SUBSEMBLY

Subsembly GmbH
Hofmannstr. 7 b
81379 München

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

26. April 2022

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.
- 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. Diese Abbildungsvorschriften werden in einer eigenen CSI-Datei gespeichert.

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.

Wichtiger Hinweis!	
	Der Import von beliebigen CSV-Dateien über CSI-Dateien mit Abbildungsvorschriften ist nicht nur mit BankingZV, sondern auch mit Banking4 möglich. Der Export von CSV-Dateien über CSI-Dateien mit entsprechenden Abbildungsvorschriften wird jedoch ausschließlich von BankingZV unterstützt.

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 und im lokalen AppData-Ordner

C:\Users\BENUTZER\AppData\Local\Subsembly\TopBankingZV\ImportCsi

gespeichert sein.

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.
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.

Feldname	Typ	Len		Beschreibung
decimalpoint	string	1	O	Dezimaltrennzeichen. Üblicherweise ist das ein Komma "," oder ein Punkt ".". Wird keines angegeben, so wird "," angenommen.
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":
  [
    { /* ... */ },
    /* ... */
  ]
}
```

Wichtiger Hinweis!	
	Der `identifier` muss zwingend eineindeutig sein. Stellen Sie also sicher, dass Sie eine ID nur einmalig vergeben, um darüber ein Dateiformat zu identifizieren.

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.

- Alternativ zu "CdtDbtInd" kann auch die Spaltenbezeichnung "_SollHaben" verwendet werden. Diese erwartet inhaltlich ein "S" für eine "Soll"-Buchung (Debit) und ein "H" für eine "Haben"-Buchung (Credit).

- Zusätzlich kann eine Spalte "_FeeAmt" für einen Gebührenbetrag festgelegt werden. Der darin enthaltene Betrag wird vom Buchungsbetrag zusätzlich abgezogen.

- 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
  -ExportFormat FormatId
  -ImportFormat FormatId
  -ImportStmt FileName
  -ImportMT940 MT940FileName
  -ImportCAMT CamtFileName
  -ImportSEPA SepaFileName
  -SubmitSEPA SepaFileName
  -ImportDTAZV DtazvFileName
  -ImportPaymts FileName
  -SendRecv SendRecvFlag,SendRecvFlag,...
  -ExportAccts FileName
  -ExportStmt FileName
  -ExportMT940 MT940FileName
  -ExportCAMT CamtFileName
  -ExportPaymts FileName
  -ExportDocList FileName
  -ExportDocuments DirectoryName
  -ExportOutboxDocuments 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.

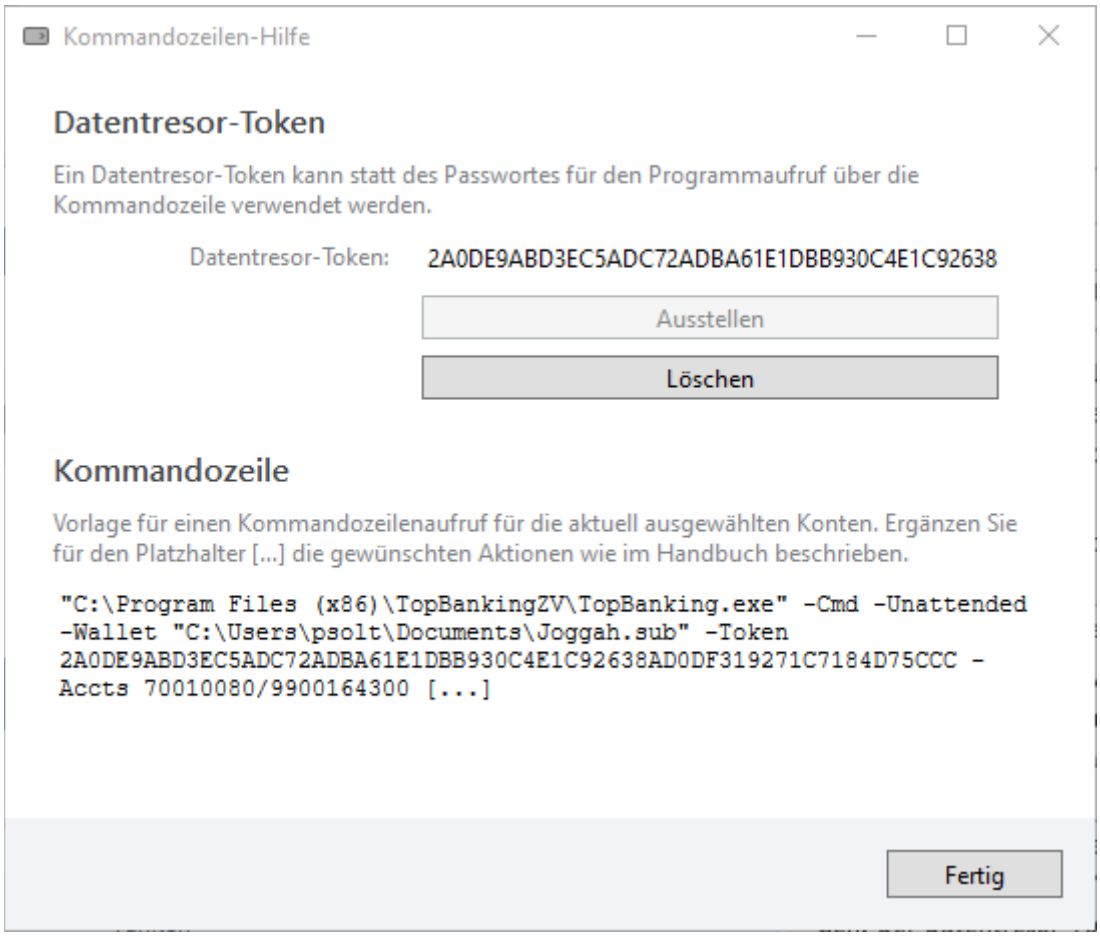


Abbildung 1: Kommandozeilen-Hilfe

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.
-ImportFormat *FormatId*	Wählt optional einen speziellen Import-Filter für die Aktionen **-ImportStmt** und **-ImportPaymts**. Die FormatId ist die im Programm im interaktiven Import-Dialog angezeigte Formatkennung.

Parameter	Beschreibung
-ExportFormat *FormatId*	Wählt optional einen speziellen Export-Filter für die Aktionen **-ExportAccts**, **-ExportStmt**, **-ExportPaymts** und **-ExportDocList**. Die FormatId ist die im Programm im interaktiven Export-Dialog angezeigte Formatkennung.
-ExportFrom *FromDate*	Anfangsdatum für einen Export von Umsatzdaten (JJJJ-MM-TT). Wird kein Anfangsdatum übergeben, so beginnt der Umsatzdatenexport mit der ersten vorhandenen Buchung.
