SUBSEMBLY

Subsembly GmbH
Hofmannstr. 7 b
81379 München

https://subsembly.com
info@subsembly.com

29. April 2021

BankingZV Integrationshandbuch

Erweiterte Funktionen und zusätzliche Anbindungsmöglichkeiten

Inhaltsverzeichnis

1 Einführung

Neben der Standard-Funktionalität von BankingZV stehen für professionelle Anwender leistungsfähige Integrationsschnittstellen zur Verfügung, über die eine effektive Nutzung und sachbezogene Anbindung von BankingZV ermöglicht wird.

Bei der Integration geht es vorrangig darum, die in BankingZV via Online-Banking erhaltenen Kontoumsätze an externe Buchhaltungssysteme weiterzuleiten, sowie von Buchhaltungssystemen vorbereitete Zahlungsaufträge in BankingZV zu importieren und auszuführen. Grundsätzlich stellt BankingZV hierfür bereits umfangreiche, manuell nutzbare Import/Export-Funktionen zur Verfügung. In diesem Dokument geht es darum, wie diese Import/Export-Funktionen erweitert und automatisiert werden können.

In den folgenden Kapiteln finden Sie:

- Eine Möglichkeit zur Erweiterung der unterstützten CSV-Datenformate für den Import und Export von Buchungen, Zahlungen und mehr.
- Einen Kommandozeilenmodus, mit dessen Hilfe ein automatisierter Import und Export, sowie die Ausführung verschiedener anderer BankingZV Funktionen ohne Benutzerinteraktion möglich ist.
- Die Möglichkeit Buchungsdateien in einem externen Datenverzeichnis zu verwalten und mit Hilfe einer EBICS Spooler Anwendung aus der Subsembly EBICS API dieses vollautomatisch zu befüllen.
- Eine von BankingZV definierte REST-Schnittstelle, welche von Buchhaltungsdiensten zur Kopplung mit BankingZV implementiert werden kann.

Je nach Systemanforderung und gewünschter Integrationstiefe bietet BankingZV für jeden Anwendungsfall eine optimale Integrationslösung.

2 Import/Export beliebiger CSV-Dateien

2.1 Überblick

Banken und Anwendungen nutzen unzählige unterschiedliche CSV-Formate für den Export und Import von Finanzdaten, wie z.B. Umsatzdaten.

Das **SUPA - Subsembly Payments Datenformat** versucht, ein einheitliches, CSV basiertes Datenformat zu etablieren. Um CSV-Dateien, welche nicht im SUPA Datenformat vorliegen, dennoch verarbeiten zu können, spezifiziert dieses Dokument den Aufbau einer formellen Abbildungsvorschrift, mit der beinahe beliebige CSV-Formate in SUPA-Daten überführt werden können.

Grundsätzlich ist der Aufbau von CSV-Dateien beispielsweise im RFC 4180 (siehe https://tools.ietf.org/html/rfc4180) definiert. Dieser grundsätzliche Dateiaufbau bildet die Basis für den CSV-Dateiparser. Alle zu importierenden CSV-Dateien müssen diesen grundsätzlichen Aufbau befolgen.

2.2 CSI-Datei

Eine **CSI-Datei** (CSV-Importfilter-Datei) enthält eine oder mehrere Abbildungsvorschriften für den Import und Export von beliebigen CSV-Dateien (Filter). Eine CSI-Datei soll immer die Dateiendung ".csi" verwenden.

Subsembly Banking liest beim Programmstart alle Dateien mit der Dateiendung ".csi" ein und sammelt alle darin definierten Filter. Ein Anwender wählt einen Filter über seinen Namen aus. Im Code wird ein Filter über seinen "`tag`" eindeutig identifiziert.

2.2.1 Wurzel-Objekt

Eine CSI-Datei ist eine UTF-8 codierte JSON Datei, welche ein einzelnes JSON Datenobjekt enthält. Dieses Wurzel-Objekt enthält folgende Felder.

Feldname	Typ	Len		Beschreibung
`tag`	string	..30	M	Eindeutige Kennung für eine .csi-Datei. Der Wert muss immer exakt "SubsemblyCSI" sein.
`version`	number		O	Versionsnummer.
`filters`	Array		O	Array von Filter-Objekten.

2.2.2 Filter-Objekt

Jedes Filter-Objekt beschreibt den Aufbau einer CSV-Datei und enthält die Regeln, um aus dieser CSV-Datei SUPA-Datenfelder zu extrahieren, bzw. eine CSV-Datei aus SUPA-Datenfeldern zu erstellen. Ein Filter-Objekt ist ein JSON-Objekt mit folgendem Aufbau.

Feldname	Typ	Len		Beschreibung
`identifier`	string	..256	M	ID des Dateiformats als umgekehrter DNS-Name, z.B. "de.commerzbank.csv" oder "com.subsembly.supa".
`class`	string	..	O	Klasse der Datenobjekte die in dieser Datei enthalten sind. Diese Klasse ist eine der für SUPA definierten Datenklassen, z.B. "Ntry", "Payee" oder "Paymt". Wird keine Klasse angegeben, so wird "Ntry", also Buchungen, angenommen.
`name`	string	..70	M	Name des CSV-Dateiformats. Dieser wird in der Benutzerschnittstelle für die Auswahl des Dateiformats verwendet.

Feldname	Typ	Len		Beschreibung
suffixes	string	..35	O	Mit Semikolon separierte Liste der für dieses Dateiformat üblicherweise verwendeten Dateiendungen. Wird dieser Wert nicht angegeben, wird als Vorgabe "csv" angenommen. Bei mehreren Einträgen wird der erste Eintrag als Standard-Dateiendung angenommen.
charset	string	..35	O	Zeichensatzcodierung der CSV-Datei. Normalerweise entweder "utf-8", "windows-1252" oder "iso-8859-1" (siehe .NET Encoding https://docs.microsoft.com/en-us/dotnet/api/system.text.encoding?view=netframework-4.8). Wird keine Codierung angegeben, so wird beim Import versucht die Codierung anhand der Daten automatisch zu erkennen, bei einem Export wird standardmäßig die Codierung "windows-1252" verwendet.
comma	string	1	O	Verwendetes Trennzeichen. Üblicherweise ist das ein Komma "," oder ein Semikolon ";". Wird keines angegeben, so wird "," angenommen.
decimalpoint	string	1	O	Dezimaltrennzeichen. Üblicherweise ist das ein Komma "," oder ein Punkt ".". Wird keines angegeben, so wird "," angenommen.

Feldname	Typ	Len		Beschreibung
datesequence	string	3	O	Reihenfolge der Zahlen in einem Datum für den Import. Muss die drei Buchstaben "D" – für Tag (Day) "M" – für Monat (Month) "Y" – für Jahr (Year) in der erwarteten Reihenfolge enthalten. Wird dieser Wert nicht angegeben, so wird die in Deutschland übliche Reihenfolge "DMY" angenommen.
datepattern	string		O	Für einen Datenexport wird hier das genaue Datums-Muster gemäß .NET Formatierungsregeln (siehe https://docs.microsoft.com/de-de/dotnet/standard/base-types/custom-date-and-time-format-strings) angegeben. Ist dieses nicht angegeben, so wird ein Datum im Format "dd.MM.yyyy" exportiert.
startrow	number		O	Zeilennummer innerhalb der CSV-Datei ab welcher der Import begonnen werden soll. Die Zeilennummer ist 0-basiert, d.h. die erste Zeile einer Datei hat die Zeilennummer 0. Wird keine Startzeile angegeben, so wird mit der ersten Zeile innerhalb der Datei begonnen. Bei einem Export wird diese Angabe ignoriert. Ein Export beginnt immer in der ersten Zeile einer Datei.

