SUBSEMBLY
Banking Apps & APIs

BankAccessServer

API Spezifikation

Version 3.5.5

Subsembly GmbH

Hofmannstr. 7b
81379 München

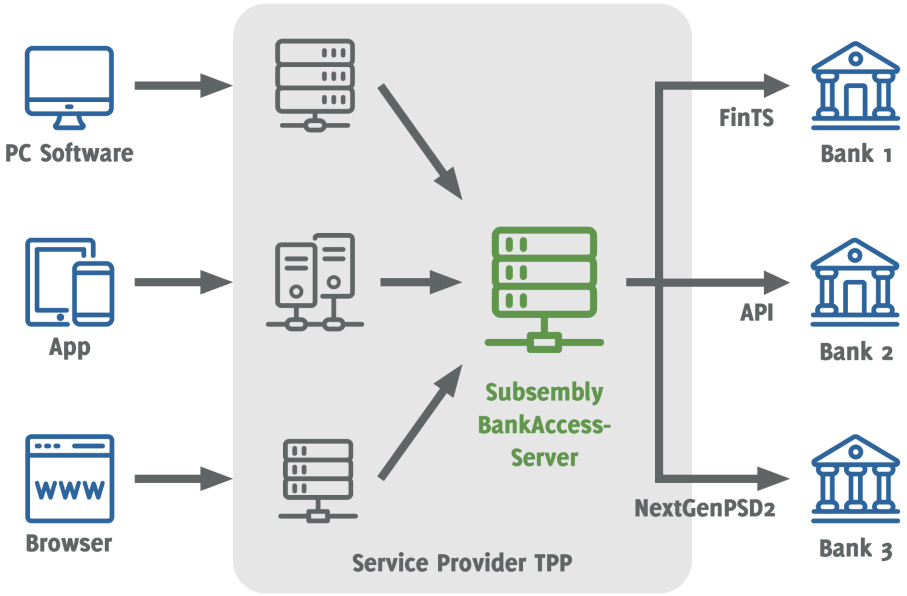
https://subsembly.com
bas@subsembly.com

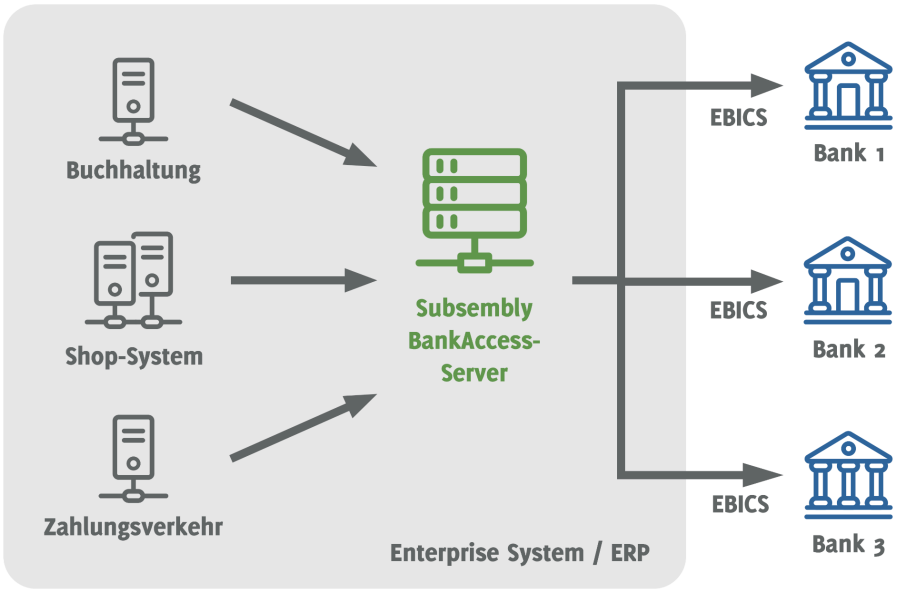
Stand: 14.11.2024

Inhaltsverzeichnis

Überblick

Der Subsembly BankAccessServer ist ein plattformneutrales, vollständig auf Basis von .NET Core realisiertes, Servermodul für den Zugriff auf Kontodaten (XS2A) und die Übermittlung von Zahlungen via FinTS (bzw. HBCI) oder EBICS. Für lizenzierte Drittanbieter besteht die Möglichkeit auf Banken über die Berlin Group Schnittstelle zuzugreifen und hierüber Kontoinformationen auszulesen und Zahlungen auszuführen.

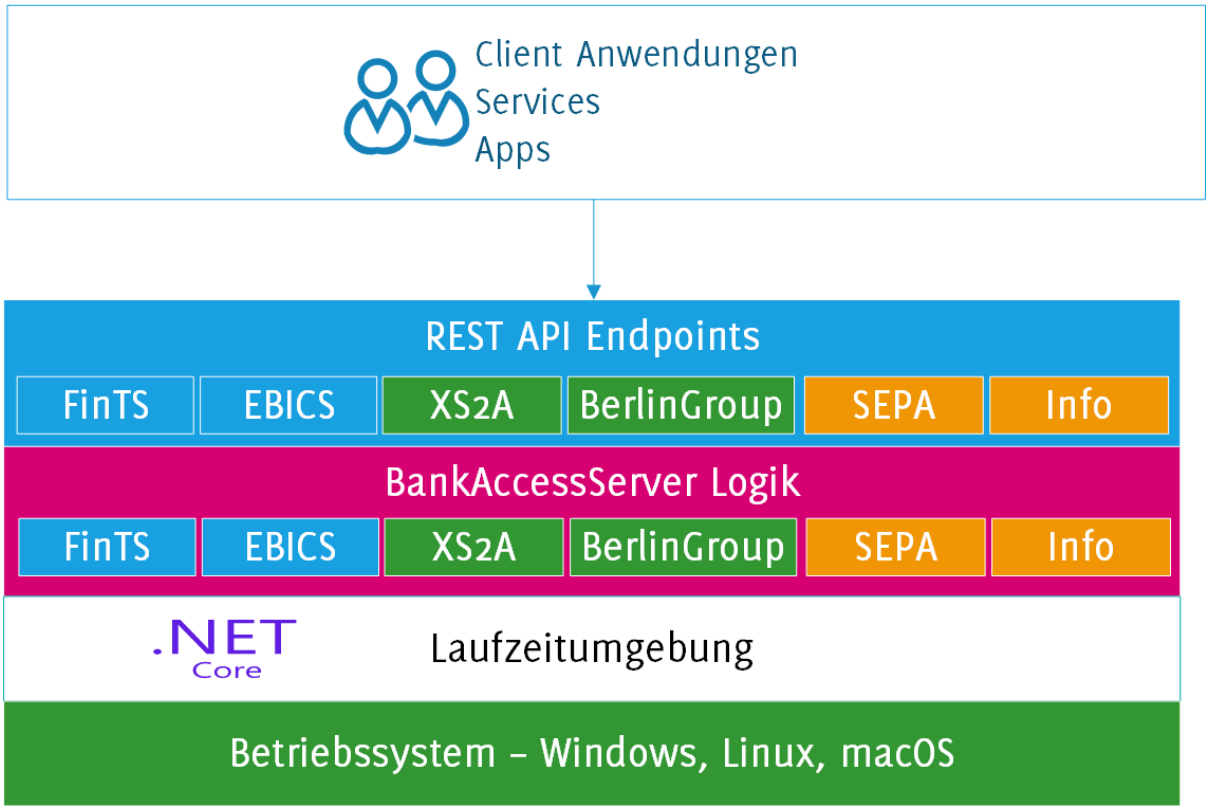




Dieses Dokument beschreibt die REST-Schnittstelle über die Client-Anwendungen auf den Subsembly BankAccessServer zugreifen.

Allgemeine Schnittstellenbeschreibung

Die clientseitige Schnittstelle des Subsembly BankAccessServer gliedert sich in folgende Schichten.



Auf unterster Ebene ist die HTTP-Transportschicht, welche von .NET Core, basierend auf dem Host-Betriebssystem, zur Verfügung gestellt wird. Auf dieser Ebene finden auch die Verschlüsselung (TLS) und weitere Sicherungsmaßnahmen statt.

Auf Basis von HTTP Requests wird die REST-Schnittstelle des Subsembly BankAccessServer definiert. Dies ist die Schicht, die Abläufe und Zustände des Subsembly BankAccessServer steuert. Aus Anwendungssicht wird über die jeweiligen Endpoints FinTS, EBICS, XS2A, BerlinGroup, SEPA und Info mit dem BankAccessServer kommuniziert.

REST Schnittstelle

Alle Funktionen des Subsembly BankAccessServers können über einen logischen Endpoint (URL) mit entsprechenden REST-Anfragen genutzt werden. An diesen Endpoint werden Requests per HTTP POST gesendet.

Die Anfrage bzw. Auftragsdaten werden im JSON-Format im HTTP-Body eingestellt. Das Ergebnis der Verarbeitung wird - ebenfalls als JSON-Dokument - in der HTTP-Response zurückgeliefert. So ergibt sich eine einfach zu nutzende, klassische REST-Schnittstelle.

FinTS

Jeder an den Subsembly BankAccessServer gesendete FinTS-Request besteht aus den Abschnitten:

- *RequestOptions*
- *Connection*
- *Orders*
- *ResponseOptions*

Die grundsätzliche Struktur des Requests entspricht folgendem Aufbau, wobei nicht alle gezeigten Elemente verpflichtend oder in dieser Kombination gültig sind.

```
{
    "RequestOptions": {
        "RequestId": ...,
        "AccessToken": ...
    },

    "Connection": {
        "LogOn": {
            "Contact" : {
                ...
            },
            "ContactSerialized": "xxx",
            "Pin": "xxx"
        },
        "Resume": {
            "Service": "xxx",
        },
        "Suspend": ...
    },
```

```
    "Orders": [
        {
            "OrderId": "xxx",
            "Type": "xxx",
            ...
        },
       ...
    ],

    "ResponseOptions": {
        "Trace": ...,
        "Contact": ...
    }
}
```

Die Antwort des Subsembly BankAccessServer enthält die *Responses* zu den übergebenen *Orders*. Zusätzlich werden anhand der Angaben in den Response-Optionen weitere Datenelemente eingestellt.

```
       {
           "Responses": [
               {
                   "OrderId": "xxx",
                   "Type": "xxx",
                   ...
               },
               ...
           ],
           "Trace": ...,
           "Contact": ...,
           "Service": ...
       }
```

In den folgenden Abschnitten werden die verschiedenen Elemente im Einzelnen beschrieben.

Request Options

Angabe der RequestId und eines Access Tokens für die aktuelle FinTS Anfrage. Weitere Informationen zu den Request Optionen finden Sie hier.

Connection

Das Connection Objekt steuert den Verbindungsaufbau (Neuanmeldung / Wiederverwendung einer bestehenden Verbindung) und den Verbindungsabbau (Abmeldung / Weiterverwendung der Verbindung) für das FinTS Protokoll.

Das Connection Objekt kann grundsätzlich die folgenden Felder enthalten:

Feldname	**Typ**		**Beschreibung**
LogOn	Object	C	Enthält Informationen für den Aufbau einer neuen Verbindung.
Resume	Object	C	Enthält Informationen für die Weiterverwendung einer bestehenden Verbindung.
SetSecurityFunction	Object	C	Festlegen des TAN Verfahrens, sofern die vorherige Response das Feld NeedSecurityFunction enthielt.
SetTanMediaName	Object	C	Festlegen des TAN Mediums, sofern die vorherige Response das Feld NeedTanMediaName enthielt.
SendTan	Object	C	Senden der TAN, sofern die vorherige Response das Feld NeedTan enthielt.
SendDecoupled	Object	C	Senden der Information, dass die decoupled Freigabe erfolgt ist.
Suspend	Boolean	O	Wird dieses Feld mit dem Wert *true* übergeben, so wird die nachgelagerte Verbindung nicht beendet und es wird ein serialisierter Sitzungskontext in der Response (Service) bereitgestellt. Über den Wert *false* wird die Verbindung im Anschluss beendet.

LogOn - Aufbau einer neuen FinTS Verbindung

Feldname	**Typ**		**Beschreibung**
Contact	Object	C	Das Contact Objekt enthält alle für den Online-Banking Verbindungsaufbau erforderlichen Parameter. Welche Parameter im Einzelnen erforderlich sind, hängt von der Art des verwendeten Online-Banking Zugangs ab.
ContactSerialized	string	C	Alternativ zu den einzelnen Feldern, können in diesem einen Feld auch die vollständigen, Base64 codierten Zugangsdaten aus einer vorherigen Anmeldung des Endanwenders verwendet werden.
Pin	string	M	Der für die Anmeldung des Endanwenders am Banksystem erforderliche Sicherheitscode, wie zum Beispiel eine Online-Banking-PIN oder das Schlüsselpasswort.
ProductName	string	O	Produktkennung, die im HKVVB Segment bei der Dialoginitialisierung geschickt wird. Sofern nicht angegeben, wird der Wert aus den appsettings verwendet.
ProductVersion	string	O	Produktversion, die im HKVVB Segment bei der Dialoginitialisierung geschickt wird. Sofern nicht angegeben, wird der Wert aus den appsettings verwendet.
OverwriteCommAddress	Boolean	O	Kann in Verbindung mit einem gespeicherten Kontakt gesetzt werden und steuert, ob die im serialisierten Kontakt gespeicherte CommAddress durch die FinBanks Adresse überschrieben wird.

Falls die Zugangsdaten bereits in serialisierter Form aus einem früheren Request vorliegen, kann statt der einzelnen Parameter auch einfach das jeweilige serialisierte *Contact* Feld eingestellt werden.

Es ist darauf zu achten, dass die Banken einen FinTS/HBCI Dialog nur wenige Minuten offen halten und jeder FinTS/HBCI Dialog letztendlich sauber beendet werden muss.

Contact

Im Idealfall sind für eine Anmeldung über FinTS nur die Bankleitzahl, Benutzerkennung und Online-Banking-PIN erforderlich. Alle weiteren Informationen, die für FinTS benötigt werden, ermittelt der Subsembly BankAccessServer automatisch anhand einer umfangreichen Datenbasis mit bekannten FinTS/HBCI Zugangsdaten. Diese Datenbasis wird von uns kontinuierlich aktualisiert und ausgeliefert.

Im Rahmen der von der PSD2 geforderten starken Kundenauthentifizierung kann die Angabe einer TAN notwendig werden.

Die für jede Anmeldung erforderlichen Authentifizierungsinformationen des Kunden, also zum Beispiel seine Online-Banking-PIN, müssen immer explizit im Feld *Pin* eingestellt werden.

Für den Bankzugang via FinTS/HBCI können folgende Felder im *Contact* explizit übergeben werden. Im einfachsten Fall sind bereits die Felder *BankCode* und *UserID* ausreichend.

Feldname	**Typ**		**Beschreibung**
BankCode	string	M	Bankleitzahl des Bankzugangs.
UserID	string	M	FinTS/HBCI Benutzerkennung des Benutzers.
DefaultCustID	string	O	FinTS/HBCI Kunden-ID des Benutzers. Wird diese nicht übergeben, so wird die Kunden-ID gleich der Benutzerkennung gesetzt.
CommAddress		O	Explizite Angabe der zu verwendenden URL inklusive Präfix (https://).
FinTSVersion		O	Explizite Angabe der zu verwendenden FinTS Version.
SecurityFunction	string	O	Explizite Auswahl des TAN-Verfahrens über die eindeutige “SecurityFunction”. Fehlt die Angabe und es gibt nur ein verfügbares TAN-Verfahren, so wird dieses implizit ausgewählt. Bei mehreren vorhandenen Verfahren muss eine eindeutige Auswahl erfolgen.
TanMediaName	string	O	Bezeichnung des TAN-Mediums. Dieser ist erforderlich, falls der Kunde über mehrere aktive TAN-Medien, z.B. mehrere freigeschaltete Handys beim SMS-TAN-Verfahren, verfügt. Eine Liste aller verfügbaren TAN-Medien

			kann über den separaten Auftrag *FinTanMediaList* abgerufen werden.
StrongCustomerAuthentication	string	O	• Default - keine SCA bis 14.09.2019, danach mit SCA • RequestSCA - SCA aktiviert (hilfreich um im Vorfeld gegen Server mit SCA testen zu können) • NoSCA - ggfs. sinnvoll wenn nach dem 14.09.2019 Server noch kein SCA aktiviert haben

Ausführliche Informationen zu den möglichen Parametern eines FinTS *Contact* Objekts finden Sie auf https://subsembly.com/apidoc/Subsembly.FinTS.FinContact.html

Resume

Sofern für die Verbindung kein Logout erfolgt, kann statt eines LogOn Objekts auch das in der letzten Response zurückgelieferte Service Objekt angegeben werden. In diesem Fall kann auf einen erneuten Login verzichtet werden und mit der bestehenden Session weitergearbeitet werden. Durch die Wiederverwendung einer bestehenden Verbindung lässt sich eine besonders gute Performance erreichen.

Feldname	**Typ**		**Beschreibung**
Service	Object	M	Serialisierter Sitzungskontext

SetSecurityFunction

Sofern der Benutzer mehr als ein TAN Verfahren besitzt, muss dieses nachfolgend spezifiziert werden, um eine eindeutige Auswahl zu treffen. Die aktuelle Session wird für folgende Aufrufe weiterverwendet und über den serialisierten Sitzungskontext (Service) dargestellt. Alle weiteren Aufrufe müssen wiederum die zuvor gesendeten Orders beinhalten.

Die Anmeldung enthält in diesem Fall in der Response das Feld **NeedSecurityFunction**, das wiederum alle Sicherheitsverfahren des Kunden enthält.

Feldname	**Typ**		**Beschreibung**
Service	Object	M	Serialisierter Sitzungskontext
SecurityFunction	int	M	Angabe der SecurityFunction, die das

			Sicherungsverfahren spezifiziert

SetTanMediaName

Weiterhin kann es notwendig sein ein TAN Medium zu spezifizieren. Das kann z.B. notwendig sein wenn der Kunde ein SMS oder App TAN Verfahren auf mehreren Endgeräten nutzt. In diesem Fall enthält die Antwort auf die Anmeldung das Feld **NeedTanMediaName**, das die verfügbaren TAN-Medien beinhaltet.

Das zu verwendende TAN Medium wird wie folgt festgelegt und muss wiederum die zuvor gesendeten Orders beinhalten.

Feldname	**Typ**		**Beschreibung**
Service	Object	M	Serialisierter Sitzungskontext
TanMediaName	string	M	Angabe des ausgewählten TAN Mediums

SendTan

Sofern im Rahmen der Anmeldung eine TAN notwendig ist, enthält die Antwort das Feld **NeedTan** mit der zugehörigen TAN-Challenge.

```
"Challenge": "Die mobileTAN zu diesem Auftrag wird als SMS an Ihre
Mobil-Telefonnummer gesendet, die bei Ihrer Bank registriert ist.",
```

Bei der TAN Übertragung müssen wiederum die zuvor gesendeten Orders beinhaltet sein.

Feldname	**Typ**		**Beschreibung**
Service	Object	M	Serialisierter Sitzungskontext
Tan	string	M	Transaktionsnummer / OTP

SendDecoupled

Sofern im Rahmen der Anmeldung eine SCA notwendig ist, enthält die Antwort das Feld **NeedDecoupled** mit dem zugehörigen Challenge Prompt.

```
"ChallengePrompt": "Bitte bestätigen Sie die Anmeldung bzw. Ihren Auftrag in
der App Ihres Kreditinstituts und klicken Sie anschließend auf [OK]."
```

Bei der Übertragung muss das in der vorhergehenden Antwort zurückgelieferte Service Object mit angegeben werden. Sofern die Freigabe der Anmeldung noch nicht erfolgt ist, wird der Auftrag erneut mit NeedDecoupled beantwortet.

Feldname	**Typ**		**Beschreibung**
Service	Object	M	Serialisierter Sitzungskontext

Orders (FinTS)

Array mit gültigen FinTS Aufträgen - eine allgemeine Beschreibung finden Sie unter dem Punkt Orders

ResponseOptions

Für FinTS Verbindungen kann festgelegt werden, ob die Antwort einen vollständigen FinTS/HBCI Trace für den vorangegangenen FinTS/HBCI Dialog enthalten soll.

Darüber hinaus kann gesteuert werden, ob die Response den finalen Zustand des Contact Objekts liefern soll. Wird das Contact Objekt gespeichert und für eine spätere Anmeldung einfach wiederverwendet, ist die FinTS/HBCI Dialoginitialisierung wesentlich schneller.

Feldname	**Typ**		**Beschreibung**
Trace	Boolean	O	Wird dieses Feld mit dem Wert *true* übergeben, so wird in den Antwortdaten ein komplettes Übertragungsprotokoll für Diagnosezwecke geliefert. Über das Flag TraceOnError in den appsettings kann festgelegt werden ob im Fehlerfall immer ein HBCI Trace zurückgeliefert werden soll.
Contact	Boolean	O	Wird dieses Feld mit dem Wert *true* übergeben, so wird in den Antwortdaten ein serialisierte Contact Feld eingestellt.
Performance	Boolean	O	Wird dieses Feld mit dem Wert *true* übergeben, so werden in der Response die FinTS Performance Informationen eingestellt.

EBICS

Jeder an den Subsembly BankAccessServer gesendete EBICS-Request besteht aus den Abschnitten:

- *RequestOptions*
- *Connection*
- *Orders*

Die grundsätzliche Struktur eines Requests entspricht folgendem Aufbau, wobei nicht alle gezeigten Elemente verpflichtend oder in dieser Kombination gültig sind.

```
{
    "RequestOptions": {
        "RequestId": ...,
        "AccessToken": ...
    },

    "Connection": {
        "ContactRef" : {
            ...
        },
        "ContactSerialized": "xxx",
        "Pin": "xxx"
    },
    "Orders": [
        {
            "OrderId": "xxx",
            "Type": "xxx",
            ...
        },
       ...
    ],
    "ResponseOptions": {
        "Trace": ...
    },
}
```

Im Gegensatz zu FinTS und XS2A Requests, bei denen häufig dynamische und wechselnde Zugangsdaten verwendet werden, finden EBICS Zugriffe in der Regel nicht ad Hoc statt. Für die Verwendung ist zunächst die Einrichtung eines EBICS Zugangs erforderlich, der vollständig über die administrativen Workflows des BankAccessServers erfolgen kann.

Für den Zugriff auf den EBICS Endpoint benötigt der Client einen Client-Access-Token mit einer der folgenden Berechtigungen.

Berechtigung	**Beschreibung**
Admin	Berechtigung ist zur Ausführung von administrativen EBICS Orders
Download	Berechtigung zur Ausführung aller EBICS Download Orders, z.B. Umsatzabrufe
Upload	Berechtigung zur Ausführung aller EBICS Upload Orders, z.B. Einreichung von SEPA Zahlungsdateien

Der BankAccessServer ist mandantenfähig und stellt den durch die Client-ID eindeutig identifizierten Mandanten jeweils einen eigenen Speicherbereich zur Verfügung. Die ID dabei ist Teil des in dem Request übermittelten Client-Access-Tokens.