-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**.

Parameter	Beschreibung
-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**.
-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 *FileName*	Importiert alle Buchungen aus der genannten Datei in das in den Parametern angegebene Konto. Ist kein explizites **-ImportFormat** angegeben, so bestimmt die Dateiendung (.json/.csv/.supa) 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.

Kommando	Beschreibung
-ImportPaymts *FileName*	Importiert die Zahlungen aus der gegebenen Datei in das gewählte Konto. Ist kein explizites **-ImportFormat** angegeben, so bestimmt die Dateiendung (.json/.csv/.supa) das erwartete SUPA-Format. Ein Zielkonto muss ausgewählt sein, andernfalls schlägt dieses Kommando fehl. Die einzelnen Zahlungen werden nur als erwartete Zahlungen importiert. Es wird kein Sammelauftrag erstellt.
-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 *FileName*	Exportiert eine Liste aller eingerichteten Konten in die angegebene Datei. Ist kein explizites **-ExportFormat** angegeben, so bestimmt die Dateiendung (.json/.csv/.supa) das generierte SUPA-Format.

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

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

Kommando	Beschreibung
-ExportOutboxDocuments *DirectoryName*	Exportiert für jeden ausgeführten oder fehlgeschlagenen Auftrag aus dem Ausgangskorb eine PDF-Datei in das angegebene Verzeichnis. Das PDF entspricht genau dem Ausdruck, wie er interaktiv im Programm erzeugt werden kann. Der Dateiname wird aus dem Datum und Zeitpunkt der Auftragserfassung entsprechend folgender Schablone gebildet: *JJJJMMTTHHMMSS_XXX_IBAN_OID*.pdf XXX ist das Kürzel für die Auftragsart. IBAN ist die IBAN des Auftraggeberkontos, sofern verfügbar. OID ist die interne ID des Auftrags und garantiert letztendlich die Eindeutigkeit des Dateinamens innerhalb eines Datentresors. Die Parameter -ExportFrom und -ExportTo können verwendet werden, um den Zeitbereich der zu exportierenden Dokumente einzuschränken.

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.

Exit Code	Beschreibung
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 direkt die in der Subsembly EBICS API mit dem EbicsAdmin angelegten EBICS Bankzugänge nutzen, sogenannte **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 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.

4.2 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.

4.2.1 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.

4.2.2 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 EBICS 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.

4.2.3 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.

4.2.4 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.

5 Integration mit Subsembly EBICS Spooler

In BankingZV können Spoolerordner des Subsembly EBICS Spoolers als Datenquelle und Ausgangskorb für Aufträge eingerichtet werden.

Wichtiger Hinweis!	
	Für die Integration mit dem Subsembly EBICS Spooler ist eine **BankingZV Firmenlizenz** erforderlich.

Über den Menüpunkt