Feldname	Typ	Len		Beschreibung
noheader	bool		O	Wird dieser Wert als `true` angegeben, so wird keine Kopfzeile erwartet. In diesem Fall müssen alle Spaltenwerte durch die Angabe der Spaltenposition festgelegt werden. Standardmäßig wird eine Kopfzeile als erste Datenzeile erwartet.
noexport	bool		O	Wird dieser Wert als `true` angegeben, so soll dieser Filter nicht für den Datenexport verwendet werden. Ein Datenimport ist immer möglich.
columns	array		O	Array mit JSON-Objekten welche die einzelnen Spalten der CSV-Datei beschreiben.

Beispiel

```
{
  "identifier": "org.mybank.csv",
  "class": "Ntry",
  "name": "Meine Bank CSV Datei",
  "suffixes": "csv;txt",
  "charset": "utf-8",
  "comma": ";",
  "decimalpoint": ".",
  "datesequence": "MDY",
  "datepattern": "MM-dd-yyyy",
  "columns":
  [
    { /* ... */ },
    /* ... */
  ]
}
```

2.2.3 Column-Objekt

Ein Column-Objekt beschreibt eine für den Import relevante, bzw. eine zu exportierende Spalte der CSV-Datei. Außerdem beschreibt es, wie diese Spalte auf ein SUPA-Datenfeld abgebildet und konvertiert wird. Ein Column-Objekt hat folgende Felder.

Feldname	Typ	Len		Beschreibung
tag	string	..30	M	Name des definierten Wertes. Dies ist der Name des SUPA-Datenfeldes entsprechend der SUPA-Beschreibung, und nicht der Name aus der Kopfzeile der CSV-Datei.
position	number		C	Falls der Wert an einer festen Spaltenposition in der CSV-Datei enthalten ist, so gibt dies die 0-basierte Spaltennummer an.
names	string	..256	C	Falls die Spaltenposition durch eine Spaltenüberschrift gegeben ist, so wird hier eine Semikolon separierte Liste von möglichen Spaltenüberschriften für diesen Wert angegeben. Beispiel: "BuchungsDatum;BookingDate". Wurde bereits eine Spaltenposition festgelegt, so wird dieser Wert ignoriert. Bei einem Export wird immer der erste Spaltenname aus dieser Liste verwendet.

Wird weder eine Position per "position", noch ein Spaltenname per "names" festgelegt, so wird versucht, ein Spaltenname entsprechend dem angegebenen "tag" angenommen. Es ist möglich verschiedene Spalten unterschiedlich, also teilweise per "position" und teilweise per "names", festzulegen.

Die Reihenfolge der Spaltenbeschreibungen innerhalb des JSON-Arrays hat keinen Einfluss auf den Import. Bei einem Export werden die Spalten jedoch in genau der angegebenen Reihenfolge exportiert.

2.3 Tipps zur Verwendung von CSI-Dateien

Folgende Möglichkeiten stehen bei Verwendung von CSI-Dateien bei Import bzw. Export zusätzlich zur Verfügung:

- Ergänzend zur SUPA-Spezifikation kann für Buchungen statt der immer positiven Spalte "Amt" und des zugehörige "CdtDbtInd"-Vorzeichenindikators auch die spezielle Spaltenbezeichnung "_DecValue" verwendet werden. Diese Spalte kann auch negative Werte mit Vorzeichen enthalten.

- Ergänzend zur SUPA-Spalte "Category" kann für den Import zusätzlich eine Spalte "_SubCategory" spezifiziert werden. Ist eine solche Spalte definiert und in der CSV-Datei ein Wert hierfür vorhanden, wird dieser als Unterkategorie an den Wert der Spalte "Category" angehängt. Ist in der CSV-Datei kein Wert für "Category" vorhanden, so wird die Spalte "_SubCategory" ignoriert.

- Ist unklar, ob in der CSV-Datei eine Kontonummer als IBAN oder als einfache Kontonummer angegeben wird, sollte diese Spalte auf das SUPA-Feld "RmtdAcctNo" abgebildet werden. Beim Import wird automatisch erkannt, ob es sich um eine IBAN handelt.

- Der vorangegangene Punkt gilt entsprechend auch für BLZ/BIC und das SUPA-Feld "RmtdAcctBankCode".

2.4 Einschränkungen bei der Verwendung von CSI-Dateien

Folgende Beschränkungen müssen bei der Verwendung von CSI-Dateien bei Import bzw. Export berücksichtigt werden:

- Es gibt keine Möglichkeit, einen Saldo aus einer CSV-Umsatzdatei zu importieren.

- Es ist nicht möglich, Daten aus Spalten zu kombinieren. Ein Spaltenwert muss immer exakt auf ein SUPA-Feld abgebildet werden. Das ist problematisch bei CSV-Dateien, welche mehrere Spalten für Verwendungszweckzeilen haben.

- Es ist nicht möglich, Spalteninhalte zu konvertieren. Das ist insbesondere ein Problem bei SUPA-Feldern, welche spezielle konstante Inhalte erwarten.

- In der CSV-Datei muss der Aufbau von Dezimalzahlen und Datumsangaben konsistent sein. Es ist nicht möglich, dass unterschiedliche Spalten unterschiedliche Formatierungen haben.
- Eine Datumsangabe ohne Trennzeichen, z.B. "20200331", wird nicht unterstützt.

3 Kommandozeilenmodus

Für die einfache Integration mit Buchhaltungssoftware oder für eine automatisierte Nutzung kann BankingZV in einem speziellen Kommandozeilenmodus betrieben werden. Wird BankingZV im Kommandozeilenmodus aufgerufen, können verschiedene Funktionen ausgeführt werden, ohne dass das Programmfenster angezeigt wird.

Ein BankingZV Aufruf im Kommandozeilenmodus hat die folgende Form:

```
TopBanking.exe
  -Cmd
  -Wallet WalletFileName
  -Password Password
  -Token Token
  -Unattended
  -AcctIBAN AcctIBAN
  -AcctBankCode AcctBankCode
  -AcctNo AcctNo
  -AcctCurrency AcctCurrency
  -AcctCountry AcctCountry
  -ExportSplits
  -ExportBatches
  -ExportAdvised
  -ExportFrom FromDate
  -ExportTo ToDate
  -ExportIds Id1,Id2,...
  -ExportReplenish
  -ImportStmt SupaFileName
  -ImportMT940 MT940FileName
  -ImportCAMT CamtFileName
  -ImportSEPA SepaFileName
  -SubmitSEPA SepaFileName
  -ImportDTAZV DtazvFileName
  -ImportPaymts SupaFileName
  -SendRecv SendRecvFlag,SendRecvFlag,...
  -ExportAccts SupaFileName
  -ExportStmt SupaFileName
  -ExportMT940 MT940FileName
  -ExportCAMT CamtFileName
  -ExportPaymts SupaFileName
  -ExportDocList SupaFileName
  -ExportDocuments DirectoryName
```

Parameter werden durch Schlüsselworte mit vorangestelltem Minuszeichen eingeleitet. Alle Parameter sind optional und können weggelassen werden. Einige Parameter benötigen ein zusätzliches Argument, das direkt hinter dem Schlüsselwort nach einem Leerzeichen angegeben werden muss. Das Argument muss in Anführungszeichen stehen, wenn es Leerzeichen enthält.