Wichtig: Für den Zugriff auf den EBICS Endpoint benötigt der Client ein Client-Access-Token, aus dem die Berechtigungen des Clients abgeleitet werden können.

Request Options

Angabe der RequestId und eines Access Tokens für die aktuelle EBICS Anfrage. Weitere Informationen zu den Request Optionen finden Sie hier.

Connection

Für jeden Request muss ein EbicsContact und das zugehörige Passwort für die Schlüsseldaten des Teilnehmers spezifiziert werden. Ein EbicsContact kann grundsätzlich als JSON Objekt oder als base-64 codiertes XML dargestellt werden.

Das Connection Objekt steuert den Verbindungsaufbau für das EBICS Protokoll und kann grundsätzlich die folgenden Felder enthalten.

Feldname	**Typ**		**Beschreibung**
ContactRef	Object	O	JSON Objekt, das alle für die EBICS Kommunikation notwendigen Informationen des Teilnehmers enthält, siehe auch: EbicsContact
ContactSerialized	string	O	Alternativ zu den einzelnen Feldern, können in diesem einen Feld auch die vollständigen, Base64 codierten Zugangsdaten verwendet werden.
KeyPassword	string	M	Passwort für die Schlüsseldatei des Teilnehmers

Contact

Folgende öffentliche Felder sind in einem EbicsContact vorhanden.

Feldname	**Typ**		**Beschreibung**
HostID	string	M	Die EBICS Host-ID des EBICS Zielsystems.
HostURL	string	M	Die Internet URL unter der das EBICS Zielsystem erreichbar ist.
PartnerID	string	M	Die EBICS Partner-ID, bzw. Kunden-ID unter der sich der EBICS Teilnehmer am EBICS Zielsystem anmeldet.
UserID	string	M	Die EBICS Teilnehmer-ID, bzw. Benutzer-ID unter der sich der EBICS Teilnehmer am EBICS Zielsystem anmeldet.
SystemID	string	C	Für rein technische EBICS Teilnehmer wird zusätzlich zur UserID eine SystemID benötigt. Andernfalls darf dieses Element nicht verwendet werden.

Version	string	O	Die zu verwendende EBICS Schemaversion. Der Wert muss entweder "H001", "H002", "H003" oder "H004" sein. Wird dieses Element nicht angegeben, so wird "H004" angenommen.
AuthenticationVersion	string	O	Version des Authentifizierungsverfahrens. Der Wert muss entweder "X001" oder "X002" sein und zur verwendeten EBICS Version passen. Wird dieses Element nicht angegeben, so wird "X002" angenommen.
EncryptionVersion	string	O	Version des Verschlüsselungsverfahrens. Der Wert muss entweder "E001" oder "E002" sein und zur verwendeten EBICS Version passen. Wird dieses Element nicht angegeben, so wird "E002" angenommen.
SignatureVersion	string	O	Version des Signaturverfahrens. Der Wert muss entweder "A004", "A005" oder "A006" sein und zur verwendeten EBICS Version passen. Wird dieses Element nicht angegeben, so wird "A006" angenommen.

ContactSerialized

Dieses Feld enthält einen string mit einem base-64 codiertem XML-Dokument. Das XML-Dokument entspricht dabei dem von der Subsembly EBICS API für einen EbicsContact erzeugtem XML. Ein solches XML-Dokument kann im EbicsAdmin der Subsembly EBICS API für einen EBICS Bankzugang exportiert oder importiert werden (Siehe https://subsembly.com/apidoc/ebics/Subsembly.EBICS.EbicsContact.html).

EbicsReturnCodes

Bei vielen Requests werden im Response zwei EbicsReturnCode Objekte mit dem technischen EBICS Return-Code und dem Business EBICS Return-Code des EBICS Zielsystems eingestellt. Ein EBICS Return-Code ist eine sechsstellige Zeichenkette, die ausschließlich Ziffern enthält. Der genaue Aufbau ist Anhang 1 der EBICS Spezifikation beschrieben. Das EbicsReturnCode Objekt enthält darüber hinaus zusätzliche optionale Elemente.

Feldname	**Typ**		**Beschreibung**
ReturnCode	string	M	Sechsstelliger Return-Code. Zum Beispiel "000000" für eine fehlerfreie Verarbeitung ohne zusätzliche Hinweise.
ReportText	string	O	Bei einem technischen Return-Code kann hier zusätzlich vom EBICS Zielsystem ein lesbarer Fehlertext eingestellt worden sein.
SymbolicName	string	O	Optional ein symbolischer Name des Return-Codes. Zum Beispiel: "EBICS_OK" oder "EBICS_USER_UNKNOWN".

Orders (EBICS)

Array mit gültigen EBICS Aufträgen - eine allgemeine Beschreibung finden Sie unter dem Punkt Orders

ResponseOptions

Für EBICS Verbindungen kann festgelegt werden, ob die Antwort einen Trace (sofern verfügbar) für den vorangegangenen Request enthalten soll.

Feldname	**Typ**		**Beschreibung**
Trace	Boolean	O	Wird dieses Feld mit dem Wert *true* übergeben, so wird in den Antwortdaten ein kompletter Trace für Diagnosezwecke geliefert.

XS2A

Lesende Kontozugriffe auf Basis von XS2A basieren auf folgender Request-Struktur:

- Request Options
- Connection
- Orders
- Response Options

Hinweis: Über den Info-Request GetXs2aContactInfoRequest können allgemeine Informationen zum angegebenen XS2A Zugang abgerufen werden, ohne dass hierfür eine Anmeldung erforderlich ist. Weitere Infos finden Sie hier .

Die grundsätzliche Struktur eines XS2A-Requests entspricht folgendem Aufbau, wobei nicht alle gezeigten Elemente verpflichtend oder in dieser Kombination gültig sind.

```
{
    "RequestOptions": {
        "RequestId": ...,
        "AccessToken": ...
    },

    "Connection": {
        "Service": {
            "Account" : {
                ...
            },
            "ContactSerialized": "xxx",
            "Xs2aId": "xxx"
        },
        "Resume": {
            "Service": "xxx"
        },
        "Credentials": {
            "UserID": "xxx",
            "Password": "xxx",
            "OAuthToken": "xxx",
            "ChallengeResponse": "xxx",
            "APIKey": "xxx",
        },
        "Suspend": ...
    },
    "Orders": [
```

```
        {
            "OrderId": "xxx",
            "Type": "xxx",
            ...
        }, ...
    ],
    "ResponseOptions": {
        "Trace": ...,
        "Contact": ...
    }
}
```

Request Options

Angabe der RequestId und eines Access Tokens für die aktuelle XS2A Anfrage. Weitere Informationen zu den Request Optionen finden Sie hier.

Connection

Das Connection Objekt steuert den Verbindungsaufbau (Neuanmeldung / Wiederverwendung einer bestehenden Verbindung) und den Verbindungsabbau (Abmeldung / Weiterverwendung der Verbindung) über den XS2A Endpoint. Das Connection Objekt kann grundsätzlich die folgenden Felder enthalten:

Feldname	**Typ**		**Beschreibung**
Service	Object	C	Ermittlung des XS2A Services über Konto-/Kreditkarteninformationen oder eine direkte Adressierung anhand des Xs2a Typs, z.B. Barclaycard oder N26. Es muss entweder das Service oder ein Resume Objekt angegeben werden.
Credentials	Object	C	Anmeldeinformationen, z.B. UserID / Password / OAuthToken / ChallengeResponse / APIKey
SetChallengeResponse	Object	C	Beinhaltet im Rahmen einer SCA Anmeldung die SCA Response (z.B. TAN-Nummer) und eine Referenz auf die aktuelle Session.
SetQueryResponse	Object	C	Beinhaltet im Rahmen einer SCA Anmeldung Informationen, die der Benutzer als Antwort auf eine Auswahl selektiert hat, z.B. das zu verwendende TAN Verfahren oder TAN Medium.

Resume	Object	C	Enthält Informationen für die Weiterverwendung einer bestehenden Verbindung. Es muss entweder das Service oder ein Resume Objekt angegeben werden.
Suspend	Boolean	O	Wird dieses Feld mit dem Wert *true* übergeben, so wird die nachgelagerte Verbindung nicht beendet und es wird ein serialisierter Sitzungskontext im Response (Session) bereitgestellt.

Die eigentlichen XS2A Zugriffe sind mit folgenden Connection-Angaben möglich:

1. Angabe der Service Informationen für die Datenquelle in Verbindung mit den Credentials. Ein Service kann folgendermaßen identifiziert werden:
    a. Informationen eines Bank- oder Kreditkartenkontos, siehe Account
    b. Base64 codierte Zugangsdaten aus einer vorherigen Anmeldung
    c. Service ID, die eine XS2A Datenquelle eindeutig kennzeichnet
2. Angabe einer gültigen Session

Account

Verfügbare Informationen für ein Bankkonto

```
{
    "AcctTpCd" = "CACC",
    "AcctCcy" = "EUR",
    "AcctIBAN" = "DE000000000000000",
    "AcctBIC" = "AAAADEBBB",
    "AcctCtry" = "DE",
    "AcctNo" = "1234567890",
    "AcctBankCode" = "10010010"
}
```

Mindestinformationen für ein Bankkonto mit IBAN

```
{
    "AcctTpCd" = "CACC",
    "AcctCcy" = "EUR",
    "AcctIBAN" = "DE000000000000000"
}
```

Mindestinformationen für ein Bankkonto ohne IBAN

```
{
    "AcctTpCd" = "CACC",
    "AcctCcy" = "EUR",
    "AcctCtry" = "DE",
    "AcctNo" = "1234567890",
    "AcctBankCode" = "10010010"
}
```

Benötigte Informationen für Kreditkarten

```
{
    "AcctTpCd" = "CRDC",
    "AcctCcy" = "EUR",
    "AcctNo" = "1234567812345678"
}
```

Bei Kreditkarten ist für die AcctNo die Kreditkartennummer anzugeben.

Orders (XS2A)

Array mit gültigen XS2A Aufträgen - eine allgemeine Beschreibung finden Sie unter dem Punkt Orders

ResponseOptions

Für XS2A Verbindungen kann festgelegt werden, ob die Antwort einen Trace (sofern verfügbar) für den vorangegangenen Request enthalten soll.

Darüber hinaus kann gesteuert werden, ob die Response den finalen Zustand des Contact Objekts liefern soll. Das gespeicherte Contact Objekt kann dann für weitere Anmeldungen in Verbindung mit den benötigten Credentials wiederverwendet werden.

Feldname	**Typ**		**Beschreibung**
Trace	Boolean	O	Wird dieses Feld mit dem Wert *true* übergeben, so wird in den Antwortdaten ein komplettes Übertragungsprotokoll für Diagnosezwecke geliefert.
Contact	Boolean	O	Wird dieses Feld mit dem Wert *true* übergeben, so wird in den Antwortdaten ein serialisierte Contact Feld eingestellt.

BerlinGroup

Kontozugriffe auf Basis des BerlinGroup Endpoints basieren auf folgender Request-Struktur:

- Request Options
- Orders
- Response Options

Die grundsätzliche Struktur eines BerlinGroup-Requests entspricht folgendem Aufbau, wobei nicht alle gezeigten Elemente verpflichtend oder in dieser Kombination gültig sind.

```
{
    "RequestOptions": {
        "RequestId": ...,
        "AccessToken": ...
    },
    "Orders": [
        {
            "OrderId": "xxx",
            "Type": "xxx",
            ...
        }, ...
    ],
    "ResponseOptions": {
        "Trace": ...,
        "BASInstanceName": ...
    }
}
```

Request Options

Angabe der RequestId und eines Access Tokens für die aktuelle BerlinGroup Anfrage. Weitere Informationen zu den Request Optionen finden Sie hier.

Orders

Array mit gültigen BerlinGroup Aufträgen - eine allgemeine Beschreibung finden Sie unter dem Punkt Orders

Response Options

In den Response Options kann festgelegt werden, ob die Antwort einen vollständigen BerlinGroup Trace für den vorangegangenen BerlinGroupRequest enthalten soll.

Feldname	**Typ**		**Beschreibung**
Trace	Boolean	O	Wird dieses Feld mit dem Wert *true* übergeben, so wird in den Antwortdaten ein komplettes Trace der über die BerlinGroup API durchgeführten Kommunikation (Request/Response) für Diagnosezwecke geliefert.

SEPA / Info

Requests für die Konvertierung von SEPA Dateien und die fachlichen Konvertierungen/Validierungen etc. über dieses Endpoints benötigen keine explizite Anmeldung. Die Requests beinhalten somit nur die eigentlichen Orders:

Requestaufbau:

```
{
     "RequestOptions": {
        "RequestId": ...,
        "AccessToken": ...
     },
    "Orders": [
        {
            "OrderId": "xxx",
            "Type": "xxx",
            ...
        }, ...
    ]
}
```

Request Options

Angabe der RequestId und eines Access Tokens für die aktuelle SEPA / Info Anfragen. Weitere Informationen zu den Request Optionen finden Sie hier.

Orders (SEPA / Info)

Array mit gültigen SEPA / Info Aufträgen - eine allgemeine Beschreibung finden Sie unter dem Punkt Orders

Request Options

Über die Request Options die Security Credentials (Access Token) und eindeutige RequestIds angegeben werden.

Feldname	**Typ**		**Beschreibung**
RequestId	string	O	Die RequestId stellt einen Identifier für eine Anfrage dar und ermöglicht eine Zuordnung zwischen dem Request und den jeweils enthaltenen Orders. Die Verwendung von RequestIds vereinfacht das Lesen der Log-Files ganz erheblich.
AccessToken	string	O	Sofern eine Zugriffssteuerung für den BankAccessServer aktivieret ist muss ein gültiges Access Token angegeben werden.

Orders

Im Block *Orders* wird ein Array mit Aufträgen übergeben, welche im Kontext der vorstehenden Verbindung ausgeführt werden sollen. Für jeden Auftrag wird ein eigenes Auftragsobjekt eingestellt. Jeder Auftrag kann die folgenden Felder enthalten. Der grundsätzliche Aufbau einer Order ist unabhängig vom verwendeten Endpoint immer gleich.

Feldname	**Typ**		**Beschreibung**
OrderId	string	O	Vom Client vergebene, beliebige Auftrags-ID. Diese wird in den Antwortdaten gespiegelt und unterstützt so die Zuordnung von Antwortdaten zum Auftrag im Client.
Type	string	M	Die gewünschte Auftragsart.
...			Auftragsspezifische Werte

Zugriffssteuerung

Ungeachtet einer abweichenden Konfiguration des vorgeschalteten Proxy Servers können nach einer Standardinstallation des BankAccessServers Requests für alle verfügbaren Endpoints ausgeführt werden.

Für die Definition individueller Zugriffsrechte kann der BankAccessServer mit individuellen API Keys, die wiederum auf bestimmte Endpoints beschränkt werden können, erstellt werden.

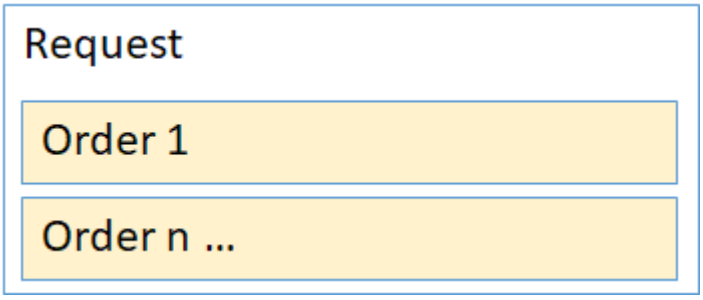
Ob bei den Requests die Gültigkeit von API Keys geprüft werden soll kann in der Datei appsettings.json im Abschnitt “BasConfig” definiert werden:

```
"BasConfig": {
      ...
      "CheckAccessToken": true | false
}
```

Weitere Informationen zur Zugriffssteuerung und der Generierung von Access Tokens finden Sie im Handbuch “BankAccessServer - Installation und Betrieb”.

Fehlerbehandlung

Ein Request kann grundsätzlich mehrere Aufträge (Orders) enthalten. Von daher gibt es auch zwei Hierarchiestufen auf den Fehler auftreten können.



Schlägt der gesamte Request fehl, z.B. aufgrund falscher Zugangsdaten oder fehlender Bankzugangsdaten, wird dieses Problem durch die HTTP Fehler 400 oder 500 ausgedrückt.

Der HTTP Fehler 400 kommt immer dann zum Tragen wenn ein Fehler bereits im BankAccessServer festgestellt werden kann und somit keine Kommunikation mit externen (Bank-)Rechnern stattfindet. Das ist z.B. der Fall wenn ein Auftrag für eine nicht vorhandene Bankleitzahl ausgeführt werden soll oder nicht alle notwendigen Auftragsparameter angegeben wurden.

Im vorgenannten Beispiel mit einer ungültigen Bankleitzahl würde der BankAccessServer keine gültige Kommunikationsadresse für die angefragte Bank ermitteln können und eine Fehler 400 (Bad Request) mit folgenden Zusatzinformationen bereitstellen:

```
{
   "Message": "'FinContact.CommAddress' is Null",
   "SymbolicName": "BAS_BANK_NOT_SUPPORTED"
}
```

Neben einem erklärenden Fehlertext (Message) wird auch ein SymbolicName zurückgeliefert, der dem aufrufenden Programm eine programmatische Fehlerbehandlung ermöglicht.

Diesem Schema folgend würde z.B. ein Auftrag mit fehlenden Login-Information wie folgt mit einem 400er Fehler quittiert werden:

```
{
   "Message": "[ Login | Resume ] is Null",
   "SymbolicName": "BAS_INVALID_REQUEST"
}
```

Demgegenüber werden Aufträge, die vom Remotesystem fehlerhaft verarbeitet werden mit einem HTTP Fehler 500 beantwortet. Das ist z.B. der Fall bei der Angabe fehlerhafter Logindaten.

```
{
   "Message": "'FinService.LogOn' Failed",
   "SymbolicName": "BAS_WRONG_PASSWORD"
}
```

Je nach Endpoint (FinTS/XS2A/EBICS) kann die Antwort zusätzlich protokoll spezifische Informationen enthalten.

Neben dem Szenario, in dem der gesamte Request fehlschlägt, kann es vorkommen, dass ledigliche einzelne Orders nicht ausgeführt werden können. Das kann z.B. im Fall einer erfolgreichen Anmeldung und der Angabe ungültiger Kontodaten bei der Umsatzabfrage vorkommen. Der gesamte Request wird mit dem HTTP Status 200 quittiert, die einzelne Response beinhaltet dann den auftragsspezifischen Fehler.

Beispiel:

```
{
   "ErrorCode": 400,
   "Message": "FinServiceResult.Error [9010 Der Auftrag wurde nicht
   ausgeführt.; 9010 Die Auftraggeber-IBAN ist falsch. (MOB90000000067)].",
   "SymbolicName": "BAS_HOST_ERROR",
   "Type": "FinTSOrderErrorResponse",
}
```

Auch hier können Endpoint-spezifische Zusatzinformationen bereitgestellt werden.

Symbolic Names

Die Auswertung der SymbolicNames ermöglicht eine programmatische Fehlerbehandlung. Eine gültige Übersicht aller SymbolicNames finden Sie in der Definition innerhalb des Swagger Files.

Nachfolgend eine Übersicht wichtiger Symbolic Names:

Symbolic Name	Fehlerszenario
BAS_NOT_AUTHORIZED	Ungültige BAS-Lizenz. Ungültiges Zugriffstoken. Fehlende Berechtigungen bei EBICS Requests.
BAS_INVALID_REQUEST	Ungültige Auftragsstruktur.
BAS_INVALID_PARAMS	Fehlende Parameter, die als verpflichtend gekennzeichnet sind.
BAS_INVALID_USER_STATE	Ungültiger Auftragsstatus, z.B. wenn ein Resume nicht möglich ist.
BAS_WRONG_PASSWORD	Fehlerhafte PIN/Zugangsdaten.
BAS_INTERNAL_ERROR	Unerwarteter Server Fehler.
BAS_INVALID_KEYDATA	Ungültige EBICS Schlüsseldaten.
BAS_BANK_NOT_SUPPORTED	Das angegebene Kreditinstitut wird nicht unterstützt.
BAS_ORDER_NOT_SUPPORTED	Ungültige/nicht erlaubte Order, z.B. weil diese für das Institut oder das ausgewählte Konto nicht möglich ist.
BAS_HOST_ERROR	Der Zielrechner hat einen Fehler zurückgeliefert.
BAS_DIALOG_NOT_ONLINE	FinTS Dialog ist nicht online

FinTS-Orders

Allgemeines

SCA-pflichtige Aufträge

Grundsätzlich muss man zwischen einfachen Abholaufträgen und SCA-pflichtigen Aufträgen unterscheiden. Darüber hinaus sind die Besonderheiten im Rahmen der von der PSD2 geforderten starken Kundenauthentifzierung zu beachten.

Bei SCA-pflichtigen Aufträgen endet der Request mit der TAN-Anforderung durch die Bank bzw. der Aufforderung zur Auftragsfreigabe in einem Decoupled Verfahren. In der Response werden alle für die SCA-Ermittlung erforderlichen Informationen (TAN-Challenge, Parameter des verwendeten TAN-Verfahrens und die Auftrags-Token, Challenge Prompt) mitgeteilt. In diesem Fall muss der Client diese Informationen aufbereiten, dem Benutzer anzeigen und die Eingabe einer TAN anfordern.

Die vom Benutzer eingegebene TAN wird dann im folgenden Request zusammen mit dem Auftrags-Token) an die Bank übermittelt und der Auftrag im Anschluss ausgeführt. Bei der Verwendung eines Decoupled Verfahrens sendet der Client im Anschluss die Information, dass die Freigabe erfolgt ist.

