag zurück.

Parameter: keine

Rückgabewert: clsResult

Über clsResult kann auf die einzelnen Werte des Auftrags zugegriffen werden.

Eigenschaften von clsResult:

Count	Anzahl der Rückmeldungen zum Auftrag
Item	Liefert je eine Rückmeldung als clsMeldung

Eigenschaften von clsMeldung:

Code	Rückmeldungscode laut HBCI
ElementTyp	Fehlerhaftes Element
Kennung	"HIRMS": Rückmeldung zum Segment "HIRMG": Rückmeldung zur Gesamtnachricht
Param	Rückmeldungs-Parameter laut HBCI
Text	Rückmeldungs-Text laut HBCI

Bei einem Fehler kann das fehlerhafte Element über
`.Online.Item().Response(x).ElementTyp` bestimmt werden - siehe Beispiele.

2.2.3 Funktion Saldo

Liefert bei einer Salden-Abfrage den Saldo zurück.

Parameter: keine

Rückgabewert: clsSaldoDaten

Über clsSaldoDaten kann auf die einzelnen Werte der Saldos zugegriffen werden.

2.2.4 Funktion SaldoAbruf

Erzeugt für den aktuellen Auftrag eine Salden-Abfrage

Parameter: keine

Rückgabewerte:

False	Fehler aufgetreten
True	Auftrag erfolgreich angelegt - Status des Auftrags nun alfStatReady

Mögliche Fehlercodes über .Error.Code:

alfErrNoRights	Benutzer hat nicht das Recht, Salden abzurufen (Business-Version)

2.2.5 Funktion Status

Gibt den Status des Auftrags zurück

Parameter: keine

Rückgabewerte:

alfStatEmpty	Auftag neu angelegt, noch nicht mit Nutzdaten gefüllt
alfStatReady	Auftrag bereit zum Senden
alfStatUnsent	Auftrag wurde nicht versendet - z.B. wegen Benutzerabbruch
alfStatSent	Auftrag versendet - Status unklar (z.B. wenn keine Antwort des Instituts erhalten wurde)
alfStatError	Auftrag versendet, Fehlerrückmeldung erhalten
alfStatWarning	Auftrag versendet, Warnung erhalten (Autrag in der Regel trotzdem ausgeführt)
alfStatSuccess	Auftrag erfolgreich ausgeführt

2.2.6 Funktion Umsatz

Liefert bei einer Umsatz-Abfrage die gelieferten Umsätze zurück.

Parameter: keine

Rückgabewert: clsUmsAbruf

Über clsUmsAbruf kann auf die einzelnen Umsätze und den zugehörigen MT940-Datensaczt.

2.2.7 Funktion UmsatzAbruf

Erzeugt für den aktuellen Auftrag eine Umsatz-Abfrage

Parameter:

Name	Typ	Beschreibung
VonDatum	String	Optional: Datum, ab dem die Umsätze geholt werden
BisDatum	String	Optional: Datum, bis zu dem die Umsätze geholt werden

Datumsformat: "JJJJMMTT" Z.B. "20090420" = "20.04.2009"

Falls VonDatum und BisDatum nicht angegeben, werden alle Umsätze, die auf dem Bankrechner verfügbar sind, zurückgeliefert.
Wird nur VonDatum angegeben werden alle Umsätze ab diesem Datum bis heute geholt.

Rückgabewerte:

False	Fehler aufgetreten
True	Auftrag erfolgreich angelegt - Status des Auftrags nun alfStatReady

Mögliche Fehlercodes über .Error.Code:

alfErrNoRights	Benutzer hat nicht das Recht, Umsätze abzurufen (Business-Version)

2.2.8 Funktion Zahlung

Erzeugt für den aktuellen Auftrag eine Zahlung (Überweisung, Lastschrift).
Vorher müssen die Zahlungs-Daten über clsAuftrag.Daten gesetzt werden.