Der erste Parameter muss **-Cmd** sein, um BankingZV in den speziellen Kommandozeilenmodus zu schalten.

Im Kommandozeilenmodus gibt es zwei Parameterarten: einfache Parameter und Kommandos.

3.1 Kommandozeilen-Hilfe

Über die Menüfunktion **Extras** steht in BankingZVeine **Kommandozeilen-Hilfe** zur Verfügung.

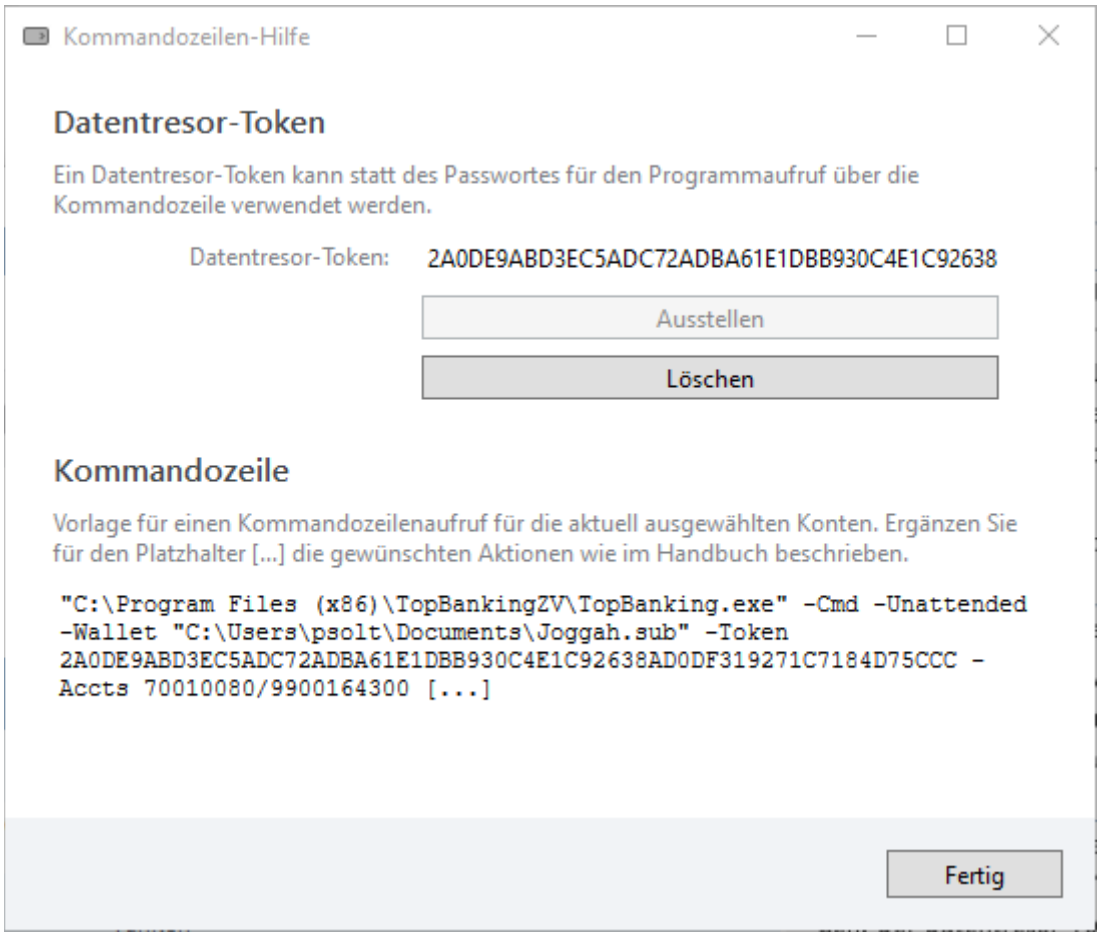

Abbildung 1: Kommandozeilen-Hilfe

Datentresor-Token
Statt des Datentresor-Passwortes kann auch ein Datentresor-Token für den Programmaufruf über die Kommandozeile verwendet werden. Über die Schaltfläche **Ausstellen** wird ein neues Datentresor-

Token generiert und angezeigt. Über die Schaltfläche **Löschen** werden alle Datentresor-Token gelöscht. Nur so ist es möglich, ein neues Datentresor-Token anzufordern.

Kommandozeile
Um einen Kommandozeilenaufruf möglichst einfach zu gestalten, wird für den gerade genutzten Datentresor sowie die aktuell ausgewählten Konten eine Vorlage für einen solchen Kommandozeilenaufruf unter Verwendung des im Dialog angezeigten Datentresor-Tokens erstellt. Der Platzhalter […] ist um die gewünschten Aktionen über Parameter und Kommandos zu ergänzen.

3.2 Einfache Parameter

Die folgenden einfachen Parameter werden im Kommandozeilenmodus unterstützt:

Parameter	Beschreibung
-Wallet *FileName*	Übergibt den Dateinamen des Datentresors, welcher geöffnet werden soll. Existiert diese Datei nicht, dann wird versucht eine neue Datentresordatei mit diesem Namen anzulegen. Konnte der Datentresor weder geöffnet noch erstellt werden, so endet TopBanking.exe sofort mit einem von Null verschiedenen Rückgabewert. Dieser Parameter muss im Kommandozeilenmodus immer angegeben werden.
-Password *Password*	Das Passwort des Datentresors, der geöffnet werden soll. Dieser Parameter sollte nicht mehr verwendet werden. Aus Sicherheitsgründen sollte für die Automatisierung immer ein Datentresor-Token mit dem Kommandozeilenparameter **-Token** verwendet werden.
-Token *Token*	Ein Datentresor-Token zum Öffnen des Datentresors. Dieser kann innerhalb des Programms über den Menüpunkt **Extras** > **Kommandozeilen-Hilfe** erzeugt werden. Der Datentresor-Token ersetzt das Passwort für den Datentresor.