Auftraggeberkonto (OrderAcct)

Praktisch alle Auftragsarten benötigen die Angabe eines Auftraggeberkontos. Dieses sollte möglichst vollständig spezifiziert werden, damit es eindeutig identifiziert werden kann. Idealerweise werden alle Daten für das Auftraggeberkonto aus einem vorherigen *FinContactInfoRequest* übernommen. Das Auftraggeberkonto wird immer als *OrderAcct* Objekt im Auftrag eingestellt.

```
"OrderAcct": {
        "AcctNo": "1234567890",
        "BankCode": "99999999",
        "Currency": "EUR",
        "IBAN": "DE00999999991234567890",
        "BIC": "BANKDEZZ"
}
```

Feldname	Typ		Beschreibung
AcctNo	string	M	Die nationale zehnstellige Kontonummer.
BankCode	string	M	Die nationale achtstellige Bankleitzahl *des Kontos*. Deutschlandweit agierende Banken

			haben oft mehrere Bankleitzahlen und führen die verschiedene Konten eines Kunden unter verschiedenen Bankleitzahlen. Deshalb ist es wichtig, dass hier immer die zu einem Konto zugehörige Bankleitzahl angegeben wird und nicht eine beliebig andere Bankleitzahl des gleichen Kreditinstituts.
Currency	string	C	Die Kontowährung. Diese muss angegeben werden, wenn sie zur Unterscheidung erforderlich ist. So kommt es bei einigen Banken vor, dass unterschiedliche Konten mit verschiedenen Währungen unter der gleichen Kontonummer geführt werden.
IBAN	string	C	Die IBAN des Kontos. Diese muss immer angegeben werden, wenn für das Konto eine IBAN existiert. Währungskonten, Wertpapierdepots und andere spezielle Kontoarten haben oft keine IBAN und werden ausschließlich über die nationale Kontonummer identifiziert. In diesem Fall kann und muss keine IBAN angegeben werden.
BIC	string	M	Die BIC der Bank.

Folgende direkte FinTS Auftragsarten werden aktuell vom Subsembly BankAccessServer unterstützt.

Erweiterte Fehlerhinweise

Im Fall eines fehlerhaften FinTS Requests wird dieser mit einem FinTSError beantwortet und enthält folgende Informationen.

Feldname	**Typ**	**Beschreibung**
Message	string	Fehlermeldung
SymbolicName	string	Fehlerklassifizierun, z.B. "BAS_WRONG_PASSWORD" bei der Verwendung ungültiger Zugangsdaten
Trace	string	HBCI Trace, sofern in den Response Options aktiviert
HIRMSStatus	array	Statusmeldungen (Code, DEG, GD, Parameter) für die einzelnen Segmente

HIRMGStatus	array	Statusmeldungen (Code, DEG, GD, Parameter) für die gesamte Nachricht

Beispiel: Fehlerhafte Anmeldung bei der Postbank aufgrund falscher Zugangsdaten:


```
{
    "Message": "{'FinBanking.BeginDialog' Failed, Wrong Pin}
{;0000;\"Anmeldung...\"\r\n;0000;\"Dialoginitialisierung
senden...\"\r\nHIRMG;9800;\"Dialogabbruch.\"\r\nIIRMG;9942;\"PIN ungültig\"\r\n}
{9942}",
    "SymbolicName": "BAS_WRONG_PASSWORD",
    "HIRMSStatus": [
        {
            "Code": 9942,
            "DEG": 0,
            "GD": 0,
            "Text": "PIN ungültig"
        }
    ],
    "HIRMGStatus": [
        {
            "Code": 9800,
            "DEG": 0,
            "GD": 0,
            "Text": "Dialogabbruch."
        }
    ]
}
```

Performance Informationen

Bei der Verarbeitung/Ausführung von FinTS Requests sind verschiedene Instanzen involviert. Die Clientanwendung schickt einen FinTS Request an den BankAccessServer, der die Anfrage unter Verwendung der FinTS API aufbereitet und an die Bank schickt und nach der bankseitigen Verarbeitung das Ergebnis an die aufrufende Anwendung zurückgibt.

Im Rahmen der FinTS Response können Performancedaten zurückgegeben werden, die wiederum wertvolle Hinweise zum Zeitverbrauch geben.

Feldname	**Typ**	**Beschreibung**
BeginFinTSSessions	int	Anzahl aktiver FinTS Sessions im BAS
BeginAllSessions	int	Anzahl aller aktiven Sessions im BAS
MaxConnections	int	Maximale Anzahl Connections
MaxThreads	int	Maximale Anzahl Threads
TotalRequest	int	Dauer des Requests in ms
FinTSRequest	int	Zeitverbrauch des Requests in ms für die Ausführung der FinTS Anfrage
BASRequest	int	Zeitverbrauch des Requests im BAS in ms
BeginDialog	Object	Zeitverbrauch in ms für den FinTS Dialogaufbau (Total, FinTSSetup, FinTSResponse, FinTSProcess, Overhead)
BeginDialogSendTan	Object	Zeitverbrauch in ms für den FinTS Dialogaufbau im Rahmen einer SCA (Total, FinTSSetup, FinTSResponse, FinTSProcess, Overhead)
Orders	array	Zeitverbrauch der einzelnen Orders (Total, FinTSSetup, FinTSResponse, FinTSProcess, Overhead) unter Angabe der jeweiligen OrderId und der Auftragsart (Type)
TermDialog	Object	Zeitverbrauch in ms für das Beenden des FinTS Dialogs (Total, FinTSSetup, FinTSResponse, FinTSProcess, Overhead)

Beispiel FinTS Anmeldung Postbank:


```
"Performance": {
        "BeginFinTSSessions": 1,
        "BeginAllSessions": 1,
```

```
        "MaxConnections": 32,
        "MaxThreads": 64,
        "TotalRequest": 688,
        "FinTSRequest": 672,
        "BASRequest": 16,
        "BeginDialog": {
            "Total": 672,
            "FinTSSetup": 16,
            "FinTSResponse": 125,
            "FinTSProcess": 16,
            "Overhead": 515
        }
    }
```


Beispiel Saldenabfrage Postbank


```
"Performance": {
        "BeginFinTSSessions": 1,
        "BeginAllSessions": 1,
        "MaxConnections": 32,
        "MaxThreads": 64,
        "TotalRequest": 609,
        "FinTSRequest": 594,
        "BASRequest": 15,
        "Orders": [
            {
                "Performance": {
                    "Total": 594,
                    "FinTSSetup": 16,
                    "FinTSResponse": 531,
                    "FinTSProcess": 0,
                    "Overhead": 47
                },
                "Type": "FinAcctBalRequest",
                "OrderId": "460"
            }
        ]
    }
```

Beispiel FinTS Abmeldung Postbank


```
    "Performance": {
        "BeginFinTSSessions": 1,
        "BeginAllSessions": 1,
        "MaxConnections": 32,
        "MaxThreads": 64,
        "TotalRequest": 344,
        "FinTSRequest": 328,
        "BASRequest": 16,
        "TermDialog": {
            "Total": 328,
            "FinTSSetup": 31,
            "FinTSResponse": 281,
            "FinTSProcess": 16,
            "Overhead": 0
        }
    }
```

Kontaktinformationen abrufen

Liefert eine Liste aller verfügbaren Konten (UPD) und TAN-Verfahren (HITAN) eines Kunden sowie Informationen zu den möglichen Aufträgen / Auftragseigenschaften (BPD).

Beispiel

```
{
        "OrderId": "xxx"
        "Type": "FinTSContactInfoRequest",
}
```

FinTS Geschäftsvorfälle

In der nachfolgenden Tabelle sind die möglichen FinTS Geschäftsvorfälle aufgeführt. Grundsätzlich wird empfohlen, vor der Ausführung eines Geschäftsvorfalls zu prüfen, ob dieser bei dem ausgewählten Konto unterstützt ist.

FinOrderBuilder	FinTS GV	Info
FinAcctBalBuilder	HKSAL	Saldenabfrage
FinAcctMvmtsSpecifiedPeriodBuilder	HKKAZ	Umsatzabfrage MT940/942
FinAcctStmtBuilder	HKEKA	Elektronischer Kontoauszug
FinAcctStmtListBuilder	HKKAU	Übersicht Kontoauszüge
FinAcctStmtPdfBuilder	HKEKP	Elektronischer Kontoauszug PDF
FinBuildingSavingsInfosBuilder	DKBKU	Übersicht Bausparkonten
FinBuildingSavingsMvmtsBuilder	DKBUM	Umsätze Bausparvertrag
FinCamtStatementBuilder	HKCAZ	Umsatzabfrage CAMT 052
FinChangePinBuilder	HKPAE	PIN Änderung
FinCreditCardBalBuilder	DKKKS	Abfrage Kreditkartensalden
FinCreditCardMvmtsBuilder	DKKKU	Abfrage Kreditkartenumsätze
FinCustMsgBuilder	HKKDM	Kundenmitteilung
FinPortfListBuilder	HKWPD	Depotaufstellung
FinPrepaidLoadBuilder	HKPPD	Prepaidkarte laden
FinRevokePinBlockBuilder	HKPSA	PIN Sperre aufheben
FinSepaAllStoBuilder	HKCDB	Daueraufträge abfragen
FinSepaCancelPostdatedBusinessSingDirDebBuilder	HKBSL	Terminierte Firmenlastschriften löschen

FinSepaCancelPostdatedSingDirDebBuilder	HKDSL	Terminierte Einzellastschriften löschen
FinSepaCancelPostdatedSingRemittBuilder	HKCSL	Terminierte Einzelüberweisungen löschen
FinSepaDirectDebitIndicationListBuilder	HKDSB	Bestand der rückgabefähigen Lastschriften abrufen
FinSepaDirectDebitRejectBuilder	HKDSW	Lastschriftrückgabe wegen Widerspruch
FinSepaMultRemittBuilder	HKCCM	Sammelüberweisung
FinSepaSingRemittBuilder	HKCCS	Einzelüberweisung
FinSepaCancelStoBuilder	HKCDL	Dauerauftragslöschung
FinSepaModifyStoBuilder	HKCDN	Dauerauftragsänderung
FinSepaSetupStoBuilder	HKCDE	Dauerauftrag anlegen
FinSepaSubmitPostdatedBusinessMultDirDebBuilder	HKBME	Einreichung terminierte Firmensammellastschrift
FinSepaSubmitPostdatedBusinessSingDirDebBuilder	HKBSE	Einreichung terminierte Firmeneinzellastschrift
FinSepaSubmitPostdatedMultDirDebBuilder	HKDME	Einreichung terminierte Sammellastschrift
FinSepaSubmitPostdatedSingDirDebBuilder	HKDSE	Einreichung terminierte Einzellastschrift
FinSepaSubmitPostdatedMultRemittBuilder	HKCME	Terminierte Sammelüberweisung
FinSepaSubmitPostdatedSingRemittBuilder	HKCSE	Terminierte Einzelüberweisung
FinSepaOutstandingPostdatedBusinessSingDirDebBuilder	HKBBS	Bestand terminierte Firmenlastschriften
FinSepaOutstandingPostdatedSingDirDebBuilder	HKDBS	Bestand terminierte Einzellastschriften
FinSepaOutstandingPostdatedSingRemittBuilder	HKCSB	Bestand terminierte Überweisungen
FinSepaReclassAcctListBuilder	HKCUM	Umbuchung Referenzkonto
FinSepaInstPaymtBuilder	HKIPZ	Echtzeitüberweisung
FinTanMediaListBuilderProperties	HKTAB	TAN Medien abrufen
FinSepaDirectDebitIndicationListBuilderProperties	HIDSB	Rückgabefähige Lastschriften abrufen
FinSepaDirectDebitRejectBuilderProperties	HKDSW	Lastschriftwiderspruch
FinSepaInstPaymtStatusBuilderProperties	HKIPS	Statusabruf Sammelüberweisung
FinSepaMultInstPaymtBuilderProperties	HKISS	Echtzeitsammelüberweisung
FinSepaMultInstPaymtStatusBuilderProperties	HKISS	Statusabruf Echtzeit-Sammler

FinWhitelistListBuilderProperties	HKPWP	Abruf der Whiteliste-Einträge
FinWhitelistModifyBuilderProperties	HKPWA	Whitelist Eintrag aktualisieren
FinWhitelistRegisterBuilderProperties	HKPWE	Whitelist Eintrag registrieren
FinWhitelistRevokeBuilderProperties	HKPWL	Whitelist Eintrag löschen
FinSepaPostdatedMultInstPaymtBuilderProperties	HKIPE	Terminierte Echtzeit-Sammelüberweisung
FinMailDownloadBuilderProperties	HKKAA	Nachricht aus Postfach abrufen
FinMailListBuilderProperties	HKPOF	Nachrichtenliste für Postfach abrufen
FinTaxExemptionListBuilderProperties	HKFRD	Freistellungsaufträge abrufen
FinAcctDetailsBuilderProperties	HKKIF	Kontoinformationen abrufen.
FinSepaAcctInfoBuilderProperties	HKSPA	SEPA Kontoverbindungen abfragen
FinPortfAcctStatementReqBuilderProperties	HKWDU	Abfrage der Wertpapierumsätze im MT-536 Format
FinPortfOrderStatReqBuilderProperties	HKWSO	Abfrage des Orderstatus im MT-513 / MT-515 Format

SCA-pflichtige Geschäftsvorfälle

SCA/TAN-Medium festlegen

Legt das TAN-Medium fest, z.B. die Telefonnummer beim SMS-TAN Verfahren oder die Chipkarte beim Chip-TAN Verfahren.

Das explizite Festlegen des TAN-Mediums ist nur dann erforderlich, falls für den Endanwender bei der Bank mehrere aktive TAN-Medien für ein TAN-Verfahren hinterlegt sind und im Contact Objekt kein TAN-Medium vorab ausgewählt wurde.

Wird ein TAN-Medium über diese Auftragsart festgelegt, so wird diese im Contact gespeichert und ist somit auch für alle weiteren Aufträge festgelegt.

Eine Liste aller verfügbaren TAN-Medien kann über den separaten Auftrag FinTanMediaListRequest abgerufen werden.

Das Festlegen des TAN-Mediums ist zwingend erforderlich wenn zuvor ein TAN-pflichtiger Auftrag mit einer FinTSNeedTanMediaNameResponse quittiert wird.

Beispiel

```
{
    "OrderId": "xxx",
    "Type": "FinTSSetTanMediaNameRequest",
    "TanMediaName": "yyy"
```

```
}
```

TAN-Medien abrufen

Abrufen der verfügbaren TAN-Medien eines Kunden (HKTAB)

Beispiel Order

```
{
        "OrderId": "xxx",
        "Type": "FinTanMediaListRequest"
}
```

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinTanMediaListBuilder.html

TAN senden

Sendet eine TAN für einen zuvor eingereichten Auftrag (HKTAN). Das Senden der TAN ist erforderlich sofern ein TAN-pflichtiger Auftrag mit einer FinTSNeedTanResponse quittiert wurde.

Beispiel Order

```
{
        "OrderId": "xxx",
        "Type": "FinTSSendTanRequest",
        "Tan": "666666",
        "Order": "<base64enc>"
}
```

Decoupled Bestätigung senden

Sendet die Bestätigung, dass der zugrunde liegende Auftrag mit einem Decoupled Verfahren freigegeben wurde. Der FinTSSendDecoupledRequest ist zu senden, sofern der eigentliche Auftrag mit einer FinTSNeedDecoupledResponse beantwortet wurde.

Beispiel Order

```
{
      "OrderId": "{{$randomInt}}",
      "Type": "FinTSSendDecoupledRequest",
      "Order": "{{Order}}"
}
```

Salden- und Umsatzabruf

Kontensalden abrufen

Abrufen der aktuellen Salden und Kreditrahmen eines Kontos (HKSAL).

Beispiel Order

```
{
	"OrderId": "xxx",
	"Type": "FinAcctBalRequest",
	"OrderAcct": { ... },
	"AllAcct": false
}
```

Weitere Infos unter: https://subsembly.com/apidoc/Subsembly.FinTS.FinAcctBalBuilder.html

Kontoumsätze im MT-940 Format abrufen

Abrufen der Kontoumsätze im Swift MT-940-Format (HKKAZ).

Beispiel

```
{
	"OrderId": "xxx",
	"Type": "FinAcctMvmtsSpecifiedPeriodRequest",
	"OrderAcct": { ... },
	"StartDate": "2017-04-15",
	"EndDate": "2017-04-30"
}
```

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinAcctMvmtsSpecifiedPeriodBuilder.html

Kontoumsätze im CAMT Format abrufen

Abrufen der Kontoumsätze im ISO 20022 CAMT-Format (HKCAZ)

Beispiel

```
{
        "OrderId": "xxx",
        "Type": "FinCamtStatementRequest",
        "OrderAcct": { ... },
        "StartDate": "2017-04-15",
        "EndDate": "2017-04-30"
}
```

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinCamtStatementBuilder.html

SEPA Überweisungen / Umbuchungen

SEPA Einzelüberweisung

Einreichen einer SEPA Einzelüberweisung über den HBCI Geschäftsvorfall HKCCS.

Der SEPA Auftrag wird im Order Objekt als SEPA Dokument eingestellt. Dabei kann das komplette SEPA Dokument als JSON oder als Base64 codiertes XML Dokument eingestellt werden.

Das Element *RequestedExecutionDate* im SEPA Auftrag wird ignoriert und durch eine sofortige Ausführung ersetzt.

Beispiel Order JSON

```
{
        "OrderId": "xxx",
        "Type": "FinSepaSingRemittRequest",
        "OrderAcct": { ... },
        "SepaDocument": { ... }
}
```

Beispiel Order XML/Base64 encoded

```
{
        "OrderId": "xxx",
        "Type": "FinSepaSingRemittRequest",
        "OrderAcct": { ... },
        "SepaDocumentSerialized": "<base64enc>"
}
```

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaSingRemittBuilder.html

SEPA Sammelüberweisung

Einreichen einer SEPA Sammelüberweisung über den HBCI Geschäftsvorfall HKCCM.

Identisch wie SEPA Einzelüberweisung, jedoch wird der Auftragstyp "FinSepaSubmitMultRemittRequest" verwendet.

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaMultRemittBuilder.html

Terminierte SEPA Einzelüberweisung

Einreichen einer SEPA Einzelüberweisung über den HBCI Geschäftsvorfall HKCSE.

Identisch wie SEPA Einzelüberweisung, jedoch wird der Auftragstyp "FinSepaSubmitPostdatedSingRemittRequest" verwendet. Das gewünschte Ausführungsdatum wird im Element *RequestedExecutionDate* in den SEPA Auftragsdaten festgelegt.

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaSubmitPostdatedSingRemittBuilder.html

Terminierte SEPA Sammelüberweisung

Erstellt eine terminierte SEPA-Sammelüberweisung (HKCME)

Identisch wie SEPA Sammelüberweisung, jedoch wird der Auftragstyp "FinSepaSubmitPostdatedMultRemittRequest" verwendet.

Weiter Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaSubmitPostdatedMultRemittBuilder.html

SEPA Umbuchung

SEPA Umbuchung auf Referenzkonto (HKCUM)

Identisch wie SEPA Einzelüberweisung, jedoch wird der Auftragstyp "FinSepaReclassRequest" verwendet. Das Element *RequestedExecutionDate* im SEPA Auftrag wird ignoriert und durch eine sofortige Ausführung ersetzt.

Weitere Infos unter:

https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaReclassBuilder.html

Bestand terminierter SEPA Überweisungen abfragen

Abfrage aller für ein Konto vorhandenen SEPA Daueraufträge (HKCSB).

Beispiel Request:

```
{
	"OrderId": "xxx",
	"Type":"FinSepaOutstandingPostdatedSingRemittRequest",
	"OrderAcct": {
		"AcctNo": "",
		"BankCode": "",
		"BIC": "",
		"CountryCode": "280",
		"Currency": "EUR",
		"HolderName": "KLAUS IGEL",
		"IBAN": ""
	}
}
```

Die Response enthält dann folgende Informationen:

Feldname	**Typ**		**Beschreibung**
Remitts	array	O	Array der vorliegenden terminierten SEPA-Terminüberweisungen FinOrderID: Eindeutige ID im Backend SepaDocument

Weitere Infos unter:

https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaOutstandingPostdatedSingRemittBuilder.html

Terminierte SEPA Überweisung löschen

Löschung einer bestehende terminierten SEPA Überweisung (HKCSL). Für die Löschung müssen neben dem Auftraggeberkonto (OrderAccount) auch die Ausführungsdetails, das SEPA Document sowie die FinOrderID, unter dem der Auftrag im Backend identifiziert werden kann, angegeben werden.

Beispiel Request:

```
{
  "OrderId": "xxx",
  "Type": "FinSepaCancelPostdatedSingRemittRequest",
  "OrderAcct": {
    ...
  },
  "FinOrderID": "45480702181909300118",
  "SepaDocument": {
    "Message": {
          ...
     }
  }
}
```

Bei der Löschung eines terminierten SEPA Überweisung handelt es sich in der Regel um einen TAN-pflichtigen Geschäftvorfall, siehe TAN senden

Weitere Infos unter:

https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaCancelPostdatedSingRemittBuilder.html

SEPA Echtzeitüberweisung

Einreichen einer SEPA Echtzeitüberweisung über den HBCI Geschäftsvorfall HKIPZ.

Der SEPA Auftrag wird im Order Objekt als SEPA Dokument eingestellt. Dabei kann das komplette SEPA Dokument als JSON oder als Base64 codiertes XML Dokument eingestellt werden.

Das Element *RequestedExecutionDate* im SEPA Auftrag wird ignoriert und durch eine sofortige Ausführung ersetzt.

Hinweis: Aktuell unterstützen nur eine Reihe an Sparkassen SCT Inst Überweisungen über FinTS und bepreisen diese zum Teil separat. Ferner ist darauf zu achten, dass das Empfängerkonto ebenfalls SCT Inst fähig sein muss, andernfalls kann keine Echtzeitzahlung vorgenommen werden.

Beispiel Order JSON

```
{
        "OrderId": "xxx",
        "Type": "FinSepaInstPaymtRequest",
        "OrderAcct": { ... },
        "SepaDocument": { ... }
}
```

Beispiel Order XML/Base64 encoded

```
{
        "OrderId": "xxx",
        "Type": "FinSepaInstPaymtRequest",
        "OrderAcct": { ... },
        "SepaDocumentSerialized": "<base64enc>"
}
```

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaInstPaymtBuilder.html

Statusabfrage SEPA Echtzeitüberweisung

Abfrage des Status einer eingereichten SEPA Echtzeitüberweisung (HKIPS). Hierfür ist die zurückgelieferte OrderId aus der FinSepaInstPaymtResponse als Parameter für die FinOrderId anzugeben.

Beispiel Order:

```
{
    "OrderId": "{{$randomInt}}",
    "Type": "FinSepaInstPaymtStatusRequest",
    "OrderAcct": {{Acct}},
    "FinOrderID": "{{OrderId}}"
}
```

Die Antwort beinhaltet den Status für den angegebenen Auftrag.

```
{
    "OrderStatus": "Success",
    "Type": "FinSepaInstPaymtStatusResponse",
    "OrderId": "876",
    "FinOrderID": "..."
}
```

Mögliche Order Stati sind:

"InSchedule",
"RejectByIntermediary",
"InProcess",
"Processed",
"InRecall",
"Failed",
"Success",
"RejectByOriginatorBank",
"RejectByBenificiaryBank"

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaInstPaymtStatusBuilder.html

SEPA Echtzeit Sammelüberweisungen

Einreichen von SEPA Echtzeit Sammelüberweisungen über den HBCI Geschäftsvorfall HKIPM.