Parameter:

Name	Typ	Beschreibung

`Typ`	`eZahlTyp`	`alfZahlUeberweisung: Überweisung erzeugen`
		`alfZahlLastschrift: Lastschrift erzeugen`
		`alfZahlSEPAUeberweisung:SEPA-Überweisung erzeugen`
		`alfZahlSEPABasisLast: SEPA-Basis-Lastschrift erzeugen`
		`alfZahlSEPAFirmenLast: SEPA-Firmen-Lastschrift erzeugen`
		`alfZahlSEPACor1Last: SEPA-Cor1-Lastschrift erzeugen`
		`alfZahlSEPATerminUeberw:SEPA-Terminüberweisung erzeugen`
		`alfZahlAusland: Auslandsüberweisung erzeugen`

Rückgabewerte:

False	Fehler aufgetreten
True	Auftrag erfolgreich angelegt - Status des Auftrags nun alfStatReady

Mögliche Fehlercodes über .Error.Code:

alfErrNoRights	Benutzer hat nicht das Recht, Zahlungen zu senden (Business-Version)

2.2.9 Property KontoID

Hiermit kann die KontoID für diesen Auftrag gelesen oder gesetzt warden.
Wird die KontoID hier nicht geändert, werden alle Aufträge für die bei `.Online.Init()` übergebene KontoID ausgeführt.

2.1 Klasse clsAuftragsDaten

Diese Klasse verwaltet jeweils die Daten eines Auftrags.

Diese Eigenschaften werden für alle Aufträge benötigt:

Property	Typ	Inhalt
AusfDatum	String	Einzugs-Datum bei Lastschriften, Ausführungsdatum bei Gutschriften im Format „JJJJMMTT“
Betrag	clsBetrag	Betrag und Währung des Auftrags
GegenKonto	clsKonto	Konto des Empfängers/Zahlungspflichtigen
GegenName	colText	Name des Empfängers/Zahlungspflichtigen
Konto	clsKonto	Eigenes Konto
Name	colText	Name des Kontoinhabers/Auftraggebers der Zahlung

Diese Eigenschaften werden nur für SEPA-Lastschriften benötigt:

Property	Typ	Inhalt
AusstellDatum	String	Ausstellungsdatum des Mandats im Format „JJJJMMTT“
GläubigerID	String	Gläubiger-ID
MandtsRef	String	Mandats-Referenz des Zahlungspflichtigen
Sequenz	String	Sequenz der Lastschrift. Mögliche Werte sind FRST = erstmalig RCUR = wiederkehrend FNAL = letztmalig OOFF = einmalig

Diese Eigenschaften werden nur für Auslandsüberweisungen benötigt:

Property	Typ	Inhalt
Auftraggeber	clsAdresse	Daten des Auftraggebers (Strasse, Ort, Land)
Bankdaten	clsAdresse	Daten der Bank des Empfängers (Name, Strasse, Ort, Land)
Empfaenger	clsAdresse	Daten des Empfängers (Strasse, Ort, Land)
	clsAdresse	Daten des Auftraggebers (Name, Strasse, Ort, Land)
Entgeltregel	String	2-Stellige Entgeltregelung „00“, „01“ oder „02“
Zahlungsart	String	2-Stellige Zahlungsart. „00“ = Standardübermittlung

Besonderheiten bei Auslandsüberweisungen:

Für den Namen des Auftraggebers ist die Eigenschaft Name von clsAuftragsDaten zu verwenden.
Für den Namen des Empfängers ist die Eigenschaft GegenName von clsAuftragsDaten zu verwenden.
Für den Namen der Bank ist Bankdaten.Name zu verwenden.

Für alle 3 Namen können jeweils 2 Zeilen á 27 Zeichen angegeben werden:
Name(1) = „Name1“
Name(2) = „Name2“

Für Land (Auftraggeber, Empfänger, Bank) ist jeweils der internationale Ländercode – also z.B. DE für Deutschland, US für USA – zu verwenden.

2.2 Klasse Error

Über diese Klasse kann der letzte Fehler ausgelesen werden.

2.2.1 Property: RaiseError

Über diese Property kann festgelegt werden, ob bei Aufruf einer Funktion der Active-X-Schnittstelle ein auffangbarer Fehler ausgelöst werden soll.
Ist RaiseError = True, wird ein Fehler ausgelöst. Ist RaiseError = False (Standard-Wert), dann wird kein Fehler ausgelöst.
In diesem Fall gibt die aufgerufene Funktion bei einem Fehler "False" zurück und der Fehler kann anschließend über Error.Code ausgelesen werden.

2.2.2 Letzten Fehler auslesen

Über
.Error.Code
und
.Error.Text

kann jeweils der letzte Fehlercode und Fehlertext ausgelesen werden.