Parameter	Beschreibung
-Unattended	Ist diese Option angegeben, so werden keine Benutzereingaben, z.B. für PIN oder TAN, durchgeführt. Sollte im Verlauf eine Benutzereingabe erforderlich sein, wird diese automatisch abgebrochen.
-Accts *Acct1,Acct2,...*	Optional eine mit Komma separierte Liste von Konten. Ein Konto kann in dieser Liste in folgenden Formen angegeben werden: *AcctIBAN* – Nur die IBAN des Kontos *AcctBankCode*/*AcctNo* – Bankleitzahl "/" Kontonummer *AcctBIC*/*AcctNo* – BIC "/" Kontonummer Zusätzlich kann noch ein Währungscode angehängt werden, falls dieser zur Unterscheidung erforderlich ist. Zum Beispiel: "50070010/12345678USD".
-AcctIBAN *AcctIBAN*	Optional die IBAN für das angegebene Kommando. Wird keine Kontonummer angegeben, so können nur Konto-übergreifende Kommandos ausgeführt werden. Alternativ kann auch *AcctNo* angegeben werden.
-AcctNo *AcctNo*	Optional die nationale Kontonummer für das angegebene Kommando. Wird keine Kontonummer angegeben, so können nur Konto-übergreifende Kommandos ausgeführt werden. Alternativ kann auch *AcctIBAN* angegeben werden.
-AcctBankCode *AcctBankCode*	Optional die Bankleitzahl des durch die Kontonummer *AcctNo* angegebenen Kontos. Ist keine Bankleitzahl angegeben, so wird diese bei der Kontosuche nicht berücksichtigt.
-AcctCountry *AcctCountry*	Optional das Länderkennzeichen zu *AcctNo*. Das Argument muss dem zweistelligen, alphanumerischen ISO Länderkennzeichen, z.B. **DE** für Deutschland, entsprechen. Ist dieses nicht angegeben, so wird dieses bei der Kontosuche nicht berücksichtigt.
-AcctCurrency *AcctCurrency*	Optional die Währung zu *AcctNo*. Das Argument muss dem ISO Währungskennzeichen, z.B. **EUR** für Euro, entsprechen. Wird dieser Parameter weggelassen, so wird die Währung bei der Kontosuche nicht berücksichtigt.
-ExportFrom *FromDate*	Anfangsdatum für einen Export von Umsatzdaten (JJJJ-MM-TT). Wird kein Anfangsdatum übergeben, so beginnt der Umsatzdatenexport mit der ersten vorhandenen Buchung.

Parameter	Beschreibung
-ExportTo *ToDate*	Enddatum für einen Export von Umsatzdaten (JJJJ-MM-TT). Wird kein Enddatum übergeben, so endet der Umsatzdatenexport mit der letzten vorhandenen Buchung.
-ExportSplits	Ist diese Option angegeben, so werden nicht die tatsächlichen Sammelbuchungen, sondern nur die im Programm aufgesplitteten Teilbuchungen exportiert.
-ExportBatches	Ist diese Option angegeben, so werden sowohl die übergeordneten Sammelbuchungen als auch die enthaltenen Teilbuchungen exportiert. Wird weder **-ExportSplits** noch **-ExportBatches** angegeben, so werden ausschließlich die Sammelbuchungen ohne die darin enthaltenen Teilbuchungen exportiert. ACHTUNG: Eine mit **-ExportBatches** exportierte SUPA Datei kann derzeit noch nicht korrekt importiert werden. Diese Exportoption dient aktuell nur für die Integration mit externen Anwendungen.
-ExportAdvised	Ohne diese Option werden mit Kontoumsätzen nur die tatsächlich gebuchten Kontoumsätze exportiert. Ist diese Option zusätzlich angegeben, so werden mit den Kontoumsätzen auch alle Buchungsvormerkungen exportiert.
-ExportIds *Id1,Id2,...*	Kommaseparierte Liste von Ids der zu exportierenden Objekte. Diese Option wirkt derzeit nur auf **-ExportDocuments**.
-ExportReplenish	Ist diese Option angegeben, so werden nur Objekte exportiert, welche noch nicht exportiert wurde. Wurde ein Objekt erfolgreich exportiert, egal ob interaktiv oder über einen Kommandozeilenaufruf, so wird es intern markiert und beim nächsten Aufruf mit -ExportReplenish nicht mehr erneut exportiert. Diese Option wirkt derzeit nur auf **-ExportDocuments**.
-ExportOverwrite	Ist die Option -ExportOverwrite angegeben, so werden gleichnamige, bereits existierende Dateien überschrieben. Ist die Option nicht angegeben, so wird beim Export immer ein eindeutiger Name generiert. Diese Option wirkt derzeit nur auf **-ExportDocuments**.

Parameter	Beschreibung
-ExportPerAcctFolder	Ist die Option -ExportPerAcctFolder angegeben, so wird unterhalb des angegebenen *DirectoryName* für jedes Konto, aus dem Dokumente exportiert werden, ein eigener Unterordner angelegt. Als Name für den Unterordner wird die jeweilige Kontonummer aus den Kontostammdaten verwendet. Diese Option wirkt derzeit nur auf **-ExportDocuments**.
-ExportSwiftMerge	Ist die Option -ExportSwiftMerge angegeben und werden in einem Aufruf mehrere SWIFT-Dokumente exportiert, so werden alle im Export enthaltenen SWIFT-Dokumente des gleichen Typs in einer Exportdatei zusammengefasst. Diese Option wirkt derzeit nur auf **-ExportDocuments**.

3.3 Kommandos

Nach den Parametern folgen auf der Kommandozeile die auszuführenden Kommandos. Werden keine Kommandos übergeben, so macht BankingZV gar nichts, außer evtl. im Rahmen der Parameterverarbeitung eine Datentresordatei anzulegen.

Jede Kommandoart darf nur einmal auf der Kommandozeile angegeben werden, aber es ist möglich, mehrere verschiedene Kommandos in nur einem Aufruf ausführen zu lassen. Ein Beispiel: Die folgende Kommandozeile aktualisiert die Umsatzdaten eines Kontos und exportiert sofort danach alle vorhandenen Umsatzdaten in die Datei „StmtOut.supa“.

```
TopBanking.exe
  -Cmd
  -Wallet “C:\MyFile.sub”
  -Password “MyPassword”
  -AcctNo 1234567890
  -AcctBankCode 10010010
  -AcctCurrency EUR
  -SendRecv Statements
  -ExportStmt “C:\StmtOut.supa”
```

Falls während der Kommandoverarbeitung ein Fehler auftritt, wird das Programm mit einem von Null verschiedenen Rückgabewert abgebrochen. Weitere auf der Kommandozeile enthaltene Kommandos werden in diesem Fall nicht mehr ausgeführt.

Die übergebenen Kommandos werden immer in der durch folgende Tabelle vorgegebenen Reihenfolge ausgeführt, unabhängig davon, in welcher Reihenfolge sie auf der Kommandozeile angegeben wurden!

Die verfügbaren Kommandos sind in folgender Tabelle aufgeführt.