Der SEPA Auftrag wird im Order Objekt als SEPA Dokument eingestellt. Dabei kann das komplette SEPA Dokument als JSON oder als Base64 codiertes XML Dokument eingestellt werden.

Beispiel Order JSON

```
{
	"OrderId": "xxx",
	"Type": "FinSepaMultiInstPaymtRequest",
	"OrderAcct": { ... },
	"SepaDocument": { ... }
}
```

Beispiel Order XML/Base64 encoded

```
{
	"OrderId": "xxx",
	"Type": "FinSepaMultiInstPaymtRequest",
	"OrderAcct": { ... },
	"SepaDocumentSerialized": "<base64enc>"
}
```

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaMultInstPaymtBuilder.html

Terminierte SEPA Echtzeit Sammelüberweisungen

Einreichen von terminierten SEPA Echtzeit Sammelüberweisungen über den HBCI Geschäftsvorfall HKIPE.

Der SEPA Auftrag wird im Order Objekt als SEPA Dokument eingestellt. Dabei kann das komplette SEPA Dokument als JSON oder als Base64 codiertes XML Dokument eingestellt werden.

Beispiel Order JSON

```
{
	"OrderId": "xxx",
	"Type": "FinSepaPostdatedMultiInstPaymtRequest",
	"OrderAcct": { ... },
	"SepaDocument": { ... }
}
```

Beispiel Order XML/Base64 encoded

```
{
	"OrderId": "xxx",
	"Type": "FinSepaMultiInstPaymtRequest",
	"OrderAcct": { ... },
	"SepaDocumentSerialized": "<base64enc>"
}
```

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaMultInstPaymtBuilder.html

Statusabfrage SEPA Sammelüberweisung

Abfrage des Status einer eingereichten SEPA Sammelüberweisung (HKISS). Hierfür ist die zurückgelieferte OrderId aus der FinSepaInstPaymtResponse als Parameter für die FinOrderId anzugeben.

Beispiel Order:

```
{
    "OrderId": "{{$randomInt}}",
    "Type": "FinSepaMultiInstPaymtStatusRequest",
    "OrderAcct": {{Acct}},
    "FinOrderID": "{{OrderId}}"
}
```

Die Antwort beinhaltet den Status für den angegebenen Auftrag.

```
{
    "OrderStatus": "Success",
    "Type": "FinSepaInstPaymtStatusResponse",
    "OrderId": "876",
    "FinOrderID": "..."
}
```

Mögliche Order Stati sind:

"InSchedule",
"RejectByIntermediary",
"InProcess",
"Processed",
"InRecall",
"Failed",
"Success",
"RejectByOriginatorBank",
"RejectByBenificiaryBank"

Lastschriften

SEPA Einzellastschrift

Einreichen einer SEPA Einzellastschrift vom Typ CORE über den HBCI Geschäftsvorfall HKDSE.

Identisch wie SEPA Einzelüberweisung, jedoch wird der Auftragstyp "FinSepaSubmitPostdatedSingDirDebRequest" verwendet und das übergebene SEPA Dokument muss eine einzelne CORE Lastschrift enthalten.

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaSubmitPostdatedSingDirDebBuilder.html

SEPA Sammellastschrift

Einreichen einer SEPA Sammellastschrift vom Typ CORE über den HBCI Geschäftsvorfall HKDME.

Identisch wie SEPA Sammelüberweisung, jedoch wird der Auftragstyp "FinSepaSubmitPostdatedMultDirDebRequest" verwendet und das übergebene SEPA Dokument muss Lastschriften vom Typ CORE enthalten.

Achtung: Einige Banken erfordern, dass ein mit diesem Geschäftsvorfall eingereichter Auftrag mindestens zwei Lastschriften enthält. Um nur eine einzelne Lastschrift einzureichen muss deshalb zwingend der Auftragstyp "FinSepaSubmitPostdatedSingDirDebRequest" verwendet werden.

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaSubmitPostdatedMultDirDebBuilder.html

SEPA Firmeneinzellastschrift

Einreichen einer SEPA Einzellastschrift vom Typ B2B über den HBCI Geschäftsvorfall HKBSE.

Identisch wie SEPA Einzelüberweisung, jedoch wird der Auftragstyp "FinSepaSubmitPostdatedBusinessSingDirDebRequest" verwendet und das übergebene SEPA Dokument muss eine einzelne B2B Lastschrift enthalten.

Weitere Infos unter:

https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaSubmitPostdatedMultDirDebBuilder.html

SEPA Firmensammellastschrift

Einreichen einer SEPA Sammellastschrift vom Typ B2B über den HBCI Geschäftsvorfall HKBME.

Identisch wie SEPA Einzelüberweisung, jedoch wird der Auftragstyp "FinSepaSubmitPostdatedBusinessMultDirDebRequest" verwendet und das übergebene SEPA Dokument muss B2B Lastschriften enthalten.

Achtung: Einige Banken erfordern, dass ein mit diesem Geschäftsvorfall eingereichter Auftrag mindestens zwei Lastschriften enthält. Um nur eine einzelne Firmenlastschrift einzureichen muss deshalb zwingend der Auftragstyp "FinSepaSubmitPostdatedBusinessSingDirDebRequest" verwendet werden.

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaSubmitPostdatedBusinessMultDirDebBuilder.html

Bestand terminierter SEPA Lastschriften abfragen

Abfrage aller der für ein Konto vorhandenen SEPA Einzel- und Firmeneinzellastschriften (HKDBS/HKBBS).

Beispiel Request:

```
{
        "OrderId": "xxx",
        "Type":"FinSepaOutstandingPostdatedSingDirDebRequest |
                FinSepaOutstandingPostdatedBusinessSingDirDebRequest",
        "OrderAcct": {
                "AcctNo": "",
                "BankCode": "",
                "BIC": "",
                "CountryCode": "280",
                "Currency": "EUR",
                "HolderName": "KLAUS IGEL",
                "IBAN": ""
        }
}
```

Die Response enthält dann folgende Informationen:

Feldname	**Typ**		**Beschreibung**
Remitts	array	O	Array der vorliegenden terminierten SEPA-Lastschriften FinOrderID: Eindeutige ID im Backend SepaDocument

Weitere Infos unter:

https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaOutstandingPostdatedSingDirDebBuilder.html

https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaOutstandingPostdatedBusinessSingDirDebBuilder.html

Terminierte SEPA Lastschriften löschen

Löschung einer bestehende terminierten SEPA Einzel- und Firmeneinzellastschrift (HKBSL / HKDSL) . Für die Löschung müssen neben dem Auftraggeberkonto (OrderAccount) auch die Ausführungsdetails, das SEPA Document sowie die FinOrderID, unter dem der Auftrag im Backend identifiziert werden kann, angegeben werden.

Beispiel Request:

```
{
  "OrderId": "xxx",
  "Type": "FinSepaCancelPostdatedSingDirDebRequest |
           FinSepaCancelPostdatedSingBusinessSingDirDebRequest",
  "OrderAcct": {
    ...
  },
  "FinOrderID": "45480702181909300118",
  "SepaDocument": {
    "Message": {
         ...
     }
  }
}
```

Bei der Löschung eines terminierten SEPA Überweisung handelt es sich in der Regel um einen TAN-pflichtigen Geschäftvorfall, siehe TAN senden

Weitere Infos unter:

https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaCancelPostdatedSingDirDebBuilder.html

https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaCancelPostdatedBusinessSingDirDebBuilder.html

Bestand rückgabefähiger Lastschriften abrufen

Abruf der rückgabefähigen Lastschriften (HKDSB) durchführen. Für den Abruf muss ein gültiges Auftraggeberkonto (OrderAccount) angegeben werden. Optional können, sofern bankseitig unterstützt, noch Datumseinschränkungen vorgenommen werden.

Beispiel Request:

```
{
    "OrderId": "{{$randomInt}}",
    "Type": "FinSepaDirectDebitIndicationListRequest",
    "OrderAcct": {{Acct}}
}
```

Die Antwort enthält ein Array der rückgabefähigen Lastschriften (FinSepaDirectDebitIndicationRS).

Weitere Infos unter:

https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaDirectDebitIndicationListBuilder.html

Rückgabe einer Lastschrift wegen Widerspruchs

Rückgabe einer Lastschriften (HKDSW) wegen Widerspruchs. Die benötigten Auftragsdetails können über den FinSepaDirectDebitIndicationListRequest abgerufen werden.

Beispiel Request:

```
{
    "OrderId": "{{$randomInt}}",
    "Type": "FinSepaDirectDebitRejectRequest",
    "DirDebInd": {
        "CustAcct": {{CustAcct}},
        "CreditorAcct": {{CreditorAcct}},
        "CreditorName": {{CreditorName}},
        "CreditorId": "{{CreditorId}}",
        "EntryDate": "{{EntryDate}}",
```

```
            "Amount": {
                "Currency": "EUR",
                "Value": 78.68
            },
            "PaymtPurpose": "{{PaymtPurpose}}",
            "MandateId": "{{MandateId}}",
            "ZkaCodes": {},
            "OrderID": "{{OrderId}}",
            "Rejected": false
        }
    }
```

Die Antwort enthält die Auftragsreferenz (OrderId), die im Rahmen einer SCA Transaktion (TAN-/Decoupled) freigegeben wird.

Weitere Infos unter:

https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaDirectDebitIndicationListBuilder.html

PIN Sperren / Änderungen

PIN Änderung

Durchführen einer PIN Änderung auf Basis der HBCI Geschäftsvorfälle DKPAE oder HKPAE.

Beispiel

```
{
        "OrderId": "xxx",
        "Type": "FinChangePinRequest",
        "NewPin": “12345
}
```

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinChangePinBuilder.html

PIN Sperre aufheben

Aufheben einer PIN-Sperre auf Basis der HBCI Geschäftsvorfälle DKPSA oder HKPSA.

Beispiel

```
{
        "OrderId": "xxx",
        "Type": "FinRevokePinBlockRequest",
}
```

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinRevokePinBlockBuilder.html

Postfachzugriff

Der Zugriff auf das Postfach bietet die Möglichkeit Liste der Nachrichten und den Inhalt einer spezifischen Nachricht abzurufen

Nachrichtenliste abrufen

Abruf der im Postfach vorhandenen Nachrichtenliste (HKPOF)

Beispiel

```
{
    "OrderId": "{{$randomInt}}",
    "Type": "FinMailListRequest",
    "StartDate": "{{StartDate}}"
}
```

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinMailDownloadBuilder.html

Inhalt einer einzelnen Nachricht abrufen

Der Abruf einer einzelnen Nachricht ist über den FinTS Geschäftsvorfall HKKAA möglich. Die Auswahl des Dokuments erfolgt über eine DocumentID, die beim Abruf der Nachrichtenliste bereitgestellt wird.

Beispiel

```
{
            "OrderId": "{{$randomInt}}",
            "Type": "FinMailDownloadRequest",
            "DocumentID": "12.08.2024 10:52:56:700307000"
}
```

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinMailDownloadBuilder.html

Kundennachricht

Kundennachricht auf Basis des HBCI Geschäftsvorfalls HKKDM an die Bank senden.

Beispiel

```
{
        "OrderId": "orderid",
        "Type": "FinCustMsgRequest",
        "OrderingCustAcct": {},
        "PlainTextMsg": "<Text>",
        "Subject": "<Betreff>",
        "Recipient": "<Empfänger>"
}
```

Weitere Infos unter: https://subsembly.com/apidoc/Subsembly.FinTS.FinCustMsgBuilder.html

Willenserklärung

Mit diesem Geschäftsvorfall ist es möglich, einen Vorgang des Kunden als Willenserklärung bestätigen zu lassen. Die benötigten Auftragsparameter können variieren. Weitere Infos unter: https://subsembly.com/apidoc/Subsembly.FinTS.FinConsentDeclarationBuilder.html

Prepaid Mobilfunkkarte laden

Lädt die Prepaid Mobilfunkkarte um den angegebenen Betrag auf (HKPPD).

Beispiel

```
{
        "OrderId": "xxx",
        "Type": "FinPrepaidLoadRequest",
        "OrderAcct": {},
        "ProviderNum": <Prepaid Provider Nummer>,
        "PhoneNumber": "Tel. mit führender 0",
        "LoadAmount": {
           "Currency" : "EUR",
           "Value" : <betrag>
        }
}
```

Weitere Infos unter:

https://subsembly.com/apidoc/Subsembly.FinTS.FinPrepaidLoadBuilder.html

Kontodetails abrufen

Mit dem Geschäftsvorfall können weitere Kontoinformationen wie Zinssätze für ein Konto abgerufen werden (HKKIF).

Beispiel

```
{
    "OrderId": "{{$randomInt}}",
    "Type": "FinAcctDetailsRequest",
    "OrderAcct": {{Acct}}
}
```

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinAcctDetailsBuilder.html

SEPA Kontoverbindungen abrufen

Mit dem Geschäftsvorfall können die SEPA Kontoverbindungen für den angemeldeten Benutzer abgefragt werden (HKSPA).

Beispiel

```
{
	"OrderId": "{{$randomInt}}",
	"Type": "FinSepaAcctInfoRequest",
	"OrderAcct": "{{Acct}}"
}
```

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaAcctInfoBuilder.html

Freistellungsaufträge abrufen

Mit dem Geschäftsvorfall können die bei der Bank hinterlegten Freistellungsaufträge abgerufen werden (HKFRD).

Beispiel

```
{
    "OrderId": "{{$randomInt}}",
    "Type": "FinTaxExemptionListRequest",
    "OrderAcct": {{Acct}}
}
```

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinTaxExcemtpionListBuilder.html

Kreditkartensalden und -umsätze abrufen

Kreditkartensalden abrufen

Abruf des Kreditkartensaldos für die angegebene Kreditkarten (DKKKS).

Beispiel

```
{
        "OrderId": "orderid",
        "Type": "FinCreditCardBalRequest",
        "OrderAcct": {},
        "CreditCardNumber": "<Kreditkartennummer>"
}
```

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinCreditCardBalBuilder.html

Kreditkartenumsätze abfragen

Abruf des Kreditkartenumsätze für die angegebene Kreditkarten im vorgegebenen Zeitraum (DKKKU).

Beispiel

```
{
        "OrderId": "orderid",
        "Type": "FinCreditCardMvmtsRequest",
        "OrderAcct": {},
        "CreditCardNumber": "<Kreditkartennummer>",
        "StartDate": "2017-05-15",
       "EndDate": "2017-05-30"
}
```

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinCreditCardMvmtsBuilder.html

Daueraufträge

Dauerauftragsbestand abfragen

Abfrage aller der für ein Konto vorhandenen SEPA Daueraufträge (HKCDB).

Beispiel Request:

```
{
	"OrderId": "xxx",
	"Type":"FinSepaAllStoRequest",
	"OrderAcct": {
		"AcctNo": "",
		"BankCode": "",
		"BIC": "",
		"CountryCode": "280",
		"Currency": "EUR",
		"HolderName": "KLAUS IGEL",
		"IBAN": ""
	}
}
```

Die Response enthält dann folgende Informationen:

Feldname	**Typ**		**Beschreibung**
AllSepaStos	array	O	Array der vorhanden Daueraufträge. Zu einem einzelnen Dauerauftrag liegen u.a. folgende Informationen vor: Details: Ausführungsdetails FinOrderID: Eindeutige ID im Backend SepaDocument

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaAllStoBuilder.html

Neuen Dauerauftrag anlegen

Anlage eines neuen SEPA Dauerauftrags (HKCDE). Für die Anlage müssen neben dem Auftraggeberkonto (OrderAccount) auch die Ausführungsdetails und das SEPA Document angegeben werden.

Beispiel Request:

```
{
  "OrderId": "xxx",
  "Type": "FinSepaSetupStoRequest",
  "OrderAcct": {
    ...
  },
  "Details": {
    "ExecDay": 30,
    "ExecuteFirstTimeOn": "2018-07-30",
    "PeriodLen": 1,
    "TimeUnit": "Month"
  },
  "SepaDocument": {
    "Message": {
          ...
     }
  }
}
```

Bei der Anlage eines Dauerauftrags handelt es sich in der Regel um einen TAN-pflichtigen Geschäftvorfall, siehe TAN senden

Die anschließende Response enthält die FinOrderId unter die der neue Dauerauftrag im Backendsystem geführt wird.

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaSetupStoBuilder.html

Bestehenden Dauerauftrag ändern

Änderung eines bestehenden SEPA Dauerauftrags (HKCDN). Für die Änderung müssen neben dem Auftraggeberkonto (OrderAccount) auch die Ausführungsdetails, das SEPA Document sowie die OrderID, unter dem der Auftrag im Backend identifiziert werden kann, angegeben werden.

Beispiel Request:

```
{
  "OrderId": "xxx",
  "Type": "FinSepaModifyStoRequest",
  "OrderAcct": {
    ...
  },
  "Details": {
    "ExecDay": 30,
    "ExecuteFirstTimeOn": "2018-07-30",
    "PeriodLen": 1,
    "TimeUnit": "Month"
  },
  "FinOrderID": "45480702181909300118",
  "SepaDocument": {
    "Message": {
          ...
     }
  }
}
```

Bei der Änderung eines Dauerauftrags handelt es sich in der Regel um einen TAN-pflichtigen Geschäftvorfall, siehe TAN senden

Die anschließende Response enthält die aktuelle FinOrderId, unter die der geänderte Dauerauftrag im Backendsystem geführt wird.

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaModifyStoBuilder.html

Bestehenden Dauerauftrag löschen

Löschung eines bestehenden SEPA Dauerauftrags (HKCDL). Für die Löschung müssen neben dem Auftraggeberkonto (OrderAccount) auch die Ausführungsdetails, das SEPA Document sowie die FinOrderID, unter dem der Auftrag im Backend identifiziert werden kann, angegeben werden.

Beispiel Request:

```
{
  "OrderId": "xxx",
  "Type": "FinSepaCancelStoRequest",
  "OrderAcct": {
    ...
  },
  "Details": {
    "ExecDay": 30,
    "ExecuteFirstTimeOn": "2018-07-30",
    "PeriodLen": 1,
    "TimeUnit": "Month"
  },
  "FinOrderID": "45480702181909300118",
  "SepaDocument": {
    "Message": {
          ...
     }
  }
}
```

Bei der Löschung eines Dauerauftrags handelt es sich in der Regel um einen TAN-pflichtigen Geschäftvorfall, siehe TAN senden

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinSepaCancelStoBuilder.html

Kontauszüge abrufen

Liste der vorliegenden Kontoauszüge holen

Abruf der elektronischen Kontoauszügen im angegebenen Format (HKKAU)

Beispiel Request:

```
{
	"OrderId": "xxx",
	"Type": "FinAcctStmtListRequest",
	"OrderAcct": {
		"AcctNo": "",
		"BankCode": "",
		"BIC": "",
		"CountryCode": "280",
		"Currency": "EUR",
		"HolderName": "KLAUS IGEL",
		"IBAN": ""
	}
}
```

Request:

Feldname	**Typ**		**Beschreibung**
OrderAcct	FinAcctRS	M	Angabe des Kontos für das der Kontoauszug abgerufen werden soll
MaxNoEntries	int	O	Maximale Anzahl der Einträge
ScrollRef	string	O	Referenz einer vorherigen Abfrage

Die Response enthält die Liste der vorliegenden Kontoauszüge. Sofern das Flag “CanDownload” true ist kann der eigentliche Kontoauszug heruntergeladen werden.

Beispiel Response:

```
"AcctStmts": [
     {
      "BuildDate": "2018-07-31",
      "StmtYear": 2018,
      "StmtNo": 2,
      "AcknowledgementStatus": "Required",
      "CanDownload": true
```

Weitere Infos unter:

https://subsembly.com/apidoc/Subsembly.FinTS.FinAcctStmtListBuilder.html

Elektronische Kontoauszüge abrufen

Abruf von elektronischen Kontoauszügen im angegebenen Format (HKEKA)

Beispiel Request:

```
{
	"OrderId": "xxx",
	"Type": "FinAcctStmtRequest",
	"StmtFormat": "PDF",
	"StmtNo": 1,
	"StmtYear": 2018,
	"OrderAcct": {
		"AcctNo": "",
		"BankCode": "",
		"BIC": "",
		"CountryCode": "280",
		"Currency": "EUR",
		"HolderName": "KLAUS IGEL",
		"IBAN": ""
	}
}
```

Anhand der Response bei der Abfrage der Kontaktinformationen ist ersichtlich, welche Parameter beim Abruf der Kontoauszüge angegeben werden können und welche Formate unterstützt werden, z.B.:

```
"StmtNoAllowed": true,
"MaxEntriesAllowed": false,
"StmtFormats": [
  "PDF"
],
"Type": "FinAcctStmtBuilderProperties"
```

Request:

Feldname	**Typ**		**Beschreibung**
OrderAcct	FinAcctRS	M	Angabe des Kontos für das der Kontoauszug abgerufen werden soll
StmtFormat	string	O	Null, ISO8583, MT940, PDF
StmtNo	int	O	Nummer des angefragten Auszugs
StmtYear	int	O	Jahr des angefragten Auszugs
MaxNoEntries	int	O	Maximale Anzahl der Einträge
ScrollRef	string	O	Referenz einer vorherigen Abfrage

Die Response enthält dann folgende Informationen:

Feldname	**Typ**		**Beschreibung**
AcctStmts	array	O	Array der zurückgelieferten Kontoauszüge

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinAcctStmtBuilder.html

Elektronische Kontoauszüge im PDF Format abrufen

Abruf von elektronischen Kontoauszügen im PDF Format (HKEKP)

Beispiel Request:

```
{
	"OrderId": "xxx",
	"Type": "FinAcctStmtPDFRequest",
	"StmtNo": 1,
	"StmtYear": 2018,
	"OrderAcct": {
		"AcctNo": "",
		"BankCode": "",
		"BIC": "",
		"CountryCode": "280",
		"Currency": "EUR",
		"HolderName": "KLAUS IGEL",
		"IBAN": ""
	}
}
```

Anhand der Response bei der Abfrage derKontaktinformationen ist ersichtlich, welche Parameter beim Abruf der Kontoauszüge angegeben werden können.

```
"StmtNoAllowed": true,
"MaxEntriesAllowed": false

"Type": "FinAcctStmtPDFBuilderProperties"
```

Request:

Feldname	**Typ**		**Beschreibung**
OrderAcct	FinAcctRS	M	Angabe des Kontos für das der Kontoauszug abgerufen werden soll
StmtNo	int	O	Nummer des angefragten Auszugs
StmtYear	int	O	Jahr des angefragten Auszugs
MaxNoEntries	int	O	Maximale Anzahl der Einträge
ScrollRef	string	O	Referenz einer vorherigen Abfrage

Die Response enthält dann folgende Informationen:

Feldname	**Typ**		**Beschreibung**
AcctStmts	array	O	Array der zurückgelieferten Kontoauszüge

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinAcctStmtPdfBuilder.html

Wertpapierdepot

Wertpapierdepotaufstellung abrufen

Abruf der Wertpapierdepotaufstellung (HKWPD) für das angegebene Konto.