Die möglichen Fehler, die auftreten können, werden bei der jeweiligen Funktion beschrieben.
Alle Fehlercodes stehen über die Enum **AlfErrCodes** zur Verfügung.

2.3 Klasse Konten

Die Klasse dient zum Auslesen der aktiven Konten aus der Datenbank.
Konten, die in ALF-BanCo auf "inaktiv" gesetzt sind, werden nicht aufgelistet.
Bei der Business-Version werden bei eingeschränkten Benutzern nur die Konten zurückgeliefert, für die der Benutzer Rechte hat.

2.3.1 Funktion Count

Anzahl der ausgelesenen Konten

Parameter: keine

Rückgabewert: Long (Anzahl Konten)

2.3.2 Funktion Item

Liefert ein bestimmtes Konto

Parameter:

Name	Typ	Beschreibung
Index	Long	Index des Kontos, das angesprochen wird

Rückgabewert: clsKontoDaten

Über clsKontoDaten kann auf die einzelnen Werte des Kontos zugegriffen werden.

2.3.3 Funktion ItemByKontoID

Liefert ein bestimmtes Konto anhand der KontoID zurück.
Die KontoID ist innerhalb einer Datenbank (eines Benutzers) eindeutig und ändert sich nicht, wenn das Konto nicht gelöscht wird.

Parameter:

Name	Typ	Beschreibung
ID	Long	KontoID des Kontos, das angesprochen wird

Rückgabewert: clsKontoDaten

Über clsKontoDaten kann auf die einzelnen Werte des Kontos zugegriffen werden.

2.3.4 Funktion Read

Konten aus Datenbank lesen.

Parameter: keine

Rückgabewerte:

False	Fehler aufgetreten
True	Konten erfolgreich gelesen

Mögliche Fehlercodes über .Error.Code:

alfErrNotLoggedIn	Benutzer nicht eingeloggt

2.4 Klasse Online

Mit dieser Klasse werden die Online-Funktionen zur Verfügung gestellt.

2.4.1 Funktion Add

Auftrag zum Senden hinzufügen.

Parameter: keine

Rückgabewert: clsAuftrag

Über clsAuftrag kann auf die Daten des Auftrags zugegriffen werden.

2.4.2 Funktion ClearDB (ab Version 4.1.2)

Vorhandene offene Aufträge, die mit der Active-X-Schnittstelle angelegt wurden, werden aus der Datenbank gelöscht

Parameter: keine

Rückgabewert: keiner

2.4.3 Funktion Count

Liefert die Anzahl der hinzugefügten Aufträge zurück.

Parameter: keine

Rückgabewert: Long (Anzahl Aufträge)

2.4.4 Funktion Init

Initialisierung der Online-Aufträge.
Wird vor dem ersten Add() aufgerufen, um ggf. bereits vorhandene Aufträge zu löschen.

Parameter:

Name	Typ	Beschreibung
KontoID	Long	Konto, für das die Aufträge angelegt werden

Rückgabewerte:

False	Fehler
True	erfolgreich

2.4.5 Funktion Item

Über Item(x) kann auf die angelegten Aufträge zugegriffen werden, um z.B. die Ergebnisse auszulesen

Parameter:

Name	Typ	Beschreibung
Index	Long	Index des Auftrags, der angesprochen wird

Rückgabewert: clsAuftrag

2.4.6 Funktion Send

Aufträge senden

Parameter: keine

Rückgabewerte:

False	Fehler
True	erfolgreich

Mögliche Fehlercodes über .Error.Code:

alfErrUserAbort	Benutzer hat abgebrochen, bevor erster Auftrag versendet wurde

2.4.7 Property AskBeforeSend (Boolean)

Hiermit wird festgelegt, ob vor dem Senden der Aufträge der Kunde nochmal gefragt wird.
Standardwert: False = nicht nachfragen

2.4.8 Property AutoClose (Boolean)

Hiermit wird festgelegt, ob das Sendefenster nach dem Sendevorgang automatisch geschlossen wird.
Standardwert: True = automatisch schließen

2.5 Klasse Umsatz

Zugriff auf Umsätze aus der Datenbank

2.5.1 Funktion Count

Anzahl der ausgelesenen Umsätze

Parameter: keine

Rückgabewert: Long (Anzahl Umsätze)

2.5.2 Funktion Item

Liefert einen bestimmten Umsatz

Parameter:

Name	Typ	Beschreibung
Index	Long	Index des Umsatzes, der angesprochen wird

Rückgabewert: clsUmsatzDaten

Über clsUmsatzDaten kann auf die einzelnen Daten des Umsatzes zugegriffen werden.