```
Extras > Mit Spoolerordner verbinden ...
```

kann ein Spoolerordner für einen EBICS Host und EBICS Partner als Bankzugang angelegt werden. Beim Abruf über diesen Bankzugang werden dann lediglich die in den Abrufordnern bereitgestellten Dateien importiert. Werden über diesen Bankzugang Aufträge versendet, so werden diese in den Spoolerordner `outbox` kopiert. Der eigentliche EBICS Sende- und Empfangsvorgang erfolgt separat durch den EBICS Spooler.

Weitere Informationen finden Sie in der Dokumentation zum Subsembly EBICS Spooler.

6 REST-Schnittstelle

Wichtiger Hinweis!	
	Für die Anbindung eines Buchhaltungsdienstes über die hier definierte REST-Schnittstelle ist eine **BankingZV Firmenlizenz** erforderlich.

6.1 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 Buchhaltungsdienst 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 ISO 20022 und den SEPA XML Datenformaten basiert.

6.2 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.

6.2.1 API Token

In allen Requests wird im Request-Header der Anfrage im Feld "Authorization" ein "API Token" als "Bearer"-Token eingestellt. Dieser API Token ist ein JWT Token nach RFC 7519. Siehe auch https://jwt.io/ für weitere Informationen zu JSON Web Tokens.

Um zu verhindern, dass durch einen betrügerisch ausgestelltes API Token Daten aus BankingZV an ein nicht autorisiertes System übertragen werden, werden alle API Token ausschließlich durch die Subsembly GmbH ausgestellt und RS256 signiert. Sowohl BankingZV als auch der Buchhaltungsdienst müssen diese Signatur prüfen. Hierzu wird dem Betreiber des Buchhaltungsdienstes der öffentliche Schlüssel zur RS256 Signatur zur Verfügung gestellt.

Im JWT Token sind mindestens folgende Felder (Claims) im Payload enthalten:

Name	Typ		Beschreibung
iss	string	M	Issuer - Der Name des Betreibers des Buchhaltungsdienstes, zu dem dieser API Token Zugriff gewährt. Dieser wird in BankingZV dem Benutzer angezeigt.
sub	string	M	Subject - Das Token-Subject ist die Basis-URL, unter welcher die Endpoints des Buchhaltungsdienstes erreichbar sind. Diese URL darf *nicht* mit einem Schrägstrich enden. Die unten definierten Endpoints werden an diese Basis-URL mit einem Schrägstrich angehängt.
aud	string	M	Audience - Muss den Wert "BankingZV" enthalten. JWT Tokens mit einem anderen Wert werden von BankingZV nicht verarbeitet.
iat	int	M	Issued At - Zeitstempel der Tokenerstellung.
exp	int	O	Expiry - Optional ein Zeitpunkt, zu dem dieser Token ungültig wird.

Lediglich die oben genannten Felder des Payloads des JWT Tokens werden in BankingZV ausgewertet. Zusätzliche Felder (Claims) können im JWT Token enthalten sein und werden in BankingZV ignoriert.

Beispiel JWT Payload:

```
{
    "iss": "Subsembly GmbH",
    "sub": "https:\/\/finance.company.co\/connect\/bzv",
    "aud": "BankingZV",
    "exp": 1640905200,
    "iat": 1635432533
}
```

Der JWT Token wird von der Subsembly GmbH für den Betreiber des Buchhaltungsdienstes generiert und diesem zur Verfügung gestellt. Der öffentliche Schlüssel zum Prüfen der RS256 Signatur des JWT Tokens ist:

```
-----BEGIN RSA PUBLIC KEY-----
MIIBCgKCAQEAtWKzsjmZplSwIGDn++yucuNFIjqXlPzhDGZNixblr/Gu8XPZZfAT
BkaN1QA6CzYIabMv2DiSAJiLO8l45u9cGwND0Me0v8zWw92ngufz/AIqVJlQdmzA
MvRriU3fVt6RVp9oeu/zAUR6nAYAIRjFLCd3dDhOLqEV4wbVFv+BotUoVY9e1rZe
bNub9dZPpVv1iAvSNX0puC+gl3qIY5y3SFuKzH2HdFN0tKmKqR4RG23FkQ8G5vvB
NxyG7H6deKLQ1mbus0P8qih5GawgLgt82PtijUDEHTvkXakO/+L/8z+ChMDmaXLF
qz3ZPoQvSvLPg52cRKaI76ZvWxYuvy3PTwIDAQAB
-----END RSA PUBLIC KEY-----
```

Der Betreiber stellt wiederum den von der Subsembly GmbH erzeugten JWT Token seinen Anwendern als API Token in einer Datei zur Verfügung.

Der Anwender muss letztendlich den API Token in BankingZV laden, um die Verbindung zum Buchhaltungsdienst zu konfigurieren. Siehe auch 6.7 BankingZV Benutzerschnittstelle.

Ein Request ohne gültiges API Token muss vom Buchhaltungsdienst mit dem HTTP Status "401 Unauthorized" quittiert werden.

6.2.2 Authentifizierung über UserToken

Für alle Requests wird ein *Autorisierungscode* (UserToken) benötigt, welcher den Anwender gegenüber dem Buchhaltungsdienst eindeutig identifiziert und authentifiziert. Der Autorisierungscode wird im Buchhaltungsdienst erzeugt und muss vom Anwender in BankingZV eingetragen werden. Siehe auch 6.7 BankingZV Benutzerschnittstelle.

Der Autorisierungscode ist ein String, der von BankingZV nicht weiter interpretiert wird. Er wird im Datentresor gespeichert und in allen an den Buchhaltungsdienst gesendeten Requests für Konten aus diesem Datentresor mitgeliefert.

Sobald im Datentresor von BankingZV ein Autorisierungscode hinterlegt ist, werden für alle Online-Konten mit IBAN die REST-Endpoints dieser Schnittstelle aufgerufen.

6.3 Fehlerbehandlung

Erfolgreiche Anfragen werden vom Buchhaltungsdienst immer mit dem HTTP-Statuscode 200 beantwortet. Bei Fehlern wird grundsätzlich nach dem Verantwortungsbereich unterschieden. Fehler, die im Verantwortungsbereich des Clients liegen (z.B. die Angabe ungültiger Daten), werden mit dem HTTP-Code 4xx beantwortet. Serverseitige Fehler werden hingegen mit einem HTTP-Code 5xx beantwortet. Die im Fehlerfall bereitgestellte ErrorResponse beinhaltet einen applikationsspezifischen Fehlercode und eine Fehlermeldung.

6.3.1 Applikationsspezifische Fehlercodes

Liste explizit festgelegter Fehlercodes.

HTTP	Code	Type
500	FMS_SERVER_ERROR	Unerwarteter Serverfehler
401	FMS_INVALID_API_TOKEN	Der im Header übergebene API-Token ist ungültig.
403	FMS_INVALID_USER_TOKEN	Der empfangene UserToken ist nicht gültig.
400	FMS_INVALID_REQUEST	Der Request enthält zwar ein gültiges JSON, der Inhalt des JSON ist aber nicht konform zu dieser Spezifikation.

6.4 Datenobjekte

In den REST Requests und Responses werden folgende allgemeine JSON Datenobjekte verwendet. Darüber hinaus basieren alle fachlichen Objekte auf der aktuellen SUPA Spezifikation, siehe: https://subsembly.com/download/SUPA.pdf.

Wichtiger Hinweis!	
	Die JSON Feldnamen für alle hier definierten Objekte beginnen mit einem Kleinbuchstaben. Die Feldnamen der SUPA Objekte beginnen mit einem Großbuchstaben.

In der Spalte **U/D** ist jeweils festgelegt in welche Übertragungsrichtung das Element jeweils vorkommen darf. **U** steht für Upload von BankingZV zum Buchhaltungsdienst. **D** steht für Download vom Buchhaltungsdienst zu BankingZV.

6.4.1 ErrorResponse

Fehlerobjekt, das bei den HTTP-Codes 4xx und 5xx zurückgeliefert wird und neben einem applikationsspezifischen Code noch eine Fehlernachricht beinhaltet.

Name	Typ	U/D	Beschreibung
code	string	D	*Verpflichtend*. Applikationsspezifischer Fehlercode
message	string	D	*Verpflichtend*. Fehlertext, der in BankingZV dem Anwender angezeigt werden soll.
diagnostic	string	D	*Optional*. Diagnosetext, der in BankingZV nur im Übertragungsprotokoll für die Fehlerdiagnose durch den Support aufgenommen wird. Dieser Text wird dem Anwender nicht angezeigt.

6.4.2 PaymtsInfo Objekt