Beispiel Request:

```
{
	"OrderId": "xxx",
	"Type": "FinPortfListRequest",
	"OrderAcct": {
		"AcctNo": "",
		"BankCode": "",
		"BIC": "",
		"CountryCode": "280",
		"Currency": "EUR",
		"HolderName": "KLAUS IGEL"
	}
}
```

Request:

Feldname	**Typ**		**Beschreibung**
OrderAcct	FinAcctRS	M	Angabe des Kontos für das der Kontoauszug abgerufen werden soll
MaxNoEntries	int	O	Maximale Anzahl der Einträge
ScrollRef	string	O	Referenz einer vorherigen Abfrage
PortfolioSecurListCcy	string	O	Angabe der Währung falls unterstützt
ExchRateQuality	FinExchRate QualityRS	O	Angabe des Kurstyps falls unterstützt

Die Response enthält die Aufstellung mit der Liste der im Depot befindlichen Wertpapiere.

Weitere Infos unter: https://subsembly.com/apidoc/Subsembly.FinTS.FinPortfListBuilder.html

Abfrage der Wertpapierumsätze

Abruf der Wertpapierumsätze im MT-536 Format (HKWDU) für das angegebene Konto.

Beispiel Request:

```
{
	"OrderId": "xxx",
	"Type": "FinPortfAcctStatementReqRequest",
	"OrderAcct": {
		"AcctNo": "",
		"CountryCode": "280",
		"Currency": "EUR",
		"HolderName": "KLAUS IGEL"
	}
}
```

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinPortfAcctStatementReqBuilder.html

Orderstatus abfragen

Abfrage des Orderstatus im MT-513 / MT-515 Format (HKWSO) für das angegebene Konto.

Beispiel Request:

```
{
	"OrderId": "xxx",
	"Type": "FinPortfOrderStatReqRequest",
	"OrderAcct": {
		"AcctNo": "",
		"CountryCode": "280",
		"Currency": "EUR",
		"HolderName": "KLAUS IGEL"
	}
}
```

Weitere Infos unter:
https://subsembly.com/apidoc/Subsembly.FinTS.FinPortfOrderStatReqBuilder.html

Bausparkonten

Kontenübersicht

Ruft die Kontenübersicht des Bausparkontos ab (DKBKU).

Request:

Feldname	**Typ**		**Beschreibung**
OrderAcct	FinAcctRS	M	Angabe des Kontos für das die Übersicht abgerufen werden soll
MaxNoEntries	int	O	Maximale Anzahl der Einträge
ScrollRef	string	O	Referenz einer vorherigen Abfrage

Response:

Die FinBuildingSavingsInfosResponse enthält ein Array von FinBuildingSavingsInfoRS. FinBuildingSavingsInfoRS hat folgenden Aufbau:

Feldname	**Typ**	**Beschreibung**
ContractOwnerName	string	Konto-/Vertragsinhaber
ContractNumber	string	Konto-/Vertragsnummer
AcctType	BuildingSavingsAcctTypeRS	Kontotyp
ContractStatus	BuildingSavingsContractStatusRS	Vertragsstatus: Closed - Aufgelöst Deceased - Vertragsinhaber verstorben GarnishedClaim - bei einem Einzel-Bausparvertrag: Mahn- und Vollstreckungsfall NoSavingsDepositPossible - keine Sparzahlungen möglich NotClosed - nicht aufgelöst NotYetEffective - noch nicht in Kraft Null - Data element was not set. Single - Einzel-Bausparvertrag = DTV (HDA=9)
Balance	SwiftBalanceRS	Konto-/Vertragssaldo

Umsätze abrufen

Ruft die Umsätze eines Bausparkontos ab (DKBUM).

Request:

Feldname	**Typ**		**Beschreibung**
OrderAcct	FinAcctRS	M	Angabe des Kontos für das die Umsätze abgerufen werden sollen
AcctType	BuildingSavingsAcctTypeRS	M	Kontotyp
MaxNoEntries	int	O	Maximale Anzahl der Einträge
ScrollRef	string	O	Referenz einer vorherigen Abfrage

Response:

Feldname	**Typ**	**Beschreibung**
ContractCurrency	string	Konto-/Vertragswährung
ContractNumber	string	Konto-/Vertragsnummer
OpeningBalance	SwiftBalanceRS	Anfangssaldo
ClosingBalance	SwiftBalanceRS	Endsaldo
BuildingSavingsMvmts	FinBuildingSavingsMvmtRS[]	Umsatzinformationen, bestehend aus Betrag, Buchungsdatum, Buchungstext und Valutadatum

PSD2 Whitelist Management

Umfasst die FinTS Requests zur Abfrage, Erstellung, Änderung und Löschung von “white-gelisteten” Zielkonten, für die im Regelfall keine weitere SCA bei Überweisungen vorgenommen werden muss.

Whitelist Einträge abfragen

Ermöglicht die Abfrage der “white-gelisteten” Empfängerkonten (HKPWB).

Request: (FinWhitelistListRequest)

Feldname	**Typ**		**Beschreibung**
OrderAcct	FinAcctRS	M	Angabe des Kontos für das die freigeschalteten Empfängerkonten abgerufen werden sollen
MaxNoEntries	int	O	Maximale Anzahl der Einträge
ScrollRef	string	O	Referenz einer vorherigen Abfrage

Response:

Die FinWhitelistListResponse enthält ein Array von FinWhitelistEntryRS Elementen. FinWhitelistEntryRS hat folgenden Aufbau:

Feldname	**Typ**	**Beschreibung**
EntryId	string	Technische ID des Eintrags. Dieser Wert wird benötigt, um vorhandene Einträge zu ändern oder zu löschen.
EntryCaption	string	Vom Benutzer vergebene Bezeichnung für den Eintrag. Dieser Wert kann vom PayeeName (Empfängername) abweichen.
PayeeName	string	Vom Benutzer vergebene Empfängername.
PayeeAcct	FinAcctRS	Angaben zum Empfängerkonto, z.B. Kontonummer, Bankleitzahl, IBAN, BIC.

Neues Empfängerkonto white-listen

Erstellt eine SCA Ausnahme für das angegebene Empfängerkonto (HKPWE).

Request: (FinWhitelistRegisterRequest)

Feldname	**Typ**		**Beschreibung**
OrderAcct	FinAcctRS	M	Angabe des Kontos für das Zahlungen an das Empfängerkonto ohne weitere SCA ermöglicht werden sollen.
EntryCaption	string	O	Vom Benutzer vergebene Bezeichnung für den Eintrag. Dieser Wert kann vom PayeeName (Empfängername) abweichen.
PayeeName	string	O	Vom Benutzer vergebene Empfängername.
PayeeAcct	FinAcctRS	M	Angaben zum Empfängerkonto, z.B. Kontonummer, Bankleitzahl, IBAN, BIC.

Response:

Die FinWhitelistRegisterResponse enthält die technische ID, die in der HIWPE Antwort bereitgestellt wurde.

Feldname	**Typ**	**Beschreibung**
EntryId	string	Technische ID des Eintrags. Dieser Wert wird benötigt, um vorhandene Einträge zu ändern oder zu löschen.

Whitelisting Eintrag ändern

Änderung eines bestehenden Whitelisting Eintrags (HKPWA). Die Identifikation des Whitelist-Eintrags erfolgt anhand der EntryId, die bei der Anlage bzw. beim Abruf zurückgeliefert wird.

Request: (FinWhitelistModifyRequest)

Feldname	**Typ**		**Beschreibung**
OrderAcct	FinAcctRS	M	Angabe des Kontos für das Zahlungen an das Empfängerkonto ohne weitere SCA ermöglicht werden sollen.
EntryCaption	string	O	Vom Benutzer vergebene Bezeichnung für den Eintrag. Dieser Wert kann vom PayeeName (Empfängername) abweichen.
PayeeName	string	O	Vom Benutzer vergebene Empfängername.
PayeeAcct	FinAcctRS	M	Angaben zum Empfängerkonto, z.B. Kontonummer, Bankleitzahl, IBAN, BIC.

Response:

Die FinWhitelistRegisterResponse enthält die technische ID, die in der HIWPE Antwort bereitgestellt wurde.

Feldname	**Typ**	**Beschreibung**
EntryId	string	Technische ID des Eintrags. Dieser Wert wird benötigt, um vorhandene Einträge zu ändern oder zu löschen.

EBICS Aufträge und Geschäftsvorfälle

Der BankAccessServer EBICS ist grundsätzlich in der Lage, alle zwischen dem Kunden und seiner Bank vereinbarten Geschäftsvorfälle durchzuführen. Wie in den beiden vorherigen Abschnitten bereits angesprochen, unterstützt der BankAccessServer die wichtigsten Auftragsarte in Form von individuellen Requests mit einem entsprechenden Mapping der Klassen und Bereitstellung der Daten im JSON Format.

Alle weiteren EBICS Auftragsarten können über den EbicsOrderGenericRequest unter Angabe des OrderTypes, der Information ob es sich um einen Abhol- oder Sendeauftrag handelt (Upload: true | false) und der jeweiligen Auftragsdaten/-parameter durchgeführt werden.

In der nachfolgenden Tabelle sind einige relevante Geschäftsvorfälle auf Basis EBICS 2.5 aufgelistet:

Geschäftsvorfall	**Upload / Download**	**Auftragsart**
PDF Tagesauszüge abrufen	D	BKA
SEPA-Basis-Lastschrift einreichen	U	CDD, CDC
SEPA-Firmen-Lastschrift einreichen	U	CDB, C2C
SEPA-Überweisungen einreichen	U	CCT, CCC
SEPA-Echtzeitüberweisung einreichen	U	CIP
Tagesauszüge abrufen	D	C53, STA
Vormerkposten abrufen	D	C52, VMK
Kundenprotokoll abrufen	D	HAC, PTK
VEU Status abrufen	D	HVD
VEU Unterschrift hinzufügen	U	HVE
VEU Storno	U	HVS
VEU Transaktionsdetails abrufen	D	HVT
VEU Übersicht abholen	D	HVU
VEU Übersicht von Zusatzinformationen	D	HVZ
Senden der Teilnehmerschlüssel	U	HIA, INI
Abholen der öffentlichen Bankschlüssel	D	HPB

Teilnehmerdaten abrufen	D	HTD

Bitte beachten Sie, dass nicht alle Banken die hier aufgezählten Geschäftsvorfälle unterstützen und diese vertraglich zwischen Bank und Kunde vereinbart werden..

Über den universellen EbicsOrderGenericRequest ist die Ausführung aller unterstützten Geschäftsvorfälle möglich.

Grundsätzlich wird über das Attribut “Upload” spezifiziert, ob Daten gesendet oder abgerufen werden sollen. Die zu sendenden Daten werden im Attribut “OrderData” definiert.

Der Antwort vom Type EbicsOrderGenericResponse können dann die jeweilige Informationen, z.B. “TechnicalReturnCode” oder “BusinessReturnCode”, entnommen werden. Download Orders enthalten die zurückgelieferten Daten im Attribut “OrderData”.

EBICS Admin Requests

Die EBICS Admin Requests dienen der Verwaltung von EBICS Zugängen/-schlüsseln und dürfen nur von Clients mit speziellen Admin-Rechten ausgeführt werden. Der Client muss in jedem Request einen Access-Token mit Admin-Berechtigung übergeben.

Initialisierung - EbicsAdminInitRequest

Initialisiert einen neuen EBICS Bankzugang, indem das übergebene EbicsContact Objekt vollständig initialisiert und zurückgeliefert wird.

Feldname	**Typ**		**Beschreibung**
Contact	EbicsContactRS	M	Es müssen mindestens die Pflichtwerte HostID, HostURL, PartnerID und UserID angegeben werden.
KeyPassword	string	M	Neues oder vorhandenes Schlüsselpasswort des Teilnehmers.
ContactOption	string	M	Wählt den Speicherort, an dem die Schlüssel gespeichert werden sollen. Zugelassen sind lediglich die Werte "BAS" für die Speicherung auf dem BAS und "CON" für die Speicherung innerhalb des EbicsContact Objekts.

Beispiel: Initialisierung eines neuen EBICS Zugangs und Speicherung auf dem BAS:

```
"Orders": [
  {
    "Contact": {
      "HostId": "DUMMY",
      "PartnerId": "XXXXXXXX",
      "UserId": "KLAUS1",
      "HostURL": "..."
    },
    "KeyPassword": "123456",
    "ContactOption": "BAS",
    "Type": "EbicsAdminInitRequest"
  }
```

Dieser Request wird nur lokal auf dem BAS ausgeführt. Es findet keine Kommunikation mit dem EBICS Zielsystem statt. Der *ReturnCode* wird deshalb nur lokal generiert und stammt nicht von einem EBICS Zielsystem.

Wird im EbicsContact keine EBICS Version übergeben, so wird der EbicsContact für die Standardversion H004 (EBICS 2.5) initialisiert.

Werden im EbicsContact keine Schlüsselversionen AuthenticationVersion, EncryptionVersion und SignatureVersion übergeben, so wird hierfür jeweils die zur vorher ermittelten EBICS Version passende Standardversion angenommen. Ist die resultierende Kombination aus EBICS-Version, AuthenticationVersion, EncryptionVersion und SignatureVersion ungültig, wird mit BAS_INVALID_PARAMS abgebrochen.

Alle noch fehlenden Kundenschlüssel werden unmittelbar generiert, so dass nach erfolgreichem Abschluss alle Kundenschlüssel verfügbar sind. Wurden neue Kundenschlüssel generiert, so ist im zurückgelieferten EbicsContact das boolsche Feld *IniLetterNeeded* mit dem Wert *true* enthalten.

Ein EbicsContact kann grundsätzlich in zwei verschiedenen Varianten initialisiert und verwendet werden: Die Schlüssel können auf dem BAS gespeichert werden (KeyOption ist "BAS"), oder die Schlüssel können im EbicsContact selbst gespeichert werden (KeyOption ist "CON").

Schlüsseldaten auf dem BAS

Die Schlüsseldaten werden auf dem BAS verwaltet und sind fest mit der Kombination aus HostID, UserID und PartnerID assoziiert. Das übergebene *KeyPassword* wird zur Verschlüsselung der Schlüsseldaten auf dem BAS verwendet. Die Schlüsseldaten werden auf dem BAS in einem gesicherten, nur vom BAS-Dienst zugreifbaren Speicherbereich gespeichert. Für jeden Mandanten wird ein separater Speicherbereich abhängig von seiner Client-ID verwendet. Auf dem BAS werden die Teilnehmerschlüssel immer als Schlüsseldatei im Subsembly Format gespeichert. (Über die Subsembly EBICS API Klasse EbicsSecurityMediumFile: https://subsembly.com/apidoc/ebics/Subsembly.EBICS.EbicsSecurityMediumFile.html).

Liegen auf dem BAS bereits Schlüsseldaten für den initialisierenden EBICS Teilnehmer vor, so werden diese einfach weiter verwendet. Das übergebene *KeyPassword* muss in diesem Fall zu den vorhandenen Schlüsseldaten passen. Ist das nicht der Fall, so wird mit BAS_WRONG_PASSWORD abgebrochen.

Passen die vorliegenden Schlüsseldaten nicht zur gewünschten EBICS-Version, so wird mit BAS_INVALID_REQUEST abgebrochen.

Liegen noch keine Schlüsseldaten vor, so wird ein kompletter, neuer Schlüsselsatz erzeugt und gespeichert. Dieser wird mit dem übergebenen *KeyPassword* gesichert.

Schlüsseldaten im EbicsContact

Die benötigten Schlüssel werden direkt im EbicsContact gespeichert und somit bei jedem Request vom Client zum BAS mit übertragen. Im übergebenen EbicsContact sollten demnach vorher noch keine Schlüssel enthalten sein. Der BAS erzeugt immer einen neuen Schlüsselsatz, sofern keine Schlüsseldatei übergeben wurde.

Die Schlüssel sind auch bei der Speicherung im EbicsContact mit dem übergebenen *KeyPassword* verschlüsselt.

Initialisierung - EbicsAdminInitResponse

Feldname	**Typ**		**Beschreibung**
ContactSerialized	string	O	Base64 encodierter EBICS Kontakt mit Schlüsseldaten. Für weitere Requests muss dieser EbicsContact verwendet werden. Wird bei Verwendung der ContactOption “CON” zurückgeliefert.

Schlüsseldatei importieren - EbicsAdminImportKeyRequest

Liegt vom Kunden bereits eine EBICS Schlüsseldatei in einem von der Subsembly EBICS API unterstütztem Format vor, so kann diese bei der Initialisierung als KeyData mitgegeben werden. Zusätzlich muss in diesem Fall das Passwort zur Schlüsseldatei im Feld KeyDataPassword übergeben werden.

Die Schlüsseldatei muss in einem von der Subsembly EBICS API unterstütztem Dateiformat vorliegen. Ist dies nicht der Fall, so wird mit BAS_INVALID_KEYDATA abgebrochen.

Kann die Schlüsseldatei nicht mit dem übergebenen KeyDataPassword entschlüsselt werden, so wird mit BAS_WRONG_PASSWORD abgebrochen.

Der BAS geht bei Vorliegen einer Schlüsseldatei davon aus, dass der EBICS Zugang bereits vollständig initialisiert und von der Bank freigeschaltet wurde und in der EBICS Schlüsseldatei alle Kundenschlüssel vorliegen. Die Kundenschlüssel werden aus der Schlüsseldatei ausgelesen und importiert.

Die Schlüsselattribute werden ebenfalls aus der Schlüsseldatei übernommen und überschreiben etwaige im Request eingestellte Schlüsselversionen. Sollten die in der EBICS Schlüsseldatei enthaltenen Schlüssel nicht zur gewählten EBICS Version passen, so wird mit BAS_INVALID_KEYDATA abgebrochen.

Ist die Schlüsseldatei unvollständig, inkonsistent oder aus anderem Grunde nicht verarbeitbar, so wird mit BAS_INVALID_KEYDATA abgebrochen.

Feldname	**Typ**		**Beschreibung**
Contact	EbicsContactRS	M	Es müssen mindestens die Pflichtwerte HostID, HostURL, PartnerID und UserID angegeben werden.
KeyPassword	string	M	Neues oder vorhandenes Schlüsselpasswort des Teilnehmers.
ContactOption	string	M	Wählt den Speicherort an dem die Schlüssel gespeichert werden sollen. Zugelassen sind lediglich die Werte "BAS" für die Speicherung auf dem BAS und "CON" für die Speicherung innerhalb des EbicsContact Objekts.
KeyData	string	M	Base64 kodierte Schlüsseldaten. Diese müssen einem von der Subsembly EBICS API unterstütztem Dateiformat entsprechen. Werden Schlüsseldaten übergeben, so werden alle Schlüssel daraus eingelesen und übernommen.
KeyDataPassword	string	M	Passwort zur Schlüsseldatei

Beispiel: Initialisierung eines neuen EBICS Zugangs und Speicherung auf dem BAS:

```
"Orders": [
  {
    "Contact": {
      "HostId": "DUMMY",
      "PartnerId": "KIG1",
      "UserId": "KLAUS1",
      "HostURL": "..."
    },
    "KeyPassword": "123456",
    "KeyData": "Base64 encoded",
    "KeyDataPassword": "123456",
    "ContactOption": "BAS",
    "Type": "EbicsAdminImportKeyRequest"
  }
```

Schlüsseldatei importieren - EbicsAdminImportKeyResponse

Feldname	**Typ**		**Beschreibung**
ContactSerialized	string	O	Base64 encodierter EBICS Kontakt mit Schlüsseldaten. Für weitere Requests muss dieser EbicsContact verwendet werden. Wird bei Verwendung der ContactOption “CON” zurückgeliefert.

EBICS Kontakt importieren - EbicsAdminImportRequest

Über den EbicsAdminImportRequest kann ein vollständig eingerichteter EBICS Zugang auch direkt im BankAccessServer verwendet werden. Sofern der EBICS Zugang mit einem Subsembly Produkt wie Banking ZV oder dem EBICS Admin eingerichtet wurde, kann dieser im XML Format exportiert werden. Für den EbicsAdminImportRequest muss das XML zuvor noch nach Base64 kodiert werden. Nach einem erfolgreichen Import erfolgt die Speicherung im BAS, so dass nachfolgende Aufträge über die “ContactRef” Connection ausgeführt werden können, z.B.

```
"Connection": {
        "ContactRef": {
            "HostId": "{{EbicsHostId}}",
            "PartnerId": "{{EbicsPartnerId}}",
            "UserId": "{{EbicsUserId}}",
            "HostURL": "{{EbicsHostUrl}}"
        },
        "KeyPassword": "{{EbicsKeyPassword}}"
    }
```

Feldname	Typ		Beschreibung
ContactSerialized	string	M	Base64 kodierten EBICS Zugang, der mit einem Subsembly Produkt wie BankingZV oder dem EBICS Admin erzeugt und exportiert wurde.
KeyPassword	string	M	Neues oder vorhandenes Schlüsselpasswort des Teilnehmers.

Beispiel: Import eines EBICS Zugangs:

```
  "Orders": [
        {
            "ContactSerialized": "{{EbicsContactSerialized}}",
            "KeyPassword": "{{EbicsKeyPassword}}",
            "Type": "EbicsAdminImportRequest"
        }
    ]
```

Die zurückgelieferte EbicsAdminImportResponse zeigt lediglich an, ob der Import erfolgreich war und beinhaltet keine weiteren Daten.

EBICS Kontakte auslesen - EbicsAdminExportAllRequest

Liest alle im BankAccessServer gespeicherten EBICS Kontakte aus und liefert die serialisierten Kontakte in der Antwort zurück. Wichtig: Zur Ausführung muss das verwendete AccessToken über die Berechtigung **ebics-remote-admin** verfügen:

Beispiel Request:

```
{
    "RequestOptions": {
        "AccessToken": "{{EbicsRemoteAdminAccessToken}}"
    },
    "Orders": [
        {
            "Type": "EbicsAdminExportAllRequest"
        }
    ]
}
```

Die Antwort enthält die serialisierten EBICS-Kontakte.

```
 "Responses": [
        {
            "SerializedContacts": [...],
            "Type": "EbicsAdminExportAllResponse"
        }
    ]
```

Aktualisierung eines EBICS Kontakts - EbicsAdminUpdateRequest

Aktualisiert einen im BankAccessServer gespeicherten EBICS Kontakt mit den übergebenen Werten für den EbicsContact. Praktische Anwendungsbeispiele sind die zu verwendende EBICS Version oder eine geänderte Host URL.