2.5.3 Funktion Read

Umsätze aus Datenbank lesen.

Parameter:

Name	Typ	Beschreibung
KontoID	Long	Konto, dessen Umsätze ausgelesen werden sollen
VonDatum	String	Optional: Datum, ab dem die Umsätze gelesen werden
BisDatum	String	Optional: Datum, bis zu dem die Umsätze gelesen werden

Datumsformat: "JJJJMMTT" Z.B. "20090420" = "20.04.2009"

Falls VonDatum und BisDatum nicht angegeben, werden alle Umsätze des Kontos zurückgeliefert. Wird nur ein VonDatum angegeben, werden alle Umsätze ab diesem Datum zurückgeliefert.

Rückgabewerte:

False	Fehler aufgetreten
True	Umsätze erfolgreich gelesen

Mögliche Fehlercodes über .Error.Code:

alfErrNotLoggedIn	Benutzer nicht eingeloggt

alfErrNoRights	Benutzer hat nicht das Recht, Umsätze zu sehen (Business-Version)

2.5.4 Property MT940 (Read only)

Gibt die ausgelesenen Umsätze als MT940-String zurück.

2.6 Klasse User

Mit dieser Klasse wird das Einloggen / Ausloggen des Benutzers realisiert.

2.6.1 Funktion Login

Benutzer einloggen.
Wird Login() mit UserID = 0 bzw. einem nicht existierenden UserName aufgerufen, dann wird das Login-Fenster von ALF-BanCo angezeigt und der Anwender kann einen Benutzer auswählen und ein Passwort eingeben.
Läuft ALF-BanCo mit eingeloggtem Benutzer, dann wird in diesem Fall dieser Benutzer auch für die ActiveX-Schnittstelle eingeloggt.

Wird eine existierende UserID oder UserName angegeben, dann wird dieser ausgewählt und der Anwender kann ggf. noch das Passwort eingeben.

Wird ein existierenden Benutzer angegeben und zusätzlich ein Password übergeben, dann wird dieses überprüft und wenn das Passwort korrekt ist, wird der Benutzer eingeloggt.

Parameter:

Name	Typ	Beschreibung
UserID	Long	Eindeutige ID eines Benutzers - wird zurückgegeben wenn Benutzer eingeloggt ist
UserName	String	Name des Benutzers. Kann übergeben werden. Wird vom eingeloggten Benutzer zurückgeliefert
Password	String	Optional. Falls UserID und/oder UserName und Password angegeben werden, dann wird das Password überprüft und der Benutzer ggf. direkt eingeloggt

Rückgabewerte:

False	Benutzer konnte nicht eingeloggt werden bzw. Benutzer hat auf "Abbrechen" geklickt
True	Benutzer erfolgreich eingeloggt

Mögliche Fehlercodes über .Error.Code:

alfErrUserAbort	Benutzer hat im Login-Fenster "Abbrechen" geklickt
alfErrAlreadyLoggedIn	Ein Benutzer ist bereits eingeloggt
alfErrUserNotFound	Benutzer nicht gefunden (wenn UserID oder UserName übergeben wurde)
alfErrInvalidPassword	Passwort ungültig (wenn Passwort übergeben wurde)
alfErrOpenDatabase	Fehler beim Öffnen der Datenbank

alfErrVersionInvalid	ALF-BanCo-Version ist nicht Business-Version (Active-X-Schnittstelle nicht verfügbar)

2.6.2 Funktion Logout

Benutzer ausloggen.
Wenn ein Benutzer eingeloggt ist, muss dieser zuerst mit Logout() ausgeloggt werden, bevor ein anderer Benutzer gewählt werden kann.