Datenblock mit Zahlungsdateien in einem bestimmten Datenformat.

Name	Typ	U/D	Beschreibung
paymtsId	string	U/D	*Verpflichtend*. Vom Server frei wählbare ID für diesen Datenblock.
paymtsStatus	string	U	*Verpflichtend*. Nur folgende Werte sind möglich: OK – Die empfangenen Zahlungen wurden komplett in BankingZV importiert. FAILED – Die empfangene Datei konnte nicht verarbeitet werden. DUPLICATE – Die empfangene `paymtsId` wurde bereits verarbeitet.
paymtsFormat	string	D	*Verpflichtend*. Datenformat des Inhalts der nachfolgenden ZIP-Datei. Es sind folgende Werte möglich: `supa.csv` – CSV Datei gemäß SUPA Spezifikation. `supa.json` – JSON Datei gemäß SUPA Spezifikation. `pain.001` – ISO 20022 Datei mit Überweisungen gemäß "pain.001.001.03". `pain.008` – ISO 20022 Datei mit Lastschriften gemäß "pain.008.001.02".
paymtsZip	string	D	*Verpflichtend*. Base-64 codierte ZIP-Datei mit Zahlungsaufträgen, die für ein Konto importiert werden sollen. Die ZIP-Datei darf nur eine einzige Datei enthalten. Sind mehrere Dateien enthalten, dann wird die ZIP-Datei komplett abgelehnt. Die enthaltene Datei muss dem oben angegebenen `paymtsFormat` entsprechen.

6.4.3 NtrysInfo Objekt

Datenblock mit Kontoumsätzen für ein Konto, bzw. zur Anforderung von Kontoumsätzen.

Name	Typ	U/D	Beschreibung
ntrysId	string	U/D	*Verpflichtend*. ID der Anforderung der Kontoumsatzdaten. Diese kann vom Server frei gewählt werden und wird im Upload der Kontoumsätze zur Identifizierung mitgeliefert. Wird üblicherweise zur Identifizierung des Kontos auf der Serverseite verwendet.
dateFrom	ISODate	U/D	*Optional*. Datum, ab dem die Umsätze angefordert bzw. übertragen werden. Wird kein Datum eingestellt, so werden alle in BankingZV für dieses Konto vorliegenden Kontoumsätze hochgeladen.
includePending	bool	U/D	*Optional*. Wird dieses als `true` eingestellt, dann sollen auch Buchungsvormerkungen (Status PEND) übertragen werden. Ist der Wert `false`, oder nicht vorhanden, so werden nur echte Buchungen (Status BOOK) übertragen.
ntrysFormat	string	U/D	*Verpflichtend*. Datenformat, in dem die Umsätze hochgeladen werden sollen. Es sind folgende Werte möglich: `supa.csv` – Eine CSV Datei gemäß SUPA Spezifikation. `supa.json` – Eine JSON Datei gemäß SUPA Spezifikation. `camt.052` – Eine ISO 20022 Datei gemäß "camt.052.001.08".
ntrysZip	string	U	*Verpflichtend*. Base-64 codierte ZIP-Datei mit den angeforderten Umsatzdaten. In der ZIP-Datei ist genau eine komprimierte Datei mit den Umsatzdaten enthalten. Diese Datei entspricht dem oben angegebenen `ntrysFormat`.

6.5 REST Endpoints

6.5.1 adviseAcct Endpoint

`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.

Der Endpoint `adviseAcct` wird nach jedem erfolgreichen Umsatzabruf eines Kontos aufgerufen. In BankingZV somit bei jedem Rundruf, bei einzelnen Kontoaktualisierungen, bei direkten Umsatzabrufen und bei Aktualisierungen der Ansicht Aufträge.

Wird ein Zugang zu einem Buchhaltungsdienst in BankingZV konfiguriert, so testet BankingZV diese Konfiguration, indem es einen `adviseAcct` Request ohne Konto, mit nur dem `userToken` sendet. Der Buchhaltungsdienst muss diesen `userToken` verifizieren und einen entsprechenden Response generieren. Ist der Test-Request gültig wird vom Buchhaltungsdienst ein leeres JSON als Response erwartet. Ist der Test-Request ungültig, wird eine der definierten Fehlerantworten (siehe 6.3.1 Applikationsspezifische Fehlercodes) erwartet.

6.5.1.1 Request Body

Name	Typ	Beschreibung
userToken	string	*Verpflichtend*. Token zur Authentifizierung des Anwenders.
acct	Bal	*Optional*. Kontoinformationen mit Salden im SUPA JSON Format.
requestPaymts	bool	*Optional*. Wird als `true` eingestellt, wenn bereits mit der Antwort Zahlungsdateien für dieses Konto bereitgestellt werden sollen. Ist der Wert `false`, oder nicht vorhanden, so sollen in der Antwort keine Zahlungsdateien übertragen werden.

Beispielrequest:

```
{
    "userToken": "<token>",
    "acct": {
        "Id": "87618185228262427",
        "AcctBankCode": "12030000",
        "AcctNo": "1234567890",
        "AcctBIC": "BYLADEM1001",
        "AcctIBAN": "DE45120300001234567890",
        "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"
    },
    "requestPaymts": true
}
```

6.5.1.2 Response Body (Success 200)

In der Antwort kann der Server den Upload von Kontoumsätzen anfordern und Zahlungsaufträge für den Import in BankingZV bereitstellen.

Name	Typ	Beschreibung
ntrysInfo	NtrysInfo	*Optional*.Aufforderung zum Upload von Kontoumsätzen für dieses Konto. Im `ntrysInfo` Objekt ist kein `ntrysZip` enthalten. Ist dieses Element nicht enthalten, so werden für dieses Konto keine Umsätze hochgeladen.
paymtsInfos	PaymtsInfo[]	*Optional*. Array mit Zahlungsdateien, die für dieses Konto importiert werden sollen. Darf nur eingestellt werden, wenn diese durch `requestPaymts` angefordert wurden.

Beispielantwort:


```
{
    "ntrysInfo": {
        "ntrysId": "23187672361768",
        "dateFrom": "2020-11-23",
        "includePending": true,
        "ntrysFormat": "supa.json"
    },
    "paymtsInfos": [
        {
            "paymtsId": "1000001",
            "paymtsFormat": "pain.001",
            "paymtsZip": "<Base64 ZIP>"
        },
        {
            "paymtsId": "1000002",
            "paymtsFormat": "pain.008",
            "paymtsZip": "<Base64 ZIP>"
        }
    ]
}
```


6.5.2 updateAcct Endpoint

`POST /updateAcct`

Über diesen Endpoint werden die für das Konto angeforderten Kontoumsätze hochgeladen und erfolgreich importierte Zahlungsdateien bestätigt. Dieser Endpoint wird unmittelbar nach dem Endpoint `adviseAcct` aufgerufen.

Bestätigt den erfolgreichen und vollständigen Import von Zahlungsdateien. Bestätigte Zahlungsdateien sollen beim nächsten Aufruf des Endpoints `adviseAcct` nicht mehr bereitgestellt werden.

6.5.2.1 Request Body

Name	Typ	Beschreibung
userToken	string	*Verpflichtend*. Token zur Authentifizierung des Anwenders.

Name	Typ	Beschreibung
ntrysInfo	NtrysInfo	*Optional*. Datenblock mit den angeforderten Kontoumsatzdaten.
paymtsInfos	PaymtsInfo	*Optional*. Array mit `PaymtsInfo` Objekten in dem die Felder `paymtsId` und `paymtsStatus` gesetzt sind. Dies ist die `paymtsId` eines zuvor erhaltenen Zahlungsblocks und der Status des versuchten Imports.

Beispielrequest:

```
{
    "userToken": "<token>",
    "ntrysInfo": {
        "ntrysId": "23187672361768",
        "ntrysFormat": "supa.json",
        "ntrysZip": "<Base64 ZIP>"
    },
    "paymtsInfos": [
        {
            "paymtsId": "1000001",
            "paymtsStatus": "OK"
        },
        {
            "paymtsId": "1000002",
            "paymtsStatus": "FAILED"
        }
    ]
}
```

6.5.2.2 Response Body (Success 200)

Eine erfolgreiche Kontoaktualisierung wird durch den Http Code 200 signalisiert. Ob der Request erfolgreich war, oder nicht, wird in BankingZV ignoriert. Es wird jedoch im Fehlerfall die im Response enthaltene Meldung ausgegeben.

Im Fehlerfall kann der Buchhaltungsdienst die Kontoumsätze erneut anfordern, indem es einen entsprechenden Wert für `dateFrom` in der Antwort zu `adviseAcct` liefert.

Konnte der Buchhaltungsdienst eine Importbestätigung nicht verarbeiten, so wird es im nächsten `adviseAcct` die gleichen Zahlungen erneut bereitstellen. Wird bei der zweiten Bereitstellung in

BankingZV festgestellt, dass die gleiche `paymtsId` bereits erfolgreich importiert wurde, so wird dies mit dem `paymtsStatus` DUPLICATE quittiert.

Erfolreicher Response:

```
{
}
```

6.6 REST Abläufe

Die im vorigen Abschnitt beschriebenen Endpoints werden in BankingZV nach *jedem* Abruf von Kontoumsätzen für ein *Bankkonto* aufgerufen. Ein Bankkonto ist jedes in BankingZV angelegte Kontokorrentkonto mit *gültiger IBAN* und einem *Online-Banking-Zugang*, über den die zugehörigen Kontoumsätze abgerufen werden können.

Es wird immer zuerst `adviseAcct` und danach `updateAcct`, wie im Folgenden beschrieben, aufgerufen

6.6.1 adviseAcct

BankingZV sendet einen POST Request an `adviseAcct` mit den Kontodetails des eben abgerufenen Bankkontos als `Acct` und dem Flag `requestPaymts` gleich `true`.

Der Empfänger - also der Buchhaltungsdienst – prüft, ob das Bankkonto für ihn relevant ist und legt es ggf. im eigenen Datenstamm an.

Werden vom Empfänger für dieses Bankkonto Kontoumsätze benötigt, so wird im Response ein `ntrysInfo` Element eingestellt. Darin kann vom Empfänger *optional* eine frei wählbare `ntrysId` eingestellt werden. Zusätzlich *kann* ein `dateFrom` eingestellt werden. Wird kein `dateFrom` eingestellt, so werden im nächsten Schritt *alle* vorliegenden Umsatzdaten für dieses Konto übertragen. `dateFrom` kann grundsätzlich beliebig weit in der Vergangenheit liegen, jedoch können natürlich nur die tatsächlich in BankingZV vorliegenden Kontoumsätze übertragen werden. Über das *optionale* Element `ntrysFormat` kann die Übertragung der Kontoumsätze in einem bestimmten

Format angefordert werden. Um die Menge der übertragenen Daten möglichst klein zu halten, sollte zumindest bei Folgeabrufen ein sinnvolles `DateFrom` eingestellt werden. Es empfiehlt sich als `dateFrom` das Datum des letzten vorliegenden Buchungstages einzustellen.

Wurden im Buchhaltungsdienst bereits Zahlungsaufträge für das in `adviseAcct` mitgeteilte Bankkonto erfasst, so *können* diese im Response in einem oder mehreren PaymtsInfo Objekten an BankingZV zum Import in dieses Konto übergeben werden. Der Buchhaltungsdienst *sollte* für jedes PaymtsInfo Objekt eine `paymtsId` vergeben. Diese wird im `updateAcct` zur Bestätigung des Erfolgs oder des Fehlschlagens des Imports wieder an den Buchhaltungsdienst übergeben. Ohne `paymtsId` wird keine Bestätigung übertragen. BankingZV speichert im Datentresor eine Historie aller bereits verarbeiteten `paymtsIds`. Wird von BankingZV eine `paymtsId` empfangen, die bereits verarbeitet wurde, so werden die zugehörigen Zahlungen verworfen und im folgenden `updateAcct` für diese `paymtsId` wird DUPLICATE angezeigt. So kann der Buchhaltungsdienst unbestätigte Zahlungsdateien immer wieder in der Antwort zu `adviseAcct` einstellen, ohne Gefahr einer unerwünschten doppelten Ausführung.