Beispiel: Änderung der EBICS Version:

```
"Orders": [
        {
            "Type": "EbicsAdminUpdateRequest",
            "Contact": {
                "Version": "H004"
            }
        }
    ]
```

Eine erfolgreiche Änderung wird durch EbicsAdminInitResponse quittiert.

```
{
    "Responses": [
        {
            "Type": "EbicsAdminInitResponse"
        }
    ]
}
```

EBICS Kontakt löschen - EbicsAdminKillRequest

Löscht die auf dem Server für den EBICS-Teilnehmer gespeicherten Schlüsseldaten.

Feldname	Typ		Beschreibung
ContactRef	EbicsContactRefRS	M	Referenz auf einen im BAS gespeicherten EBICS Kontakt.

Beispiel Order:

```
  "Orders": [
   {
     "ContactRef": {
       "HostId": "DUMMY",
       "PartnerId": "KIG1",
       "UserId": "KLAUS1"
     },
     "Type": "EbicsAdminKillRequest"
   }
 ]
```

Dieser Request wird nur lokal auf dem BAS ausgeführt. Es findet keine Kommunikation mit dem EBICS Zielsystem statt.

Für das Löschen der Schlüsseldaten ist kein Teilnehmerpasswort erforderlich. Somit können die Schlüsseldaten auch gelöscht werden, wenn der Teilnehmer sein Passwort vergessen hat.

Durch die Client-ID im Client-Access-Token ist gewährleistet, dass ein mandantenübergreifendes Löschen von Schlüsseldaten nicht möglich ist.

Öffentlichen Schlüssel abrufen - EbicsAdminGetPubKeyInfoRequest

Liefert den öffentlichen Schlüssel. Ein Client kann diesen Request z.B. nutzen, um die Daten zu erhalten, die er zum Generieren eines INI-Briefs benötigt. Alternativ kann auch ein vom BAS im PDF-Format generierter INI-Brief per EbicsAdminGenerateINILetterRequest abgerufen werden.

Feldname	Typ		Beschreibung
Connection	EbicsConnection	M	Angabe des EBICS Kontakts und Schlüsselpassworts - siehe: Connection

Beispiel:

```
"Connection": {
  "ContactRef": {
    "HostId": "DUMMY",
    "PartnerId": "KIG",
    "UserId": "KLAUS"
  },
  "KeyPassword" : "123456"
},
"Orders": [
  {
    "Type": "EbicsAdminGetPubKeyInfoRequest"
  }
]
```

Der Request wird nur lokal auf dem BAS ausgeführt. Es findet keine Kommunikation mit dem EBICS Zielsystem statt.

Öffentlichen Schlüssel abrufen - EbicsAdminGetPubKeyInfoResponse

Feldname	Typ		Beschreibung
PubKeyInfo	EbicsPubKeyInfoRS	C	Informationen zum öffentlichen Schlüssel als JSON *EbicsPubKeyInfo* Objekt. Darin ist, unter Anderem, der Hashwert (Fingerprint) des öffentlichen Schlüssels enthalten.

INI Brief erzeugen - EbicsAdminGenerateINILetterRequest

Erzeugt ein PDF mit kombinierten INI- und HIA-Brief aus den vorliegenden Schlüsseln.

Feldname	Typ		Beschreibung
Connection	EbicsConnection	M	Angabe des EBICS Kontakts und Schlüsselpassworts - siehe: Connection

Beispiel:

```
"Connection": {
  "ContactRef": {
    "HostId": "DUMMY",
    "PartnerId": "KIG",
    "UserId": "KLAUS"
  },
  "KeyPassword" : "123456"
},
"Orders": [
  {
    "Type": "EbicsAdminGenerateINILetterRequest"
  }
]
```

INI Brief erzeugen - EbicsAdminGenerateINILetterResponse

Feldname	Typ		Beschreibung
INILetter	string	C	Base-64-codierte Binärdaten, welche ein PDF mit dem INI-Brief enthalten.

Signaturschlüssel übertragen - EbicsAdminINIRequest

Überträgt den öffentlichen Signaturschlüssel des Teilnehmers an den EBICS Server in einem INI Auftrag. Der Teilnehmer muss auf dem EBICS Server im entsprechenden Zustand "NEU" oder "HIA" der Initialisierung sein.

Feldname	Typ		Beschreibung
Connection	EbicsConnection	M	Angabe des EBICS Kontakts und Schlüsselpassworts - siehe: Connection

Beispiel:

```
"Connection": {
  "ContactRef": {
    "HostId": "DUMMY",
    "PartnerId": "KIG",
    "UserId": "KLAUS"
  },
  "KeyPassword" : "123456"
},
"Orders": [
  {
    "Type": "EbicsAdminINIRequest"
  }
]
```

Signaturschlüssel übertragen - EbicsAdminINIResponse

Feldname	Typ		Beschreibung
TechnicalReturnCode	EbicsReturnCodeRS	M	Technischer EBICS Return-Code aus dem EBICS Response-Header.
BusinessReturnCode	EbicsReturnCodeRS	M	Bankfachlicher EBICS Return-Code aus dem EBICS Response-Body.

Authentifizierungs- und Verschlüsselungsschlüssel übertragen - EbicsAdminHIARequest

Überträgt die öffentlichen Authentifizierungs- und Verschlüsselungsschlüssel des Teilnehmers an den EBICS Server in einem HIA Auftrag. Der Teilnehmer muss auf dem EBICS Server im entsprechenden Zustand "NEU" oder "INI" der Initialisierung sein.

Feldname	Typ		Beschreibung
Connection	EbicsConnection	M	Angabe des EBICS Kontakts und Schlüsselpassworts - siehe: Connection

Beispiel:

```
"Connection": {
  "ContactRef": {
    "HostId": "DUMMY",
    "PartnerId": "KIG",
    "UserId": "KLAUS"
  },
  "KeyPassword" : "123456"
},
"Orders": [
  {
    "Type": "EbicsAdminHIARequest"
  }
]
```

Authentifizierungs- und Verschlüsselungsschlüssel übertragen - EbicsAdminHIAResponse

Feldname	Typ		Beschreibung
TechnicalReturnCode	EbicsReturnCodeRS	M	Technischer EBICS Return-Code aus dem EBICS Response-Header.
BusinessReturnCode	EbicsReturnCodeRS	M	Bankfachlicher EBICS Return-Code aus dem EBICS Response-Body.

Bankschlüssel - EbicsAdminHPBRequest

Lädt die aktuellen Bankschlüssel vom EBICS Banksystem und speichert diese im EbicsContact. Der EBICS Kontakt wird auf dem BAS automatisch aktualisiert (ContactOption=”BAS”) bzw. an den Aufrufer in der Response zurückgeliefert (ContactOption=”CON”).

Die Response liefert die Hashwerte der Bankschlüssel an den Aufrufer zurück. Der EBICS Teilnehmer muss für diesen Request bereits vom Kreditinstitut vollständig freigeschaltet worden sein.

Feldname	**Typ**		**Beschreibung**
Connection	EbicsConnection	M	Angabe des EBICS Kontakts und Schlüsselpassworts - siehe: Connection

Beispiel:

```
"Connection": {
  "ContactRef": {
    "HostId": "DUMMY",
    "PartnerId": "KIG",
    "UserId": "KLAUS"
  },
  "KeyPassword" : "123456"
},
"Orders": [
  {
    "Type": "EbicsAdminHPBRequest"
  }
]
```

Bankschlüssel - EbicsAdminHPBResponse

Feldname	**Typ**		**Beschreibung**
TechnicalReturnCode	EbicsReturnCodeRS	M	Technischer EBICS Return-Code aus dem EBICS Response-Header.
BusinessReturnCode	EbicsReturnCodeRS	M	Bankfachlicher EBICS Return-Code aus dem EBICS Response-Body.
ContactSerialized	string	C	aktualisierter EBICS Kontakt - nur bei Verwendung ContactOption=”CON”
BankAutPubKeyInfo	EbicsPubKeyInfoRS	M	Öffentlicher Authentifizierungsschlüssel der Bank. Darin ist, unter Anderem, der Hashwert (Fingerprint) des öffentlichen Schlüssels enthalten.
BankEncPubKeyInfo	EbicsPubKeyInfoRS	M	Öffentlicher Verschlüsselungsschlüssel der Bank. Darin ist, unter Anderem, der Hashwert (Fingerprint) des öffentlichen Schlüssels enthalten.

EBICS Order Requests

Ist ein EBICS Bankzugang vollständig initialisiert und bereit, so können normale EBICS Aufträge ausgeführt werden. Neben generischen Upload- und Download-Orders gibt es verschiedene Orders für spezielle Auftragsarten.

In EBICS ist jeder Auftrag letztendlich entweder ein Dateidownload oder Dateiupload. Für die Ausführung von EBICS Aufträgen über den BAS muss der Client ein Access-Token mit der Berechtigung Download und/oder Upload präsentieren.

Teilnehmerberechtigung - EbicsOrderHTDRequest

Liefert Informationen zu den Konten und Berechtigungen eines EBICS Teilnehmers. Zur Ausführung dieses Auftrags wird die Download Berechtigung benötigt.

Feldname	**Typ**		**Beschreibung**
Connection	EbicsConnection	M	Angabe des EBICS Kontakts und Schlüsselpassworts - siehe: Connection

Beispiel:

```
  "Connection": {
    "ContactRef": {
      "HostId": "DUMMY",
      "PartnerId": "KIG",
      "UserId": "KLAUS"
    },
    "KeyPassword" : "123456"
  },
  "Orders": [
    {
      "Type": "EbicsOrderHTDRequest"
    }
  ]
```

Teilnehmerberechtigung - EbicsOrderHTDResponse

Feldname	**Typ**		**Beschreibung**
TechnicalReturnCode	EbicsReturnCodeRS	M	Technischer EBICS Return-Code aus dem EBICS Response-Header.
BusinessReturnCode	EbicsReturnCodeRS	M	Bankfachlicher EBICS Return-Code aus dem EBICS Response-Body.
Weitere auftragsbezogene Daten			

Protokollabruf - EbicsOrderPTKRequest

Liefert das PTK Protokoll. Zur Ausführung dieses Auftrags wird die Download Berechtigung benötigt.

Feldname	**Typ**		**Beschreibung**
DateTimeFrom	string	O	Von Datum yyyy-mm-dd
DateTimeTo	string	O	Bis Datum yyyy-mm-dd

Beispiel:

```
"Connection": {
  "ContactRef": {
    "HostId": "DUMMY",
    "PartnerId": "KIG",
    "UserId": "KLAUS"
  },
  "KeyPassword" : "123456"
},
"Orders": [
  {
   "Type":"EbicsOrderPTKRequest",
   "DateTimeFrom": "2018-04-11",
   "DateTimeTo": "2018-04-12"

  }
]
```

Protokollabruf - EbicsOrderPTKResponse

Feldname	**Typ**		**Beschreibung**
TechnicalReturnCode	EbicsReturnCodeRS	M	Technischer EBICS Return-Code aus dem EBICS Response-Header.
BusinessReturnCode	EbicsReturnCodeRS	M	Bankfachlicher EBICS Return-Code aus dem EBICS Response-Body.
ProtocolText	string	C	Das vollständige Originalprotokoll
ProtocolEntries	array EbicsProtocolEntryRS	C	Die aus dem Originalprotokoll geparsten einzelnen Einträge werden hier als ein Array mit Objekten des Typs EbicsProtocolEntryRS eingestellt.

Tagesauszüge abrufen - EbicsOrderSTARequest

Abholen der SWIFT Tagesauszüge im MT940 Format. Zur Ausführung dieses Auftrags wird die Download Berechtigung benötigt.

Feldname	Typ		Beschreibung
DateTimeFrom	string	O	Von Datum yyyy-mm-dd
DateTimeTo	string	O	Bis Datum yyyy-mm-dd

Beispiel:

```
"Connection": {
  "ContactRef": {
    ...
  },
  "KeyPassword" : "123456"
},
"Orders": [
  {
   "Type":"EbicsOrderSTARequest",
   "DateTimeFrom": "2018-04-11",
   "DateTimeTo": "2018-04-12"
  }
]
```

Tagesauszüge abrufen - EbicsOrderSTAResponse

Feldname	Typ		Beschreibung
TechnicalReturnCode	EbicsReturnCodeRS	M	Technischer EBICS Return-Code aus dem EBICS Response-Header.
BusinessReturnCode	EbicsReturnCodeRS	M	Bankfachlicher EBICS Return-Code aus dem EBICS Response-Body.
BookedTrans	SwiftStatementRS	C	Gebuchte Transaktionen im MT940 Format

Untertägige Umsätze abrufen - EbicsOrderC52Request

Abruf von Intraday Kontoumsätzen (Bank to Customer Account Report). Zur Ausführung dieses Auftrags wird die Download Berechtigung benötigt.

Feldname	Typ		Beschreibung
DateTimeFrom	string	O	Von Datum yyyy-mm-dd
DateTimeTo	string	O	Bis Datum yyyy-mm-dd

Beispiel:

```
"Connection": {
  "ContactRef": {
    ...
  },
  "KeyPassword" : "123456"
},
"Orders": [
  {
   "Type":"EbicsOrderC52Request",
   "DateTimeFrom": "2018-05-01"
  }
]
```

Untertägige Umsätze abrufen - EbicsOrderC52Response

Feldname	Typ		Beschreibung
TechnicalReturnCode	EbicsReturnCodeRS	M	Technischer EBICS Return-Code aus dem EBICS Response-Header.
BusinessReturnCode	EbicsReturnCodeRS	M	Bankfachlicher EBICS Return-Code aus dem EBICS Response-Body.
AdvicesStatement	array SepaDocumentRS	C	Tagesumsätze / Vormerkposten

Kontoauszüge abrufen - EbicsOrderC53Request

Abruf von gebuchter Kontoauszüge (Bank to Customer Statement). Zur Ausführung dieses Auftrags wird die Download Berechtigung benötigt.

Feldname	**Typ**		**Beschreibung**
DateTimeFrom	string	O	Von Datum yyyy-mm-dd
DateTimeTo	string	O	Bis Datum yyyy-mm-dd

Beispiel:

```
"Connection": {
  "ContactRef": {
    ...
  },
  "KeyPassword" : "123456"
},
"Orders": [
  {
   "Type":"EbicsOrderC53Request",
   "DateTimeFrom": "2018-05-01"
  }
]
```

Kontoauszüge abrufen - EbicsOrderC53Response

Feldname	**Typ**		**Beschreibung**
TechnicalReturnCode	EbicsReturnCodeRS	M	Technischer EBICS Return-Code aus dem EBICS Response-Header.
BusinessReturnCode	EbicsReturnCodeRS	M	Bankfachlicher EBICS Return-Code aus dem EBICS Response-Body.
AccountStatement	array SepaDocumentRS	C	Tagesauszug

Notifications - EbicsOrderC54Request

Anzeige Gutschriften / Belastungen (Bank to Customer Debit Credit Notification). Zur Ausführung dieses Auftrags wird die Download Berechtigung benötigt.

Feldname	Typ		Beschreibung
DateTimeFrom	string	O	Von Datum yyyy-mm-dd
DateTimeTo	string	O	Bis Datum yyyy-mm-dd

Beispiel:

```
"Connection": {
  "ContactRef": {
    ...
  },
  "KeyPassword" : "123456"
},
"Orders": [
  {
   "Type":"EbicsOrderC54Request",
   "DateTimeFrom": "2018-05-01"
  }
]
```

Notifications - EbicsOrderC54Response

Feldname	Typ		Beschreibung
TechnicalReturnCode	EbicsReturnCodeRS	M	Technischer EBICS Return-Code aus dem EBICS Response-Header.
BusinessReturnCode	EbicsReturnCodeRS	M	Bankfachlicher EBICS Return-Code aus dem EBICS Response-Body.
DebitCreditNotification	array SepaDocumentRS	C	Anzeige Gutschriften / Belastungen

Kundeninformationen - EbicsOrderHKDRequest

Abruf der Kunden-/Teilnehmerinformationen. Beispiel:

```
"Connection": {
  "ContactRef": {
    ...
  },
  "KeyPassword" : "123456"
},
"Orders": [
  {
    "Type":"EbicsOrderHKDRequest"
  }
]
```

EbicsOrderHxDResponse

Feldname	**Typ**		**Beschreibung**
TechnicalReturnCode	EbicsReturnCodeRS	M	Technischer EBICS Return-Code aus dem EBICS Response-Header.
BusinessReturnCode	EbicsReturnCodeRS	M	Bankfachlicher EBICS Return-Code aus dem EBICS Response-Body.
AddressInfo	EbicsAddressInfoRS	C	Adressdaten des EBICS Partners
BankINfo	EbicsBankInfoRS	C	Bankinformationen des EBICS Partners
AccountInfos	array	C	Liste der zurückgelieferten Konten vom Typ EbicsAccountInfoRS
OrderInfos	array	C	Liste der zurückgelieferten Auftragsinformationen vom Typ EbicsAuthOrderInfoRS
UserInfos	array	C	Benutzerdaten aller zurückgelieferten User vom Typ EbicsUserInfoRS

OrderTypes	array	C	Liste der 3stelligen EBICS Auftragsarten (string)

Protokollabruf - EbicsOrderHACRequest

Abruf des Kundenprotokolls im XML Format.

Feldname	**Typ**		**Beschreibung**
DateTimeFrom	string	O	Von Datum yyyy-mm-dd
DateTimeTo	string	O	Bis Datum yyyy-mm-dd

Beispiel:

```
  "Connection": {
    "ContactRef": {
      ...
    },
    "KeyPassword" : "123456"
  },
  "Orders": [
    {
      "Type":"EbicsOrderHACRequest"
    }
  ]
```

Protokollabruf - EbicsOrderProtocollResponse

Feldname	**Typ**		**Beschreibung**
TechnicalReturnCode	EbicsReturnCodeRS	M	Technischer EBICS Return-Code aus dem EBICS Response-Header.
BusinessReturnCode	EbicsReturnCodeRS	M	Bankfachlicher EBICS Return-Code aus dem EBICS Response-Body.
ProtocolEntries	array EbicsProtocolEntryRS	C	Die aus dem Originalprotokoll geparsten einzelnen Einträge

			werden hier als ein Array mit Objekten des Typs EbicsProtocolEntryRS eingestellt.

PDF Abruf Kontoinformationen - EbicsOrderBKIRequest

Abruf der elektronischen Kontozusatzinformationen im PDF Format.

Feldname	**Typ**		**Beschreibung**
DateTimeFrom	string	O	Von Datum yyyy-mm-dd
DateTimeTo	string	O	Bis Datum yyyy-mm-dd

Beispiel:

```
  "Connection": {
    "ContactRef": {
      ...
    },
    "KeyPassword" : "123456"
  },
  "Orders": [
    {
      "Type":"EbicsOrderBKIRequest"
    }
  ]
```

PDF Abruf Kontoinformationen - EbicsOrderDownloadZipResponse

Feldname	**Typ**		**Beschreibung**
TechnicalReturnCode	EbicsReturnCodeRS	M	Technischer EBICS Return-Code aus dem EBICS Response-Header.
BusinessReturnCode	EbicsReturnCodeRS	M	Bankfachlicher EBICS Return-Code aus dem EBICS Response-Body.
ZipEntries	array	C	Liste der im ZIP File enthaltenen Dokumente vom Type EbicsOrderZipEntryRS

PDF Auszüge abrufen - EbicsOrderBKARequest

Abruf der elektronischen Kontoauszüge im PDF Format.

Feldname	Typ		Beschreibung
DateTimeFrom	string	O	Von Datum yyyy-mm-dd
DateTimeTo	string	O	Bis Datum yyyy-mm-dd

Beispiel:

```
"Connection": {
  "ContactRef": {
    ...
  },
  "KeyPassword" : "123456"
},
"Orders": [
  {
    "Type":"EbicsOrderBKARequest"
  }
]
```

PDF Auszüge abrufen - EbicsOrderDownloadZipResponse

Feldname	Typ		Beschreibung
TechnicalReturnCode	EbicsReturnCodeRS	M	Technischer EBICS Return-Code aus dem EBICS Response-Header.
BusinessReturnCode	EbicsReturnCodeRS	M	Bankfachlicher EBICS Return-Code aus dem EBICS Response-Body.
ZipEntries	array	C	Liste der im ZIP File enthaltenen Dokumente vom Type EbicsOrderZipEntryRS

SEPA Zahlungsaufträge - EbicsOrderSepaRequest

Ermöglicht die Übertragung/Ausführung von SEPA Zahlungsaufträgen (Überweisungen/Lastschriften). Die SEPA Dokumente können entweder in der JSON Notation oder als Base64 codiertes XML Dokument angegeben werden.

Feldname	Typ		Beschreibung
SepaDocument	string	C	Sepa Document in JSON Notation
SepaDocumentSerialized	string	C	Base64 encoded XML

Beispiel Order JSON

```
{
	"OrderId": "xxx",
	"Type": "EbicsOrderSepaRequest",
	"SepaDocument": { ... }
}
```

Beispiel Order XML/Base64 encoded

```
{
	"OrderId": "xxx",
	"Type": "EbicsOrderSepaRequest",
	"SepaDocumentSerialized": "<base64enc>"
}
```

SEPA Zahlungsaufträge - EbicsOrderGenericResponse

Feldname	Typ		Beschreibung
TechnicalReturnCode	EbicsReturnCodeRS	M	Technischer EBICS Return-Code aus dem EBICS Response-Header.
BusinessReturnCode	EbicsReturnCodeRS	M	Bankfachlicher EBICS Return-Code aus dem EBICS Response-Body.
Success	boolean	O	Zeigt an ob der Auftrag erfolgreich übermittelt wurde

OrderSignature	array	O	Liste der Benutzersignaturen vom Typ EbicsOrderSignatureRS

EBICS VEU - Verteilte elektronische Unterschrift

EBICS VEU (Verteilte Elektronische Unterschrift) ermöglicht es Banken und Unternehmen, Zahlungsaufträge und andere Geschäftsvorfälle sicher zu unterzeichnen und freizugeben.

Durch die Möglichkeit, mehrere Unterschriften an verschiedenen Standorten zu leisten, wird der Freigabeprozess flexibler und sicherer gestaltet. So können Zahlungen nicht nur intern koordiniert, sondern auch extern durch mehrere Parteien bestätigt werden, bevor sie zur Ausführung freigegeben werden.

Der BankAccessServer unterstützt folgende VEU-Geschäftsvorfälle:

Auftragsart	BAS-Request	Beschreibung
HVE	EbicsOrderHVERequest	VEU Unterschrift hinzufügen
HVS	EbicsOrderHVSRequest	VEU Storno
HVU	EbicsOrderHVURequest	VEU Übersicht abholen
HVZ	EbicsOrderHVZRequest	VEU Übersicht von Zusatzinformationen
HVD	EbicsOrderHVDRequest	VEU Status abrufen
HVT	EbicsOrderHVTRequest	VEU Transaktionsdetails abrufen

Generische EBICS Aufträge

Über die generischen Aufträge können prinzipiell alle bankfachlichen EBICS Upload und Download Auftragsarten abgebildet werden.