Kommando	Beschreibung
-ImportStmt *SupaFileName*	Importiert alle Buchungen aus der genannten SUPA-Datei in das in den Parametern angegebene Konto. Die Dateiendung (.json/.csv/.supa) bestimmt das erwartete SUPA-Format.
-ImportMT940 *SwiftFileName*	Importiert alle Umsatzdaten aus der gegebenen SWIFT MT-940 bzw. VR-NetWorld MT-940 Datei (Endungen „.sta“ oder „.940“) in die zugehörigen Konten.
-ImportCAMT *CamtFileName*	Importiert alle Umsatzdaten aus der gegebenen CAMT 052 oder CAMT 053 Datei in die zugehörigen Konten.
-ImportSEPA *SepaFileName*	Importiert die SEPA-Zahlungen aus der angegebenen SEPA-XML-Datei. Wurde ein Konto ausgewählt, dann werden die Zahlungen zusätzlich zu einem Sammelauftrag zusammengefügt.
-SubmitSEPA *SepaFileName*	Importiert die SEPA-XML-Datei und legt gleich einen entsprechenden Sammelauftrag bereit zur Übertragung in den Ausgangskorb. Hierfür muss immer ein Konto angegeben werden.
-ImportDTAZV *DtazvFileName*	Importiert die DTAZV-Zahlungen aus der angegebenen DTAZV-Datei als einzelne Zahlungen. **Anmerkung:** EU Standardüberweisungen können in nicht im Format DTAZV importiert werden. Verwenden Sie statt dessen das SEPA oder das SUPA Format mit Zahlungsdienst SEPA.
-ImportPaymts *FileName*	Importiert die Zahlungen aus der gegebenen SUPA-Datei in das gewählte Konto. Ein Zielkonto muss ausgewählt sein, andernfalls schlägt dieses Kommando fehl. Die einzelnen Zahlungen werden nur als erwartete Zahlungen importiert. Es wird kein Sammelauftrag erstellt.

Kommando	Beschreibung
-SendRecv *SendRecvFlags*	Nachdem alle angeforderten Import-Kommandos verarbeitet wurden, wird ein Senden/Empfangen ausgelöst. Die als Argument übergebenen *SendRecvFlags* bestimmen, welche Daten hierbei übertragen werden. Das Argument ist eine kommaseparierte Liste folgender Schlüsselwörter: **Rundruf** – Alle für den Rundruf eingestellten Daten werden abgeholt, oder **Balances** – Der aktuelle Online-Kontostand wird abgeholt **Statements** – Alle neuen Kontoumsätze werden abgerufen **StandingOrders** – Der Dauerauftragsbestand wird abgerufen **PostdatedRemitts** – Der Bestand terminierter Überweisungen wird abgerufen **Portfolio** – Die Depotaufstellung wird abgerufen **Assets** – Alle Festgeldanlagen werden abgerufen **Files** – Alle vorliegenden Dokumente werden abgerufen **PayChecks** – Alle vorliegenden VEU.Aufträge werden abgerufen Wurde über die Parameter ein Konto festgelegt, so werden die Daten nur für dieses Konto übertragen. Wurde kein Konto gewählt, so werden diese Daten für alle vorhandenen Konten (sofern zutreffend) übertragen. Wird das Schlüsselwort **Rundruf** übergeben, so darf kein anderes Schlüsselwort mit übergeben werden.
-ExportAccts *SupaFileName*	Exportiert eine Liste aller eingerichteten Konten im SUPA Dateiformat. Die Dateiendung (.json/.csv/.supa) bestimmt das generierte SUPA-Format.
-ExportStmt *SupaFileName*	Exportiert alle Umsatzdaten im SUPA Dateiformat für das gewählte Konto und den gewählten Zeitraum. Die Dateiendung (.json/.csv/.supa) bestimmt das generierte SUPA-Format.
-ExportMT940 *SwiftFileName*	Exportiert alle Umsatzdaten im SWIFT MT-940 Dateiformat für das gewählte Konto und den gewählten Zeitraum.
-ExportCAMT *CamtFileName*	Exportiert alle Umsatzdaten im CAMT 052 Dateiformat für das gewählte Konto und den gewählten Zeitraum.

Kommando	Beschreibung
-ExportPaymts *SupaFileName*	Exportiert alle selbst erstellten Zahlungen eines Konto in eine SUPA-Datei. Die Dateiendung (.json/.csv/.supa) bestimmt das generierte SUPA-Format.
-ExportDocList *SupaFileName*	Exportiert eine Datei mit den Meta-Daten aller vorliegenden Dokumente für die gewählten Konten in eine SUPA-Datei. Die Parameter -ExportFrom und -ExportTo können verwendet werden, um den Zeitbereich einzuschränken. Die Dateiendung (.json/.csv/.supa) bestimmt das generierte SUPA-Format.

Kommando	Beschreibung
-ExportDocuments *DirectoryName*	Exportiert alle Dokumente in das angegebene Verzeichnis. Werden keine Konten spezifiziert, so werden die Dokumente aus allen Konten exportiert. Werden Konten spezifiziert, so werden nur Dokumente aus diesen Konten exportiert. Die Parameter -ExportFrom und -ExportTo können verwendet werden, um den Zeitbereich der zu exportierenden Dokumente einzuschränken. Zusätzlich kann der Parameter -ExportIds verwendet werden, um gezielt Dokumente anhand ihrer ID zu exportieren. Ist die Option -ExportOverwrite angegeben, so werden gleichnamige, bereits existierende Dateien überschrieben. Ist die Option nicht angegeben, so wird beim Export immer ein eindeutiger Name generiert. Ist die Option -ExportPerAcctFolder angegeben, so wird unterhalb des angegebenen *DirectoryName* für jedes Konto, aus dem Dokumente exportiert werden, ein eigener Unterordner angelegt. Als Name für den Unterordner wird die jeweilige Kontonummer aus den Kontostammdaten verwendet. Ist die Option -ExportSwiftMerge angegeben und werden in einem Aufruf mehrere SWIFT-Dokumente exportiert, so werden alle im Export enthaltenen SWIFT-Dokumente des gleichen Typs in einer Exportdatei zusammengefasst. Ist gleichzeitig die Option -ExportPerAcctFolder angegeben, so werden gleichartige SWIFT-Dokumente je Konto zusammengefasst. Zusammenfassen bedeutet, dass die SWIFT-Dokumente in chronologischer Reihenfolge hintereinander in eine Datei geschrieben werden. Als Dateiname für die zusammengefasste Datei wird ein sekundengenauer Zeitstempel plus dem Dokumentennamen des ersten SWIFT-Dokuments verwendet, zum Beispiel: 20210319152211_Depotaufstellung.mt535

3.4 Rückgabewerte

Bei der Rückkehr setzt BankingZV den „Exit Code“ des Prozesses, um die erfolgreiche Verarbeitung oder einen Fehler anzuzeigen. Wurden alle angeforderten Kommandos erfolgreich ausgeführt, dann wird der Rückgabewert (Exit Code) auf Null gesetzt. Tritt ein Fehler auf, so wird ein von Null verschiedener Rückgabewert entsprechend der folgenden Tabelle gesetzt.

Exit Code	Beschreibung
0	TopBanking.exe hat alle Kommandos erfolgreich ausgeführt.
1000	Der Anwender hat ein Kommando manuell abgebrochen. Zum Beispiel bei der PIN-Eingabe. Die folgenden Kommandos wurden nicht ausgeführt.
9000	Von einer Bank wurde bei der Übertragung ein Fehlercode empfangen.
10000	Die übergebenen Parameter sind ungültig. Überprüfen Sie die Kommandozeile.
10001	Der gewählte Datentresor konnte nicht selektiert werden. Prüfen Sie, ob der angegebene Datentresor eine korrekte Datentresordatei ist und beschreibbar ist.
10002	Der gewählte Datentresor konnte nicht geöffnet werden. Vermutlich war das übergebene Passwort falsch.
10003	Der gewählte Datentresor existiert nicht und konnte nicht angelegt werden. Überprüfen Sie, ob der angegebene Ort beschreibbar ist.
10004	Die übergebene Importdatei war ungültig.
10005	Es liegt keine Lizenz für den Kommandozeilenmodus vor. Für die Nutzung des Kommandozeilenmodus ist eine BankingZV Lizenz erforderlich.
20000	Ein interner Fehler ist aufgetreten.