Rückgabewerte:

False	Fehler aufgetreten
True	Benutzer erfolgreich ausgeloggt

2.6.3 Property IsLoggedIn (Read only)

Rückgabewerte:

False	kein Benutzer eingeloggt
True	Benutzer eingeloggt

2.6.4 Property UserName (Read only)

Rückgabewert: Name des eingeloggten Benutzers

2.1 Sammelaufträge

Ab Version 7.5.2 werden auch Sammelaufträge unterstützt.
Dafür gibt es in der Klasse
`clsAuftrag`
diese neuen Funktionen:

Name	Parameter	Beschreibung
AddSammelEinzel	Auftragsdaten	Fügt einen Einzelauftrag dem aktuellen Sammelauftrag hinzu
SammelEinzelCount	-	Gibt die Anzahl der aktuell im Sammelauftrag enthaltenen Einzelaufträge zurück
WriteSammel	Name	Schreibt den aktuellen Sammelauftrag mit übergebenem Namen in die Datenbank

und diese neuen Properties:

Name	Typ	Beschreibung
Datum	String	Ausführungsdatum oder Einzugsdatum des Sammelauftrags im HBCI-Format (YYYYMMTT). Bei Sammelüberweisungen kann das Datum leer sein, dann wird der Auftrag sofort ausgeführt.
GlaeubigerID	String	Gläubiger-ID für die Sammellastschrift
SammelTyp	eSammelTyp	Einer der folgenden Werte:

		alfSammelUeberweisung alfSammelBasisLast alfSammelFirmenLast alfSammelCor1Last
Sequenz	String	Sequenz der Sammellastschrift. Zulässig sind: FRST (= erstmalig) RCUR (= wiederkehrend) FNAL (= letztmalig) OOFF (= einmalig)

Beispielcode, um einen Sammelauftrag in die Datenbank zu schreiben (damit dieser ggf. später manuell ausgeführt wird) :

```
Dim SammelAuftrag As clsAuftrag
Dim Auftrag As clsAuftragsDaten

Dim KontoID As Long

KontoID = GetKontoID

mBanCo.Online.Init KontoID

Set SammelAuftrag = mBanCo.Online.Add

SammelAuftrag.SammelTyp = alfSammelUeberweisung

SammelAuftrag.GlaeubigerID = "DE98ZZZ09999999999"
SammelAuftrag.Sequenz = "RCUR"
SammelAuftrag.Datum = "20190510"

'Einzelaufträge setzen, hier ein Auftrag als Beispiel:

Set Auftrag = New clsAuftragsDaten
Auftrag.Daten.Betrag.Wert = "1,23"
Auftrag.Daten.GegenKonto.IBAN = "DE10020030123456 78"
Auftrag.Daten.GegenKonto.BIC = "BICADE01"
Auftrag.Daten.Vwz(1) = "Testüberweisung"
Auftrag.Daten.GegenName(1) = "Empfaenger-Name"
SammelAuftrag.AddSammelEinzel Auftrag
Set Auftrag = Nothing

'Sammelauftrag in Datenbank schreiben
SammelAuftrag.WriteSammel "name"

Set SammelAuftrag = Nothing
```

Beispielcode, um einen Sammelauftrag zu erzeugen und sofort zu senden:

```
Dim SammelAuftrag As clsAuftrag
Dim Auftrag As clsAuftragsDaten

Dim KontoID As Long

KontoID = GetKontoID
```

```
mBanCo.Online.Init KontoID

Set SammelAuftrag = mBanCo.Online.Add

SammelAuftrag.SammelTyp = alfSammelUeberweisung

SammelAuftrag.GlaeubigerID = "DE98ZZZ09999999999"
SammelAuftrag.Sequenz = "RCUR"
SammelAuftrag.Datum = "20190510"

'Einzelaufträge setzen, hier ein Auftrag als Beispiel:

Set Auftrag = New clsAuftragsDaten
Auftrag.Daten.Betrag.Wert = "1,23"
Auftrag.Daten.GegenKonto.IBAN = "DE1002003012345678"
Auftrag.Daten.GegenKonto.BIC = "BICADE01"
Auftrag.Daten.Vwz(1) = "Testüberweisung"
Auftrag.Daten.GegenName(1) = "Empfaenger-Name"
SammelAuftrag.AddSammelEinzel Auftrag
Set Auftrag = Nothing

'Sammelauftrag in Datenbank schreiben
SammelAuftrag.WriteSammel "name"

'Sammelauftrag senden
If mBanCo.Online.Send Then
      'Rückmeldungen auslesen...

End If

Set SammelAuftrag = Nothing
```