EbicsOrderGenericRequest - Download

Generischer EBICS Download-Auftrag. Über den generischen EbicsOrderDownload können prinzipiell alle bankfachlichen EBICS Download-Auftragsarten abgebildet werden. Zur Ausführung dieses Auftrags wird die Download Berechtigung benötigt.

Feldname	**Typ**		**Beschreibung**

OrderType	string	M	Angabe der 3-stelligen alphanumerischen EBICS Auftragsart. Beispiel: "STA" zum Abruf von Umsätzen im MT-940-Format oder "C53" zum Abruf von Umsätzen im CAMT-Format.
OrderAttribute	string	O	Optional spezielle EBICS Auftragsattribute. Die EBICS Auftragsattribute werden normalerweise vom BAS selbst gewählt, nur in ganz speziellen Sonderfällen kann es erforderlich sein, dass spezielle Auftragsattribute übergeben werden müssen. Ab EBICS 3.0 entfallen die Auftragsattribute komplett.
OrderParams	<abhängig vom Auftragstyp>	O	Angabe der EBICS Auftragsparameter - Die zulässigen Arten von Auftragsparametern werden im weiteren Verlauf beschrieben.

Beispiel eines Umsatzabrufs im MT940 Format:

```
"Orders":[
{
    "Type":"EbicsOrderGenericRequest",
    "OrderType": "STA",
    "Upload": false,
     "OrderAttribute": "DZHNN",
    "OrderParams":{
            "Type":"EbicsStandardOrderParamsRS",
            "DateTimeFrom": "2017-10-01",
            "DateTimeTo": "2017-10-10"
     }
  }
]
```

EbicsOrderGenericResponse - Download

Feldname	Typ		Beschreibung
TechnicalReturnCode	EbicsReturnCodeRS	M	Technischer EBICS Return-Code aus dem EBICS Response-Header.
BusinessReturnCode	EbicsReturnCodeRS	M	Bankfachlicher EBICS Return-Code aus dem EBICS Response-Body.
OrderData	string	C	Bei Datenabruf, z.B. beim Abrufen von Kontoumsätzen, werden die abgerufenen Daten als Base64-codierte Binärdaten hier eingestellt. Das Format der Daten hängt von der Auftragsart ab.

Es ist zu beachten, dass beim Abruf von CAMT-Daten vom EBICS Zielsystem eine ZIP-Datei geliefert wird. Diese wird vollständig im Element *OrderData* eingestellt.

EbicsOrderGenericRequest - Upload

Generischer EBICS Upload-Auftrag. Über den generischen EbicsOrderUpload können prinzipiell alle bankfachlichen EBICS Upload-Auftragsarten abgebildet werden. Zur Ausführung dieses Auftrags wird die Upload Berechtigung benötigt.

Feldname	**Typ**		**Beschreibung**
OrderType	string	M	Angabe der 3-stelligen alphanumerischen EBICS Auftragsart. Beispiel: "STA" zum Abruf von Umsätzen im MT-940-Format oder "C53" zum Abruf von Umsätzen im CAMT-Format.
OrderData	string	M	Die einzureichenden Zahlungsdateien werden als Base-64-codierte Binärdaten eingestellt. Das Format der Daten hängt von der Auftragsart ab.
OrderAttribute	string	O	Optional spezielle EBICS Auftragsattribute. Die EBICS Auftragsattribute werden normalerweise vom BAS selbst gewählt, nur in ganz speziellen Sonderfällen kann es erforderlich sein, dass spezielle Auftragsattribute übergeben werden müssen. Ab EBICS 3.0 entfallen die Auftragsattribute komplett.
OrderParams	<abhängig vom Auftragstyp>	O	Angabe der EBICS Auftragsparameter - Die zulässigen Arten von Auftragsparametern werden im weiteren Verlauf beschrieben.

EbicsOrderGenericResponse - Upload

Feldname	Typ		Beschreibung
TechnicalReturnCode	EbicsReturnCodeRS	M	Technischer EBICS Return-Code aus dem EBICS Response-Header.
BusinessReturnCode	EbicsReturnCodeRS	M	Bankfachlicher EBICS Return-Code aus dem EBICS Response-Body.

Sollen SRZ-Aufträge im Container-Format übermittelt werden, so müssen diese schon vorher in das Container-Format gebracht werden und der daraus resultierende, vollständige Datenblock im Element OrderData eingestellt werden.

EBICS Auftragsparameter

Die Auftragsparameter (OrderParams) innerhalb einer EBICS Order müssen einen zulässigen Typ / Parameterwerte beinhalten.

Type	Parameter
EbicsStandardOrderParamsRS	**DateTimeFrom** string Anfangsdatum **DateTimeTo** string Enddatum
EbicsHVListOrderParamsRS	**OrderParamsTag** string XML tag used to serialize this order parameters instance
EbicsHVPickOrderParamsRS	**OrderID** string Auftragsnummer **OrderParamsTag**

	string XML tag used to serialize this order parameters instance **OrderType** string Auftragsart **PartnerID** string PartnerID
EbicsHVTOrderParamsRS	**CompleteOrderData** boolean Sollen die gesamten Auftragsdaten abgeholt werden? **FetchLimit** integer Anzahl abzurufender Daten **FetchOffset** integer The first order entry within the original batch order that is requested.

EBICS Order Spooler Requests

EBICS Order Requests werden immer synchron ausgeführt, d.h. ein Client wartet immer solange bis der Server eine Antwort liefert. Bei der Ausführung sehr großer Zahlungsaufträge oder dem Abruf sehr vieler Umsätze kann die Ausführung durchaus lange Zeit in Anspruch nehmen.

Für die Ausführung umfangreicher Aufträge bietet der BankAccessServer auch eine asynchrone Variante an, bei der die eigentlichen Aufträge im Hintergrund über den Ebics Spooler ausgeführt werden. Der Ablauf sieht in diesem Fall wie folgt aus:

Umsatzabruf:

- Der Client führt einen `EbicsOrderDownloadCamt053SpoolerRequest` / `EbicsOrderDownloadCamt052SpoolerRequest` aus und erhält in der Antwort eine Ebics Spooler Id.
- Im Hintergrund ruft der Ebics Spooler die Umsatzdaten ab und stellt die CAMT Dateien bereit.
- Der Client kann mit dem `EbicsOrderStatusSpoolerRequest` für die Ebics Spooler Id den Auftragsstatus abrufen.
- Für einen ausgeführten Auftrag kann im Anschluss mit dem `EbicsOrderGetFileNamesSpoolerRequest` eine Liste der abgerufenen CAMT Dateien abgerufen werden.
- Abruf der Inhalte wahlweise im XML oder JSON Format über den `EbicsOrderGetFilesSpoolerRequest`.

Senden von Aufträgen:

- Übertragen der Zahlungsdatei zum BankAccessServer via `EbicsOrderSubmitSpoolerRequest`.
- Der `EbicsOrderUploadSpoolerRequest` startet die Übertragung zum Bankserver und liefert eine Ebics Spooler Id zurück.
- Der Client kann mit dem `EbicsOrderStatusSpoolerRequest` für die Ebics Spooler Id den Auftragsstatus abrufen.
- Für einen ausgeführten Auftrag kann im Anschluss mit dem `EbicsOrderGetFileNamesSpoolerRequest` eine Liste der verarbeiteten Dateien abgerufen werden.
- Bei Bedarf Abruf der Inhalte wahlweise im XML oder JSON Format über den `EbicsOrderGetFilesSpoolerRequest`.

Bei der Verwendung des Ebics Spoolers werden die empfangenen / zu sendenden Dateien in der im BankAccessServer konfigurierten Verzeichnisstruktur gespeichert. Folgende Unterverzeichnisse werden bei Bedarf angelegt und speichern die jeweiligen Dateien:

- camt053 - empfangene CAMT53 Dateien
- camt052 - empfangene CAMT52 Dateien
- outbox - zu sendende Zahlungsaufträge
- sent - gesendete Zahlungsaufträge
- failed - fehlgeschlagene Aufträge

EbicsOrderDownloadCamt052SpoolerRequest

Abruf der Vormerkposten im Camt 052 Format. Über den Parameter OrderType kann die zu verwendende Auftragsart (C52 / VMK) vorgegeben werden.

Feldname	Typ		Beschreibung
OrderType	string	O	Angabe der 3-stelligen alphanumerischen EBICS Auftragsart C52 / VMK. Standardwert: C52
Sync	boolean	O	Im synchronen Modus wird keine `EbicsOrderTaskSpoolerResponse` zurückgeliefert, sondern direkt die `EbicsOrderDownloadResultSpoolerResponse`. Standardwert: false

Beispiel: Abruf der Vormerkposten mit der Auftragsart VMK:

```
"Orders": [
        {
            "DateTimeFirst": "{{StartDate}}",
            "OrderType": "VMK",
            "Type": "EbicsOrderDownloadCamt052SpoolerRequest",
            "Sync" : false
        }
    ]
```

Die Antwort enthält dann die Ebics Spooler Id für die folgende Statusabfrage:

```
"Responses": [
        {
            "Id": "515234481",
            "Type": "EbicsOrderTaskSpoolerResponse"
        }
    ]
```

EbicsOrderDownloadCamt053SpoolerRequest

Umsatzabruf im Camt 053 Format. Über den Parameter OrderType kann die zu verwendende Auftragsart (C53 / STA) vorgegeben werden.

Feldname	**Typ**		**Beschreibung**
OrderType	string	O	Angabe der 3-stelligen alphanumerischen EBICS Auftragsart C53 / STA. Standardwert: C53
Sync	boolean	O	Im synchronen Modus wird keine `EbicsOrderTaskSpoolerResponse` zurückgeliefert, sondern direkt die `EbicsOrderDownloadResultSpoolerResponse`. Standardwert: false
DateTimeFirst	string	O	Legt bei einem erstmaligen Download fest, ab welchem Datum der Abruf beginnen soll.

Beispiel: Abruf der Vormerkposten mit der Auftragsart VMK:

```
 "Orders": [
        {
            "DateTimeFirst": "{{StartDate}}",
            "OrderType": "STA",
            "Type": "EbicsOrderDownloadCamt053SpoolerRequest",
            "Sync": true
        }
    ]
```

Die Antwort enthält dann die Ebics Spooler Id für die folgende Statusabfrage:

```
"Responses": [
        {
            "Id": "515234481",
            "Type": "EbicsOrderTaskSpoolerResponse"
        }
    ]
```

EbicsOrderSubmitSpoolerRequest

Upload der Zahlungsdateien in die Outbox des BankAccessServers.

Feldname	Typ		Beschreibung
Files	array	O	Array der EbicsFiles, die neben einer FileId den Auftragstyp und den Base64 encodeten Payload enthalten.
SepaDocuments	array	O	Array an vollständigen Sepa Dokumenten in der Json Notation

Beispiel: Upload einer Einzelüberweisung mit Base64 codierter Payload.

```
"Orders": [
        {
            "Files": [
                {
                    "File": "{{EbicsBase64EncodedPayload}}",
                    "OrderType": "CCT",
                    "FileId": "BAS_{{$guid}}"
                }
            ],
            "Type": "EbicsOrderSubmitSpoolerRequest"
        }
    ]
```

EbicsOrderUploadSpoolerRequest

Übermittelt alle Dateien in der Outbox zum Ebics Bankserver. Dabei kann die Übertragung synchron oder asynchron erfolgen. Bei der synchronen Übertragung wird in der Antwort eine `EbicsOrderUploadResultSpoolerResponse` zurückgeliefert, im asynchronen Fall eine `EbicsOrderTaskSpoolerResponse`.

Feldname	Typ		Beschreibung
Sync	boolean	O	Im synchronen Modus wird keine `EbicsOrderTaskSpoolerResponse` zurückgeliefert, sondern direkt die `EbicsOrderUploadResultSpoolerResponse`. Standardwert: false

Beispiel:

```
"Orders": [
        {
            "Type": "EbicsOrderUploadSpoolerRequest",
            "Sync": true
        }
    ]
```

EbicsOrderStatusSpoolerRequest

Abruf des Auftragsstatus auf Basis einer EbicsSpoolerId, die als Antwort auf eine Ebics Spooler Order zurückgeliefert wurde.

Feldname	Typ		Beschreibung
Id	string	m	Id einer zuvor ausgeführten Ebics Spooler Order

Beispiel:

```
"Orders": [
        {
            "Id":"{{EbicsSpoolerId}}",
```

```
            "Type": "EbicsOrderStatusSpoolerRequest"
        }
    ]
```

In der Antwort wird in Abhängigkeit zur ursprünglichen Auftragsart eine `EbicsOrderDownloadResultSpoolerResponse` oder `EbicsOrderUploadResultSpoolerResponse` zurückgeliefert.

Beispielantwort:

```
{
    "Responses": [
        {
            "FileTransferResult": {
                "Timestamp": "2021-04-20 13:05:20Z",
                "ErrorClass": "Note",
                "TechnicalReturnCode": {
                    ...
                },
                "BusinessReturnCode": {
                    ...
                },
                "SuccessOrNoData": true,
                "TransactionID": "BEC5FABA273FF939021C24BC9DD68F96"
            },
            "Type": "EbicsOrderDownloadResultSpoolerResponse"
        }
    ]
}
```

EbicsOrderGetFileNamesSpoolerRequest

Ruft eine Liste der vorhandenen Dateien für ein bestimmtes Verzeichnis ab. Zulässige Verzeichnisnamen sind:

- camt053 - empfangene CAMT53 Dateien
- camt052 - empfangene CAMT52 Dateien
- outbox - zu sendende Zahlungsaufträge
- sent - gesendete Zahlungsaufträge
- failed - fehlgeschlagene Aufträge

Feldname	**Typ**		**Beschreibung**
Boxname	string	M	Spezifiziert den Verzeichnisnamen für den die Dateiliste abgerufen werden soll.
DateTimeFrom	string	O	Optionale Datumsangabe, ab der Dateien zurückgeliefert werden.
AccountInfo	EbicsAccountInfoRS	O	Einschränkung der Dateien auf Basis der definierten Kontoinformationen, siehe EbicsAccountInfoRS. Wird kein Wert angegeben, so werden die Dateien für alle vorhandenen Konten zurückgeliefert.

Beispiel: Abruf von Camt 053 Dokumenten ab einem bestimmten Zeitpunkt:

```
"Orders": [
        {
            "Type": "EbicsOrderGetFileNamesSpoolerRequest",
            "BoxName": "camt053",
            "DateTimeFrom": "{{StartDate}}"
        }
    ]
```

Die Antwort enhält nunmehr die vorhandenen Dateien in dem camt053 Verzeichnis:

```
"Responses": [
        {
            "FileNames": [
                {
                    "Date": "2021-09-28 00:00:00Z",
                    "OrderType": "C53",
```

```
                    "AccountInfo": {
                        "AcctNo": "XXXXXXXXXXX",
                        "BankCode": "70040041",
                        "Currency": "EUR"
                    },
                    "SequenceID": "0000000137",
                    "FileName":
"2021-09-28_C53_70040041.XXXXXXXXXXX_EUR_0000000137.xml",
                    "Suffix": ".xml",
                    "IsHidden": false,
                    "IsEncrypted": false,
                    "ByteSize": 3038
                }
            ],
            "Type": "EbicsOrderGetFileNamesSpoolerResponse"
        }
```


EbicsOrderGetFilesSpoolerRequest

Führt den Abruf der eigentlichen Dateiinhalte durch. Dabei können die Daten wahlweise als XML / Base64 encoded oder im Json Format abgerufen werden.

Feldname	Typ		Beschreibung
Boxname	string	M	Spezifiziert den Verzeichnisnamen aus dem die Datei abgerufen werden soll.
FileNames	array	M	Angabe der Dateinamen, deren Inhalte abgerufen werden sollen.
Json	boolean	O	Legt fest ob die Datei als Base64 kodiertes XML oder im Json Format abgerufen wird.

Beispiel: Abruf einer Camt 053 Datei im Json Format:


```
"Orders": [
        {
            "Type": "EbicsOrderGetFilesSpoolerRequest",
            "BoxName": "camt053",
```

```
            "FileNames": [
                "2021-04-19_C53_DE92201304000OXXXXXXXX_EUR_2101000001.xml"
            ],
            "Json" : true
        }
    ]
```

EbicsFile

Enthält alle Informationen einer zu übertragenden Ebics Datei.

Feldname	Typ		Beschreibung
File	string	M	Base64 codierte XML Datei, die die eigentlichen Auftragsdaten beinhaltet.
OrderType	string	M	Ebics Auftragsart, z.B. CCT
FileId	string	O	Optionale File Id, die zur Generierung eines eindeutigen Dateinamens verwendet werden soll.

EbicsAccountInfoRS

Repräsentiert Kontoinformationen, Auszug:

Feldname	Typ		Beschreibung
Currency	string	O	Kontowährung
BIC	string	O	BIC des Kontos
IBAN	string	O	IBAN des Kontos
AcctNo	string	O	Nationale Kontonummer
BankCode	string	O	Nationale Bankleitzahl

XS2A Orders

Der XS2A Endpoint unterstützt folgende fachliche Orders:

- Abruf von Informationen zum XS2A Contact und, falls verfügbar, den zugeordneten Konten/Kreditkarten
- Abrufen der aktuellen Kontosalden und Kreditrahmen
- Abrufen der Konto-/Kreditkartenumsätze

Wichtig: Über den Info-Request GetXs2aContactInfoRequest können allgemeine Informationen zum angegebenen XS2A Zugang abgerufen werden, ohne dass hierfür eine Anmeldung erforderlich ist.

Spezifische Informationen zum XS2A Zugang abrufen

Spezifischer Abruf der Contact Infos, Konten- und Kreditkarten (falls vorhanden) und Anzahl der Tage, bis zu dem Konto-/Kreditkartenumsätze abgerufen werden können (falls vorhanden). Der passende XS2A Zugriff wird anhand des spezifizierten Account Objekts ermittelt.

Beispiel Request:

```
{
  "Connection": {
    "Service": {
      "Account" : {
        "AcctIBAN": "DE38100110012629 58XXXX",
        "AcctCcy": "EUR",
        "AcctTpCd": "CACC"
      }
    },
    "Credentials": {
      "UserID": "xxxx",
      "Password": "xxxx"
    }
  },
  "Orders": [
    {
      "Type": "Xs2aContactInfoRequest"
    }
  ]}
```

Beispiel Response:


```
{
      "Contact": {
            "AccountServiceTypes": "Bank",
            "Authentication": "UserPassword",
            "BankCode": "10011001",
            "Capabilities": [
              "ImplementsBalance",
              "ImplementsDocuments",
              "ImplementsStatement"
            ],
            "LoginURL": "https://api.tech26.de/",
            "ProductName": "N26",
            "Profile": {
                  "UserIDLabel": "E-Mail",
                  "PasswordLabel": "Passwort"
            },
            "Xs2aName": "N26 API"
      },
      "MaxStatementDays": 90,
      "Accounts": [
            {
                  "AcctCtry": "DE",
                  "AcctIBAN": "IBAN",
                  "AcctNo": "Kontonummer",
                  "AcctBIC": "NTSBDEB1XXX",
                  "AcctBankCode": "10011001",
                  "AcctCcy": "EUR",
                  "AcctNm": "N26 Girokonto",
                  "AcctTpCd": "CACC",
                  "OwnrNm": "Klaus Igel"
            }
      ],
      "Type": "Xs2aContactInfoResponse"
}
```

Anmeldungen mit starker Kundenauthentifizierung

Die typische Anmeldung mit UserID / Password für den Xs2aContactInfoRequest sieht wie folgt aus:

```
"Connection": {
        "Service": {
            "Account": {{Xs2aAcctCRDC}}
        },
        "Credentials": {
            "UserID": "{{Xs2aUserIDCRDC}}",
            "Password": "{{Xs2aPasswordCRDC}}"
        }
    }
```

Sofern ein Anbieter keine SCA erfordert beinhaltet die Antwort bereits die Xs2aContactInfoResponse.

Ist eine SCA erforderlich, so gibt es zwei Antworttypen, die beide eine Session-Referenz beinhalten:

1. NeedQueryResponse - es ist eine weitere Auswahl für die Art der SCA, z.B. des TAN Mediums, erforderlich
2. NeedChallengeResponse - Informationen zur geforderten SCA/TAN Challenge

Beispielablauf: NeedQueryResponse mit Auswahl des TAN2go Endgeräts

```
"NeedQueryResponse": {
        "QueryFieldName": "TanMediaIndex",
        "QueryHeading": "TAN2go Endgerät auswählen",
        "QueryPrompt": "Bitte senden Sie die TAN2go an folgendes Endgerät:",
        "QueryChoices": [
            {
                "Value": "0",
                "Text": "GooglePixel",
                "Default": false
            },
            {
                "Value": "1",
                "Text": "Iphone",
                "Default": false
            }
        ]
```

```
    },
    "Session": "63d0b022-8a60-4106-9d32-4394a7bc52ee"
```

In diesem Fall muss im folgenden Request der ausgewählte Wert, der Feldname und die zurückgelieferte Session angegeben werden

```
"Connection": {
        "SetQueryResponse": {
            "Session": "{{xs2aSession}}"
        },
        "Credentials": {
            "UserID": "{{Xs2aUserIDCRDC}}",
            "Password": "{{Xs2aPasswordCRDC}}",
            "QueryResponseFieldName": "TanMediaIndex",
            "QueryResponseFieldValue":"1"
        }
    }
```

Die Antwort enthält nunmehr den ChallengeType, sofern verfügbar die ChallengeData (z.B. bei ChipTAN), den DisplayText mit Hinweisen für den Benutzer sowie die Session Referenz.

```
"NeedChallengeResponse": {
       "ChallengeType": "SimpleTAN",
       "DisplayText": "Bitte bestätigen Sie die Anmeldung mit der TAN aus Ihrer
DKB-TAN2go-App."
   }
```

Im letzten Schritt erfolgt die Übertragung der ChallengeResponse, auf die dann mit der Xs2aContactInfo Response geantwortet wird.

```
"Connection": {
        "SetChallengeResponse": {
            "Session": "{{xs2aSession}}"
        },
        "Credentials": {
             "ChallengeResponse": "187876"
        },
        "Suspend": false
    }
```

Kontosalden abfragen

Abfrage der bereitgestellten Salden für die angegebene Kontoverbindung / Kreditkarte.