4 Integration Subsembly EBICS API

BankingZV kann nahtlos mit der Subsembly EBICS API und dem EBICS Spooler integriert werden. So können in BankingZV direkt die in der Subsembly EBICS API mit dem EbicsAdmin angelegte EBICS Bankzugänge genutzt werden. Auch eine Zusammenarbeit mit dem EBICS Spooler aus der Subsembly EBICS API ist möglich.

4.1 Externe EBICS Bankzugänge

Wurden im System EBICS Bankzugänge mit der Subsembly EBICS API angelegt, so erscheinen diese automatisch auch in BankingZV in der Ansicht **Online-Banking Einstellungen** und werden dort mit dem Zusatz **(extern)** angezeigt. Hierfür ist keine weitere Konfiguration erforderlich, alle im EbicsAdmin der Subsembly EBICS API angezeigten Bankzugänge können sofort von Bankkonten in BankingZV für das Online-Banking genutzt werden.

Änderungen, die am EBICS Bankzugang in BankingZV vorgenommen werden, wirken sich auch auf den von der Subsembly EBICS API genutzten EBICS Bankzugang aus, da es sich um ein und dieselbe Speicherdatei handelt.

Ein extern mit der Subsembly EBICS API gespeicherter EBICS Bankzugang steht in allen Datentresoren gleichermaßen zur Verfügung. Um Konflikte zu vermeiden, ist deshalb auf eine möglichst klare und eindeutige Namensvergabe zu achten.

4.1.1 SUBSEMBLY_EBICS_CONTACTSFOLDERPATH

Die Subsembly EBICS API speichert die Bankzugänge standardmäßig im Windows-Anwenderverzeichnis

%APPDATA%\Subsembly\EBICS

Für jeden EBICS Bankzugang wird dort eine XML-Datei mit den Zugangsdaten geführt. Für eine Benutzer- und Arbeitsplatzübergreifende Integration kann über die Umgebungsvariable

SUBSEMBLY_EBICS_CONTACTSFOLDERPATH

ein anderer Speicherordner festgelegt werden. Diese könnte zum Beispiel auf ein Netzverzeichnis verweisen, auf das von mehreren Arbeitsstationen zugegriffen werden kann.

Bitte achten Sie darauf, einen Speicherordner mit ausreichendem Zugriffsschutz und idealerweise Verschlüsselung zu verwenden.

4.1.2 Arbeiten mit externen EBICS Bankzugängen

BankingZV stellt verschiedene Funktionen für das Arbeiten mit externen EBICS Bankzugängen zur Verfügung. Diese befinden sich alle im Kontextmenü in der Ansicht **Online-Banking Einstellungen**. Folgende Menüpunkte werden angeboten.

4.1.2.1 EBICS Bankzugang auslagern

Über diesen Menüpunkt kann ein in BankingZV angelegter und im Datentresor gespeicherter EBICS Bankzugang komplett in den Standardspeicherordner der Subsembly EBICS API ausgelagert werden. Die im Datentresor gespeicherten Bankzugangsdaten werden vollständig gelöscht. Konten, welche mit diesem Bankzugang verknüpft waren, werden automatisch mit dem ausgelagerten Bankzugang verknüpft und funktionieren deshalb weiterhin.

Nach der Auslagerung erscheint der EBICS Bankzugang auch im EbicsAdmin der Subsembly EBICS API und kann dort von anderen Anwendungen, welche auf der Subsembly EBICS API basieren, verwendet werden.

4.1.2.2 EBICS Bankzugang integrieren

Ein im EbicsAdmin der Subsembly EBICS API angelegter EBICS Bankzugang wird in BankingZV in der Ansicht **Online-Banking Einstellungen** mit dem Zusatz **(extern)** angezeigt. Ein solcher Bankzugang kann über diesen Menüpunkt in den Datentresor integriert und aus dem Standardspeicherordner der Subsembly EBICS API gelöscht werden. Anschließend steht dieser Bankzugang nur mehr in dem Datentresor zur Verfügung, in dem er integriert wurde. Für externe Anwendungen der Subsembly EBICA API und aus anderen Datentresoren steht der Bankzugang dann nicht mehr zur Verfügung.

Eine vom Bankzugang referenzierte Schlüsseldatei wird nicht in den Datentresor integriert. Der integrierte Bankzugang benötigt weiterhin die extern gespeicherte Schlüsseldatei.

4.1.2.3 EBICS Bankzugang exportieren

Die Daten aus einem Datentresor-mit internem EBICS Bankzugang können über diesen Menüpunkt in eine XML-Datei exportiert werden. Der Bankzugang im Datentresor bleibt hierbei unverändert erhalten. Die exportierte XML-Datei ist kompatibel mit der Subsembly EBICS API und kann von dieser direkt zum Erstellen von EbicsContact Objekten verwendet werden.

4.1.2.4 EBICS Bankzugang importieren

Liegt eine mit der Subsembly EBICS API oder dem EbicsAdmin erzeugte XML-Datei eines EbicsContact Objekts vor, so kann diese in den Datentresor importiert werden. Da alle im Standardordner der Subsembly EBICS API gespeicherten EBICS Bankzugänge sowieso direkt in BankingZV verfügbar sind, ist dies nur für individuell, separat gespeicherte Bankzugangsdaten sinnvoll. Die ausgewählte Datei bleibt beim Import unverändert.

Über die Menüpunkte "EBICS Bankzugang exportieren" und "EBICS Bankzugang importieren" kann ein EBICS Bankzugang in Form einer Datei auch von einem System auf ein anderes System übertragen werden. Vergessen Sie hierbei nicht, auch die evtl. erforderliche Schlüsseldatei ebenfalls auf das Zielsystem zu übertragen.

4.2 Externe CAMT Buchungsdateien

Über den EBICS Spooler Mechanismus der Subsembly EBICS API können die via EBICS abgerufenen Kontoumsatzdateien und Buchungsvormerkungsdateien auch in externen Verzeichnissen gespeichert werden. Ist ein externes EBICS Spooler-Verzeichnis konfiguriert, so steht in den Einstellungen des EBICS Bankzugangs in BankingZV die Option "CAMT Dateien im Spooler-Verzeichnis ablegen" zur Verfügung. Wird diese Option für einen EBICS Bankzugang gesetzt, so werden automatisch alle abgerufenen Kontoumsatzdateien und Buchungsvormerkungsdateien darin zwischengespeichert.

Eine Buchhaltungssoftware könnte nun auf diese externen Buchungsdateien zugreifen und die Buchungen daraus automatisch importieren und verarbeiten.