Wichtiger Hinweis!	
	Der Buchhaltungsdienst muss darauf achten, dass das in den Zahlungsdateien enthaltene Ausführungsdatum (Requested Execution Date bzw. Requested Collection Date) zum Zeitpunkt des Abrufs gültig ist, also nicht in der Vergangenheit liegt.

6.6.2 updateAcct

Der Endpoint `updateAcct` wird in BankingZV immer unmittelbar nach der Verarbeitung der Antwort zu `adviseAcct` aufgerufen. Auf Grund von immer möglichen System- und Übertragungsfehlern gibt es aber keine Garantie, dass `updateAcct` tatsächlich aufgerufen wird.

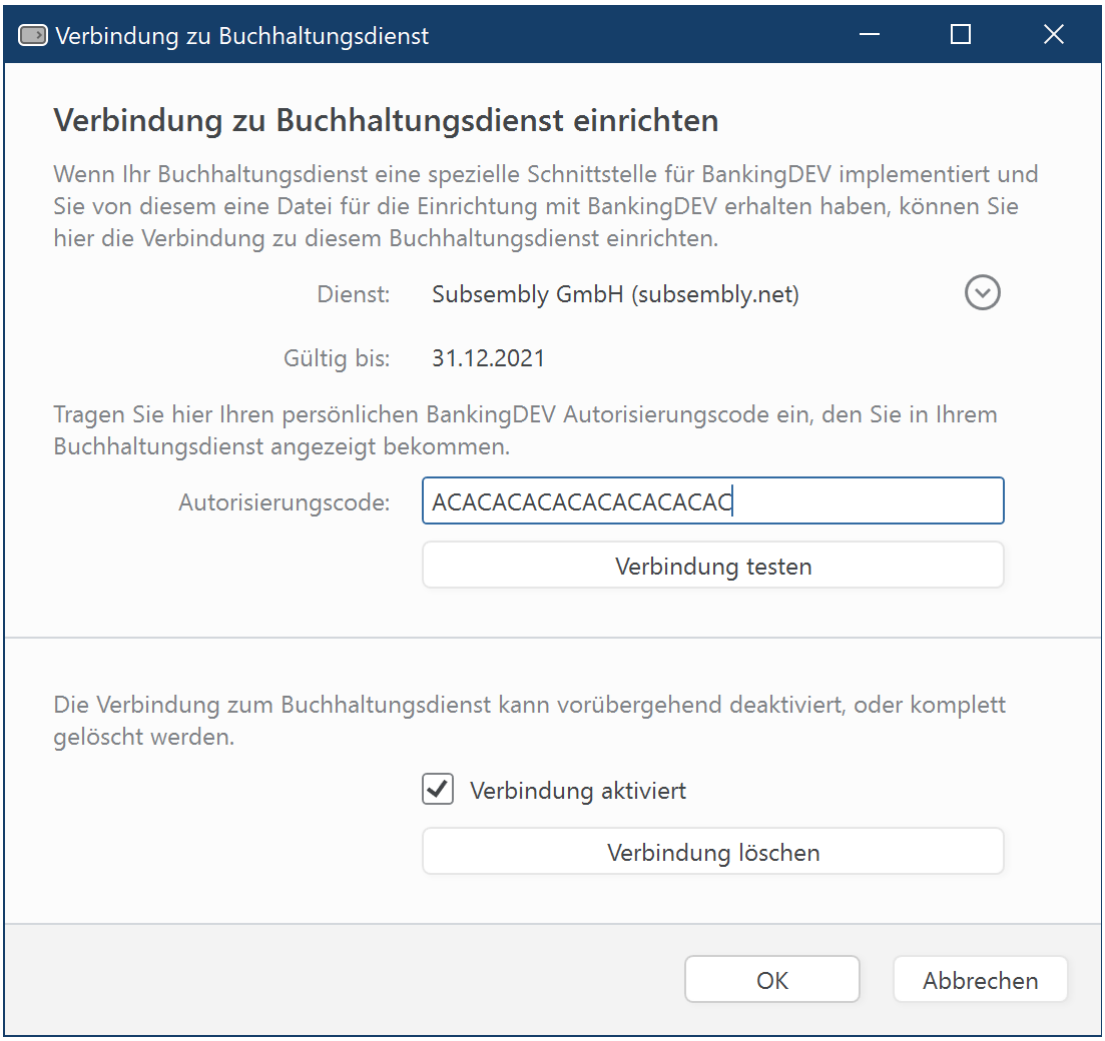
Wurden in der unmittelbar vorangegangenen Antwort auf `adviseAcct` vom Buchhaltungsdienst Kontoumsatzdaten angefordert, so werden diese als `ntrysZip` eingestellt. Ein `ntrysZip` wird auch dann eingestellt, wenn im angeforderten Zeitraum keine Buchungen vorliegen. Der Wert für `ntrysZip` ist niemals ein leerer String. Die ZIP-Datei selbst kann jedoch leer sein, oder in der ZIP-Datei kann eine leere Datei enthalten sein. Ist das angeforderte `fromDate` heute oder ist das Datum

kein Bankarbeitstag (Wochenende, Feiertag), so werden die Kontoumsätze bereits ab dem unmittelbar vorangehenden Bankarbeitstag hochgeladen. Dieses Vorgehen dient dazu, eine lückenlose Datenversorgung des Buchhaltungsdienstes sicherzustellen.

Für jedes in `adviseAcct` enthaltene PaymtInfo Element mit `paymtsId` wird ein PaymtInfo Element mit dem Ergebnis des Imports übertragen. War im PaymtInfo des Buchhaltungsdienstes keine `paymtsId` enthalten, so wird hierfür zwar keine Bestätigung im `updateAcct` generiert, die enthaltenen Zahlungsdateien aber dennoch importiert, wenn möglich. Im PaymtInfo Element der Bestätigung werden von BankingZV nur die Felder `paymtsId` und `paymtsStatus` eingestellt. Der Buchhaltungsdienst muss die Zuordnung zum Originalauftrag über seine `paymtsId` herstellen.

6.7 BankingZV Benutzerschnittstelle

In BankingZV wird ein Dialogfenster zur Erfassung des API-Tokens und des Authorisierungscodes bereitgestellt. Darin wird der API-Token aus einer Datei über eine Dateiauswahl geladen. Nach erfolgreichem Laden des API-Tokens wird der Name des Buchhaltungsdienstes und der Hostname der URL des Endpoints angezeigt.



Das Dialogfenster wird über den Menüpunkt "Mit Buchhaltungsdienst verbinden ..." aufgerufen, welcher nur bei Vorhandensein einer BankingZV Firmenlizenz zur Verfügung steht.

Bei White-Label-Versionen von BankingZV ist es möglich, das API-Token des zugehörigen Buchhaltungsdienstes fest zu hinterlegen. In diesem Fall kann der Anwender in diesem Dialogfenster den API-Token nicht manuell auswählen oder löschen.