4.2.1 SUBSEMBLY_EBICS_SPOOLERFOLDERPATH

Nur wenn eine Umgebungsvariable mit dem Namen SUBSEMBLY_EBICS_SPOOLERFOLDERPATH auf ein existierendes Verzeichnis verweist, kann die EBICS Spooler-Verzeichnis Integration in BankingZV aktiviert werden.

Um diese Umgebungsvariable zu konfigurieren und den Umsatzabruf über dieses Verzeichnis zu aktivieren, sind folgende Konfigurationsschritte und Einstellungen erforderlich:

- Sie verwenden BankingZV mindestens in der Version 7.6.3. Prüfen Sie die aktuell verwendete Version über den Menüpunkt **BankingZV** > **Updates** und nehmen ggf. ein Update vor.
- Rufen Sie in den Windows-Einstellungen die **Erweiterten Systemeinstellungen** auf und legen Sie bei **Umgebungsvariablen...** eine neue Benutzervariable

  SUBSEMBLY_EBICS_SPOOLERFOLDERPATH

  an. Als Wert tragen Sie den kompletten Dateipfad zum gewünschten Speicherordner ein. Übernehmen Sie die Systemeinstellungen mit OK.
- Anschließend starten Sie BankingZV neu und wechseln Sie in die Ansicht **Online-Banking Einstellungen**. Öffnen Ihren EBICS Bankzugang zum Bearbeiten und setzen beim Punkt **Integration EBICS Spooler** das Häkchen bei **CAMT-Dateien im Spoolerverzeichnis ablegen** (im Formular ganz unten).

Beim Abruf werden die CAMT Dateien unterhalb des konfigurierten Verzeichnisses in Unterverzeichnissen wie folgt angelegt.


```
%SUBSEMBLY_EBICS_SPOOLERFOLDERPATH%
    <HostID>
        <PartnerID>
            "camt052"
            "camt053"
```

Unter %SUBSEMBLY_EBICS_SPOOLERFOLDERPATH% wird je ein Verzeichnis benannt mit der Host-ID des abgerufenen EBICS Bankzugangs angelegt. Darunter wird ein weiteres Verzeichnis, benannt mit der Partner-ID (bzw. Kunden-ID) des abgerufenen EBICS Bankzugangs, angelegt. Die abgerufenen Buchungsvormerkungen werden darin im Unterordner "camt052" gespeichert, die Kontoumsatzdateien im Unterordner "camt053".

Die Namensgebung der gespeicherten Dateien folgt den EBICS Konventionen für Datencontainer:

```
JJJJ-MM-TT_CCC_KKK…KKK_WWW_AAA…AAA.xml
```

JJJJ-MM-TT	Erstellungsdatum der Originaldatei (nicht das Erstellungsdatum der Datei im lokalen Dateisystem)
CCC	Datenformat der Datei. Im Ordner camt052 ist dies immer C52, im Ordner camt053 ist dies immer C53.
KKK…KKK	Die Kontoidentifikation, normalerweise IBAN. Ist für das Konto keine IBAN vorhanden, wird stattdessen ein 11-stelliger BIC (8-stellige BICs werden durch „XXX“ rechtsbündig ergänzt) bzw. die 8-stellige deutsche Bankleitzahl, jeweils gefolgt von einem Punkt "." gefolgt von der (nationalen) Kontonummer verwendet.
WWW	Die Kontowährung gemäß ISO 4217.
AAA…AAA	Zusätzliche ID um einen eindeutigen Dateinamen zu erhalten. Bei C53 Dateien basiert diese ID auf der Kontoauszugsnummer, bei C52 Dateien wird ein Zeitstempel herangezogen.

Muster für camt053 Dateinamen:

Für Konto mit IBAN:

```
2021-01-08_C53_DE87200500001234567890_EUR_000001.xml
```

Für deutsche Kontonummer mit Bankleitzahl:

```
2021-01-08_C53_20050000.1234567890_EUR_000001.xml
```

Für deutsche Kontonummer mit BIC:


```
2021-01-08_C53_BANKDEFF123.1234567890_EUR_000001.xml
```

Die heruntergeladenen und im Spooler-Verzeichnis gespeicherten CAMT-Dateien werden niemals automatisch gelöscht.

Der Vorteil dieser Vorgehensweise ist, dass die Umsatz-Daten für eine bestimmte EBICS Partner-ID (bzw. Kunden-ID) systemweit nur einmal heruntergeladen werden müssen und auf die so gespeicherten CAMT-Dateien von verschiedenen Datentresoren und anderen Subsembly EBICS API Anwendungen zugegriffen werden kann.

Bitte achten Sie darauf, einen Speicherordner mit ausreichendem Zugriffsschutz und idealerweise Verschlüsselung zu verwenden.

4.2.2 Auftragsarten und Konvertierung

Eine ganz besondere Eigenschaft des Spooler-Verzeichnisses ist, dass die Dateien immer im CAMT 052 und CAMT 053 Datenformat gespeichert werden, selbst dann, wenn sie via EBICS in einem anderen Datenformat abgerufen wurden.

Werden Kontoumsätze beispielsweise im Format SWIFT MT-940 mit der Auftragsart "STA" abgerufen, so werden diese vor der Speicherung im Spooler-Verzeichnis in ein oder mehrere Dateien im CAMT 053 Format gespeichert. Ebenso werden im Format SWIFT MT-942 mit der Auftragsart "VMK" abgerufene Vormerkposten konvertiert und im Format CAMT 052 gespeichert.

Eine externe Anwendung kann diese Daten also einheitlich im entsprechenden CAMT Format einlesen und benötigt keine eigene Konvertierung.

Die zu verwendende Auftragsart kann in BankingZV in den Einstellungen des EBICS Bankzugangs gewählt werden. Wenn möglich, ist hier natürlich immer "C52" und "C53" zu bevorzugen.

Wird ein externer EBICS Bankzugang verwendet, so wirkt sich eine hier durchgeführte Einstellung auch auf den EbicsSpooler der Subsembly EBICS API aus.

4.3 EBICS Spooler

In der Subsembly EBICS API ist ein Programm EbicsSpooler enthalten, welches den automatisierten Abruf von Kontoumsatzdateien und Vormerkungsdateien via EBICS ermöglicht. Werden in BankingZV externe EBICS Bankzugänge und externe CAMT Buchungsdateien, so wie oben beschrieben, verwendet, dann aktualisiert der EbicsSpooler aus der Subsembly EBICS API auch die von BankingZV genutzten externen CAMT Buchungsdateien. Über den EbicsAdmin kann ein Windows Task Scheduler Task (Computerverwaltung / Aufgabenplanung) eingerichtet werden, welcher den EbicsSpooler in regelmäßigen Abständen automatisch ausführt.

Für die Nutzung des EbicsSpoolers ist eine separate Lizenz der Subsembly EBICS API, unabhängig von der BankingZV Lizenz erforderlich.

5 REST-Schnittstelle

5.1 Einführung

Buchhaltungssoftware ist häufig mit der Anforderung konfrontiert, den Zahlungsverkehr über Bankkonten eines Unternehmens direkt zu integrieren. Einfache, manuelle Export/Import-Schnittstellen sind arbeitsaufwändig in der Bedienung und führen leicht zu Fehlern in der Benutzung.

Wird die Buchhaltungssoftware vom Anbieter als Cloud-Service angeboten, ergeben sich bei einer direkten Integration des Kontozugriffs auf Serverseite folgende Problemstellungen für den Anbieter:

- Die Anbindung von FinTS oder PSD2 Schnittstellen erfordert eine aufwändige und teure Zertifizierung des Anbieters als Kontoinformations- und Zahlungsauslösedienst durch die BaFin.
- Der Anwender muss bei jeder über FinTS/PSD2 angebundenen Bank mindestens alle 90 Tage eine manuelle 2-Faktor-Authentifizierung durchführen.
- Der Anbieter der Buchhaltungssoftware muss die vielen verschiedenen Bankschnittstellen unterstützen und die Endanwender entsprechend supporten können.

Eine einfache Möglichkeit, diese Probleme zu umgehen und dem Nutzer dennoch eine komfortable Anbindung seiner Bankkonten zu ermöglichen, ist die Anbindung von BankingZV als Datenquelle für Kontoumsatzdaten und als Instrument zur Ausführung von Zahlungen aller Art.

BankingZV stellt hierzu eine konfigurierbare REST-Schnittstelle zur Anbindung von externer, Cloud basierter Buchhaltungssoftware zur Verfügung. BankingZV agiert hierbei als REST Client. Der Buchhaltungsdienst stellt einen dazu passende REST Service bereit.

Über diese REST Schnittstelle werden folgende Informationen mit dem Buchhaltungssystem ausgetauscht.

- Neue Kontoumsätze werden automatisch von BankingZV zum REST Service hochgeladen.

- Neue, zur Zahlung anstehende Überweisungen und Lastschriften werden von BankingZV vom REST Service regelmäßig abgeholt und als auszuführende Aufträge in BankingZV importiert.

Der Aufbau der bankfachlichen Daten folgt der SUPA Spezifikation (Subsembly Payments Dateiformat, siehe: https://subsembly.com/supa.html), die inhaltlich auf den ISO 20022 und den SEPA XML Datenformaten basiert.

5.2 Authentifizierung

Sowohl BankingZV als auch der Benutzer muss sich gegenüber dem REST Service authentifizieren. Es muss gewährleistet sein, dass nur ein authentifizierter und autorisierter Nutzer die Dienste der REST-Schnittstelle nutzen kann.

5.2.1 API Token

In allen Requests wird im Request-Header der Anfrage im Feld "Authorization" ein von Subsembly für den Betreiber erzeugter "API Token" als "Bearer"-Token eingestellt.

Der API Token ist ein JWT Token mit HS256 Signatur mit folgendem Payload:

Feldname	Typ		Beschreibung
iss	string	M	Issuer - Der Herausgeber muss dem Internet Hostnamen des Servers entsprechen.
sub	string	M	Subject - Das Token-Subject ist die Basis-URL unter welcher die Endpoints des Buchhaltungssystems erreichbar sind.
iat	int	M	Issued At - Zeitstempel der Tokenerstellung.
exp	int	O	Expiry - Zeitpunkt zu dem dieser Token ungültig wird.

Der API Token wird normalerweise vom Betreiber seinen Anwendern als Datei zur Verfügung gestellt. Dieser kann in BankingZV geladen werden und wird dann im HTTP-Header bei allen REST Requests als Bearer-Token eingestellt.

5.2.2 Authentifizierung über UserToken

Für alle Requests wird ein UserToken benötigt, welcher den Anwender gegenüber dem Buchhaltungssystem eindeutig identifiziert und authentifiziert. Der UserToken wird im Buchhaltungssystem erzeugt und muss vom Anwender in BankingZV eingetragen werden.

5.3 REST Endpoints

5.3.1 Kontoinformationen und Zahlungsaufträge

`POST /adviseAcct`

Lädt Kontoinformationen (inkl. Salden) gemäß SUPA Spezifikation hoch. Dabei kann ein nicht vorhandenes Konto im Server angelegt, oder das Konto ignoriert werden. In der Antwort werden das Datum zurückgeliefert, ab dem Umsätze hochgeladen werden sollen, sowie Zahlungsaufträge, die für dieses Konto in BankingZV importiert werden sollen.

5.3.1.1 Request Body

Parameter	Type	Description
UserToken	string	Token zur Authentifizierung des Anwenders
Acct	Acct	Konto im SUPA JSON Format mit Salden.
RequestPaymts	bool	

Beispielrequest:


```
{
    "UserToken": "lalala",
    "Acct": {
        "Id": "87618185228262427",
        "AcctBankCode": "12030000",
        "AcctNo": "1234567890",
        "AcctBIC": "BYLADEM1001",
        "AcctIBAN": "DE00120300001234567890",
        "AcctNm": "Privatgiro",
        "AcctCtry": "DE",
        "AcctTpCd": "CACC",
        "AcctCcy": "EUR",
        "OwnrNm": "Max Mustermann",
        "BalAmt": "100.01",
        "BalCdtDbtInd": "CRDT",
        "BalDt": "2021-04-09",
        "CurBalAmt": "100.01",
        "CurBalCdtDbtInd": "CRDT",
        "CurBalDt": "2021-04-09",
        "AvlAmt": 100.01,
        "CdtLineAmt": 100.01
    }
}
```


5.3.1.2 Response Body (Success 200)

Parameter	Type	Description
Status	string	Gibt an, ob für das Konto Umsatzdaten hochgeladen werden sollen, oder nicht.
AcctInfo	AcctInfo	Kontozuordnung und kontobezogenes Startdatum ab dem Umsätze benötigt werden.
PaymtsZip	string	Optional eine gezippte SUPA Datei mit Zahlungsaufträgen die für dieses Konto importiert werden sollen.

Beispielantwort:


```
{
    "Status": "YAY",
    "AcctInfo": {
        "ClientAcctId": "23187672361768",
        "ServerAcctId": "78789452111212",
        "DateFrom": "2018-11-23"
```



```
    },
    "PaymtsZip": "jghjhjhg"
}
```

5.3.1.3 AcctInfo Objekt

Beinhaltet die Kontozuordnung zwischen Client und Server sowie das Startdatum, ab dem Umsätze benötigt werden.

Parameter	Type	Description
ClientAcctId	string	Clientseitige ID des Kontos
ServerAcctId	string	Serverseitige ID des Kontos
DateFrom	ISODate	Datum, ab dem die Umsätze benötigt werden.

5.3.2 Umsätze

`POST /uploadStatementEntries`

Upload der Kontoumsätze im SUPA Format (ZIP komprimiert) für ein Konto.

5.3.2.1 Request Body

Parameter	Type	Description
UserToken		
NtrysZip	string	Gezippte Liste der Umsätze im SUPA Format. Base64 encoded.

Beispielrequest:

```
{
    "UserToken": "kkjkjkjhkjh"
    "NtrysZip": "string"
}
```

5.3.2.2 Response Body (Success 200)

Ein erfolgreicher Upload der Umsatzdaten wird durch den Http Code 200 signalisiert.

5.4 BankingZV Benutzerschnittstelle

In BankingZV wird ein Dialogfenster zur Erfassung des API-Tokens und des Anwendertokens bereitgestellt.