Beispiel Request:

```
{
  "Connection": {
    "Service": {
      "Xs2aId": "N26",
    },
    "Credentials": {
      "UserID": "xxxx",
      "Password": "xxxx"
    }
  },
  "Orders": [
    {
      "Type": "Xs2aDownloadBalancesRequest",
      "Account" : {
           "AcctIBAN": "DE38100110012629 58XXXX",
           "AcctCcy": "EUR",
           "AcctTpCd": "CACC"
      }
```

```
      }
   ]
}
```

Beispiel Response:


```
Balances": [
     {
      "Amount": {
         "Amount": 441.24,
         "CreditDebitIndicator": "Credit",
         "CreditDebitIndicatorCode": "CRDT",
         "Currency": "EUR",
         "IsDebit": false,
         "Value": 441.24
      },
      "CreditLineIncluded": false,
      "Date": "2017-12-07 00:00:00Z",
      "Type": "ClosingBooked",
      "TypeCode": "CLBD"
   }
],
"Type": "Xs2aDownloadBalancesResponse"
```


Grundsätzlich kann die Antwort folgende Saldentypen enthalten:

CLBD	Closing booked balance.
CLAV	Closing available balance.
FWAV	Forward available balance.
ITBD	Interim booked balance.
PRCD	Previously closed booked balance.

Umsatzabfrage

Ruft die Umsätze ab dem angegebenen Datum und für die spezifizierte Kontoverbindung / Kreditkarte ab. Die Umsätze werden im ISO 20022 CAMT Format bereitgestellt.

Beispiel Request - Abfrage der Umsätze ab 01.12.2017

```
{
  "Connection": {
    "Service": {
      "Account" : {
        "AcctIBAN": "DE38100110012629 58XXXX",
        "AcctCcy": "EUR",
        "AcctTpCd": "CACC"
      }
    },
    "Credentials": {
      "UserID": "user",
      "Password": "password"
    },
    "Suspend": false
  },
  "Orders": [
    {
      "Type": "Xs2aDownloadStatementRequest",
      "Account" : {
          "AcctIBAN": "IBAN",
          "AcctCcy": "EUR",
          "AcctTpCd": "CACC"
      },
      "FromDateTime" : "2017-12-01"
    }
  ]
}
```

Die Antwort enthält das gültige Sepa Statement Document.

Dokumentenliste abrufen

Abruf der beim Service Provider verfügbaren Dokumente, z.B. Kontoauszüge oder Kredikartenabrechnungen im PDF Format.

Beispiel Request:

```
{
  "Connection": {
    "Service": {
      "Xs2aId": "N26",
    },
    "Credentials": {
      "UserID": "xxxx",
      "Password": "xxxx"
    }
  },
  "Orders": [
    {
      "Type": "Xs2aDownloadDocumentListRequest",
      "Account" : {
           "AcctIBAN": "DE38100110012629 58XXXX",
           "AcctCcy": "EUR",
           "AcctTpCd": "CACC"
       }
    }
  ]
}
```

Beispiel Response:


```
"DocumentsProperties": [
    {
     "CreDt": "2018-04-01",
     "ElctrncSeqNb": 4,
     "ElctrncStmtId": "statement-2018-04",
     "ElctrncStmtYr": 2018,
     "FileTtl": "Kontoauszug",
     "FileTp": "SPDF"
  },
    {
     "CreDt": "2018-03-01",
     "ElctrncSeqNb": 3,
     "ElctrncStmtId": "statement-2018-03",
     "ElctrncStmtYr": 2018,
     "FileTtl": "Kontoauszug",
     "FileTp": "SPDF"
  },
    {
     "CreDt": "2018-02-01",
     "ElctrncSeqNb": 2,
     "ElctrncStmtId": "statement-2018-02",
     "ElctrncStmtYr": 2018,
     "FileTtl": "Kontoauszug",
     "FileTp": "SPDF"
  }
],
"Type": "Xs2aDownloadDocumentListResponse”
```


Der Xs2aDownloadDocumentRequest ermöglicht den Download der jeweiligen Kontoauszüge / Kreditkartenabrechnungen unter Angabe der entsprechenden DocumentProperties

Dokument abrufen

Abruf eines einzelnen Dokumentes, z.B. Kontoauszug oder Kredikartenabrechnung im PDF Format.

Beispiel Request:

```
{
  "Connection": {
    "Service": {
      "Xs2aId": "N26",
    },
    "Credentials": {
      "UserID": "xxxx",
      "Password": "xxxx"
    }
  },
  "Orders": [
    {
      "Type": "Xs2aDownloadDocumentRequest",
      "Account" : {
           "AcctIBAN": "DE38100110012629 58XXXX",
           "AcctCcy": "EUR",
           "AcctTpCd": "CACC"
       },
       "DocumentProperties": {
           "CreDt": "2018-04-01",
           "ElctrncSeqNb": 4,
           "ElctrncStmtId": "statement-2018-04",
           "ElctrncStmtYr": 2018,
           "FileTtl": "Kontoauszug",
           "FileTp": "SPDF"
       }
    }
  ]
}
```

Beispiel Response:

```
"Document": "<Base64 encoded doc>"
"Type": "Xs2aDownloadDocumentResponse"
```

Das eigentliche Dokument wird dabei Base64 encoded zurückgeliefert und muss für die weitere Verwendung noch decodiert werden.

BerlinGroupOrders

Der im BankAccessServer verfügbare BerlinGroup Endpoint unterstützt aktuell die folgenden fachlichen Basisfunktionalitäten:

- Ermittlung der Bank-/Profilinformationen auf Basis einer BIC / IBAN

- Bankspezifische Pre Authentication
  - Zwischen TPP und Bank
  - Zwischen Kunde und Bank

- Payment Initiation Service (PIS)
  - Initiieren von Zahlungen
  - Löschen von Zahlungen
  - Abrufen Zahlungsinformationen
  - Abrufen des Status einer Zahlung
  - Abrufen des SCA Status einer Zahlungsautorisierung

- Account Information Service (AIS)
  - Erteilen eines Consents
  - Löschen eines Consents
  - Abfrage der Consent-Informationen
  - Abfrage des Consent Status
  - Abruf der Zahlungsverkehrskonten
  - Lesen von Kontosalden
  - Umsatzabruf

- Payment Instrument Issuing Service (PIIS)
  - Abfrage ob ein angefragter Betrag verfügbar ist

Profildaten ermitteln

Für die von uns im Rahmen des BerlinGroup Endpoints unterstützten Banken können die jeweiligen Profildaten anhand der übergebenen BIC oder einer deutschen IBAN abgefragt werden. Die Profildaten sind dabei die Basis für den Aufbau institutsspezifischer Requests und beinhalten z.B. URLs, unterstützte SCA Verfahren und Payment Produkte, Consent-Arten, Datenformate und weitere spezifische Informationen.

Wichtiger Hinweis: Im Zuge Weiterentwicklung der BerlinGroup Spezifikation, Anpassungen bei den produktiven PSD2 Serverumgebungen sowie durch die Unterstützung neuer Geschäftsvorfälle haben wir mit dem BAS Release 3.3 neue Profilinformationen eingeführt.

Da die bekannten CSV-basierten Bank- und Profildaten bereits bei zahlreichen Anwendern im Einsatz sind, haben wir uns entschlossen, die neuen Daten parallel bereitzustellen. Neuen Anwendern wird empfohlen, in der Entwicklung immer auf die umfangreichen neuen Profil- und Bankdaten im JSON Format zurückzugreifen.

Profildaten anhand der BIC ermitteln

Ermittelt die Bank-/Profildaten anhand der übergebenen BIC.

Abfrage der Profildaten im neuen Format:

```
{
		"OrderId": "{{$randomInt}}",
		"Type": "BGGetNewProfileByBICRequest",
		"BIC": "EASYATW1XXX"
}
```

Abfrage der Profildaten im alten Format *-deprecated-*:

```
{
		"OrderId": "{{$randomInt}}",
		"Type": "BGGetProfileByBICRequest",
		"BIC": "EBSGDEMXXXX",
		"Sandbox": false
}
```

Die Ausgabe beinhaltet in der Regel ein Profil, kann aber in besonderen Konstellationen (z.B. Fraspa/1822, gleiche BIC, unterschiedliche Profile) auch mehrere Einträge enthalten. In diesem Fall muss vom Aufrufer das passende Profil selektiert werden.

Profildaten anhand der IBAN ermitteln

Neben der BIC basierten Ermittlung kann für die Ermittlung der Profildaten auch eine deutsche IBAN verwendet werden.

Abfrage der Profildaten im neuen Format:

```
{
            "OrderId": "{{$randomInt}}",
            "Type": "BGGetNewProfileByGermanIBANRequest",
            "IBAN": "{{BGIban}}"
}
```

Abfrage der Profildaten im alten Format *-deprecated-*:

```
{
            "OrderId": "{{$randomInt}}",
            "Type": "BGGetProfileByGermanIBANRequest",
            "IBAN": "{{BGIban}}",
            "Sandbox": false
}
```

Neue Profildaten

Id - string
Eindeutige Id des Profils, z.B. sparkasse

AspspId - string
Eindeutige Kennzeichnung des ASPSPs, entspricht in der Regel der BIC

Country - string
2 stelliger Ländercode nach ISO-3166, z.B. DE

AspspHost - string
Host des PSD2/XS2A Servers des ASPSP

AspspTokenHost - string
Host des OAuth Servers des ASPSP

AspspAuthorizationHost - string
Host des Authorization Servers des ASPSP

BIC - string
BIC der angefragten Bank

BankName - string
Name der angefragten Bank

BankCode - string
Bankleitzahl der angefragten Bank

ConsentsPath - string
Pfad für Consent Requests, der an den AspspHost angehängt wird

AisPath - string
Pfad für AIS Aufrufe, der an den AspspHost angehängt wird

PisPath - string
Pfad für PIS Aufrufe, der an den AspspHost angehängt wird

PiisPath - string
Pfad für PIIS Aufrufe, der an den AspspHost angehängt wird

TokenPath - string
Pfad, der an den OAuth2TokenHost angehängt wird

AuthorizationPath - string
Pfadangabe für den Authorisation Endpoint, z.B. /auth

EncryptPasswordUrl - string
Zertifikats URL für die Passwortverschlüsselung

UseSignature - bool
Kennzeichnung, ob eine Inhaltsverschlüsselung mit dem QSEAL Zertifikat erfolgen muss

ScaFlows - Array BGProfileScaFlow
Gibt an, welche SCA Flows unterstützt werden.
Mögliche Werte: "redirectBerlinGroup", "redirectOauth", "decoupled", "embedded"

PsuIdRequired - bool
Muss bei Verwendung der Redirect SCA eine PSU ID angegeben werden

PsuIdType - string
Angabe des Wertes, der für PSU ID Type gesetzt werden muss, z.B. DE_ONLB_NORIS

SuppressPsuIdTypeForRedirect - bool
Information, ob die PSU IP Adresse bei weiteren AIS Aufrufen angegeben werden darf

TppExplicitAuthorisationRedirectPreferred - string
Gibt an, ob eine explizite Autorisierung gewünscht wird. Dieses Merkmal wird in der Praxis von einigen Banken zur Einleitung eines Decoupled Flows verwendet, z.B. Comdirect / Commerzbank

StrictUrlValidation - bool
Gibt an, ob die Redirect URL des TPPs mit der Zertifikats-URL übereinstimmen muss

UseLinkUrlForEmbeddedAuthorisation - bool
Merkmal, ob die im Rahmen der Embedded SCA zurückgelieferte URL weitere Informationen beinhaltet, die bei nachfolgenden Authorisierungsschritten verwendet werden muss

EncryptedPassword - bool
Flag, ob das PSU Passwort verschlüsselt übertragen werden muss

TppRedirectUriForEmbeddedSca - bool

Angabe, ob die Redirect URL auch bei einer Embedded SCA mit angegeben werden muss

IgnoreSteeringLinks - bool

Hinweis, ob die in den Responses zurückgelieferten Links akkurat sind und zur weiteren Steuerung des Workflows verwendet werden können

PreferredConsentAccess - BGNewProfileConsentAccess

Angabe des präferrierten Consent-Typs. Mögliche Werte sind: "Detailled", "BankOffered", "GlobalAllPsd2", "GlobalAvailable"

SupportedConsentAccess - Array BGNewProfileConsentAccess

Liste der unterstützten Consent-Typen. Mögliche Werte sind: "Detailled", "BankOffered", "GlobalAllPsd2", "GlobalAvailable"

MaxConsentDays - integer

Maximale Gültigkeit des Consents in Tagen, z.B. 90

AddDaysForOneTimeConsent - int

Anzahl der Tage, die für einen Einmal-Consent zum heutigen Datum hinzugefügt werden müssen

NeedAdditionalInfoForOwnerName - bool

Angabe, ob für die Abfrage des Kontoinhabers der AccountAccess mit dem Suffix „withOwnerName“ verwendet werden muss

OwnerNameField - BGNewProfileOwnerNameField

Angabe, in welchem Feld der Name des Kontoinhabers bereitgestellt wird. Mögliche Werte sind: "OwnerName", "Name"

ConsentAuthorisationEndpoints - bool

Flag, ob die Consent-Autorisierungs-Endpoints vorhanden sind und somit z.B. die Abfrage des SCA Status unterstützt wird

CombinedServiceIndicatorSupported - bool

Wird der Combined Service Indicator bei der Consenterstellung unterstützt?

SupportedTransactionFormat - Array BGNewProfileTransactionFormat

Liste der unterstützten Formate, in denen die Umsätze bereitgestellt werden können. Mögliche Werte sind: "Json", "Camt", "CamtZip", "Mt940"

SupportedStandingOrderFormat - Array BGNewProfileTransactionFormat

Liste der unterstützten Formate, in denen Dauerauftragsinformationen bereitgestellt werden können. Mögliche Werte sind: "Json", "Camt", "CamtZip", "Mt940"

ExplicitTransactionFormat - bool

Explizites Format, das bei der Umsatzabfrage als Accept Header angegeben werden muss, z.B. "application/json"

ExplicitStandingOrderFormat - bool

Explizites Format, das bei der Abfrage von Daueraufträgen als Accept Header angegeben werden muss, z.B. "application/json"

BookingStatus - Array BGProfileBookingStatus

Angabe der bei einer Umsatzabfrage möglichen Buchungs-Statusangaben. Möglich Werte sind: "information", "both", "booked", "pending"

AccountDetailsSupported - bool

Angabe, ob Kontodetails abgefragt werden können bzw. ob der AccountDetails Endpoint vom ASPSP angeboten wird

SupportedPayments - Array BGProfilePaymentData

Liste der angebotenen Payment-Services-/Produkte:

Mögliche Werte für Payment-Services:

"payments", "bulk-payments", "periodic-payments"

Mögliche Werte für Payment-Produkte:

"sepa-credit-transfers", instant-sepa-credit-transfers", "target-2-payments", "cross-border-credit-transfers", "pain.001-sepa-credit-transfers", "pain.001-instant-sepa-credit-transfers", "pain.001-target-2-payments", "pain.001-cross-border-credit-transfers"

SupportedPaymentInformationFormat - Array BGNewProfilePaymentFormat

Liste der unterstützten Formate in denen die Payment Informationen abgefragt werden können. Mögliche Werte sind: "Json", "Pain"

PaymentStatusFormat - BGNewProfilePaymentFormat

Format, in dem der Paymentstatus abgefragt werden kann. Mögliche Werte sind: "Json", "Pain"

ExplicitPaymentStatusFormat - bool

Explizites Format, das bei der Abfrage des Payment Status als Accept Header angegeben werden muss, z.B. "application/json"

ExplicitPaymentInformationFormat - bool

Explizites Format, das bei der Abfrage der Payment Informationen als Accept Header angegeben werden muss, z.B. "application/json"

PaymentAuthorisationEndpoints - bool

Flag, ob die Payment-Autorisierungs-Endpoints vorhanden sind und somit z.B. die Abfrage des SCA Status unterstützt wird

RelativeScaOAuth2Link - bool

Angabe, ob es sich bei den in der Response zurückgelieferten OAuth2 Links um relative oder absolute URLs handelt

ScaOAuth2LinkAuthorise - bool

Gibt an, ob es sich bei der zurückgelieferten URL um einen Authorise Link handelt (OAuth2 Redirect)

ScaOAuth2LinkMetadata - bool

Gibt an, ob es sich bei der zurückgelieferten URL um einen Link für die Ermittlung der Metadaten handelt (OAuth2 Redirect)

ScaOAuth2AuthoriseLinkFi - bool

Gibt an, ob der Authorisierungslink im Format der Finanz Informatik erzeugt werden muss (OAuth2 Redirect)

ScopeNameAis - string

Scope Name für AIS Anfragen (OAuth2 Redirect)

ScopeNamePis - string

Scope Name für PIS Anfragen (OAuth2 Redirect)

AppendScopeRoleToPath - bool

Kennzeichen, ob der Scope Name an den Pfad als URL Parameter angehängt werden muss (OAuth2 Redirect)

SupressClientIdInTokenRequest - bool

Muss die ClientId in Token Requests unterdrückt werden? (OAuth2 Redirect)

ProfileInfo - Array BGNewProfileInfo

Gibt an, ob der Kontoinhaber in der Kontoliste und den Kontodetails zurückgeliefert wird. Mögliche Werte sind: "OwnerNameInAccountList", "OwnerNameInAccountDetails"

Header - BGNewProfileCustomHeader

Gibt an welchen ergänzenden Header-Informationen für Requests benötigt werden. Mögliche Werte sind:

"NeedBearerToken" / "BearerTokenName"

"NeedClientId" / "ClientIdName"

"NeedBic" / "BicName":

PreAuthHeader - BGNewProfileCustomHeader

Gibt an welchen ergänzenden Header-Informationen im Rahmen der Pre-Authorisation bei den Requests benötigt werden. Mögliche Werte sind:

"NeedBearerToken" / "BearerTokenName"

"NeedClientId" / "ClientIdName"

"NeedBic" / "BicName":

PreAuthFlow - BGNewProfilePreAuthFlow

Angabe des zu verwendenden Pre Authorisation Flows. Mögliche Werte sind: "ClientCredentials", "ClientCredentialsWithScope", "OAuth2Authorisation", "PsuLogin", “PsuLogin Json", “Ing"

AuthorisationHeaderForStatusRequests - bool

Muss ein Authorisation Header für Status Requests angegeben werden?

ReplaceCodeChallengeInRedirectUrl - bool

Angabe, ob die Code Challenge in der zurückgelieferten Redirect URL ersetzt werden muss (OAuth2 Redirect)

RequestedExecutionDateSupported - bool

Kann bei Zahlungen ein Ausführungstermin (RequestedExecutionDate) angegeben werden

AvoidExceedingDateFrom - bool

Sofern true, reagiert der ASPSP bei Überschreiten des möglichen “Von Datums” in Umsatzabfragen mit einem serverseitigen Fehler

BasicAuthForTokenRequests - bool

Gibt an, ob für Token Requests eine Basic Authentication notwendig ist. Dies ist aktuell bei der Fidor Bank erforderlich

Psd2Variant - string

Definiert die zu verwendende API Variante. Mögliche Werte sind:

BerlinGroup, OpenBanking, ING, BankNorwegian

DisableRemoteCertificateValidation - bool

Gibt an, ob für erfolgreiche API Requests die Zertifikatsvalidierung deaktiviert werden muss. Bei Bedarf kann mit ASPSPs kommuniziert werden, die ungültige Zertifikate verwenden.

Alte Profildaten -deprecated-

Die Profildaten beinhalten folgende Informationen, die von einer Clientanwendung gezielt ausgewertet werden können.

Name	Typ	Beschreibung
Id	string	Eindeutige Id des Profils, z.B. sparkasse
AspspId	string	Eindeutige Kennzeichnung des ASPSPs, entspricht in der Regel der BIC
Country	string	2 stelliger Ländercode nach ISO-3166, z.B. DE
AspspHost	string	Host des PSD2/XS2A Servers des ASPSP
OAuth2TokenHost	string	Host des OAuth Servers des ASPSP
IDPAdress	sting	Server Adresse des Identity Providers, der im Rahmen einer Pre-Authentifizierung verwendet wird. Verwendung z.B bei der Ersten Bank und Sparkasse / Sparda Banken
AspspPisPath	string	Pfad für PIS Aufrufe, der an den AspspHost angehängt wird
AspspAisPath	string	Pfad für AIS Aufrufe, der an den AspspHost angehängt wird
AspspConsentsPath	string	Pfad für Consent Aufrufe, der an den AspspHost angehängt wird
AspspPiisPath	string	Pfad für PIIS Aufrufe, der an den AspspHost angehängt wird
OAuth2TokenPath	string	Pfad, der an den OAuth2TokenHost angehängt wird

AuthorizationHost	string	Host des Authorisierungsservers des ASPSPs
AuthorizationPath	string	Pfadangabe Authorisierungsaufrufe
PreAuthenticationRequired	bool	Ist eine vor der Verwendung der API eine Pre-Authentication notwendig
SignatureRequired	bool	Angabe, ob eine Inhaltssignatur notwendig ist
SupportedPayments	array	Unterstützte Payment Produkte/Services
SupportedConsentTypes	array	Unterstützte Arten der Consenterteilung, z.B. Detailed
ConsentWithOwnerName	bool	Angabe, ob bei der Consenterteilung explizit der Name des Kontoinhabers angefragt werden kann
ConsentWithBalance	bool	Angabe, ob bei der Consenterteilung Salden im Rahmen der Kontenabfrage zurückgeliefert werden können
SupportedScaFlows	array	Unterstützte SCA-Flows, z.B. Redirect
SupportedTransactionFormats	array	Unterstützte Formate, in denen die Transaktionsdaten zurückgeliefert werden, z.B. json
SupportedPaymentInitiationStatusFormats	array	Unterstützte Formate, in denen die Payment Initiation Daten zurückgeliefert werden, z.B. json
SupportedPaymentInformationFormats	array	Unterstützte Formate, in denen die Payment Information Daten zurückgeliefert werden, z.B. json
CustomHeaders	array	Angabe zusätzliche Http-Header für die Berlin Group Requests, z.B. API-Key, Authorisation
PsuData	array	Angabe, wie die PsuDaten übermittelt werden müssen, z.B. encryptedPassword, password
SupportedBookingStatus	array	Information welche Booking Stati unterstützt werden, z.B. booking, pending
ScaOAuth2LinkAuthorise	bool	Kann der zurückgelieferte scaOAuth2 Link direkt für die Generierung eines Kundenlinks verwendet werden
ScaOAuth2LinkMetadata	bool	Kann der zurückgelieferte scaOAuth2 Link für den Abruf von Metadaten verwendet werden
ScaOAuth2AuthoriseLinkFi	bool	Angabe ob der Authorisation Link im speziellen FI Format generiert werden soll -> `BGOAuth2GenerateAuthoriseRequest -> FiLink`
PSUIDType	string	Angabe des Wertes, der für PSU ID Type gesetzt werden muss, z.B. `DE_ONLB_NORIS`
PSUIDRedirectRequired	bool	Muss auch bei Verwendung der Redirect SCA eine PSU ID angegeben werden

SuppressPSUIPAddressForAIS	bool	Information, ob die PSU IP Adresse bei weiteren AIS Aufrufen angegeben werden darf
NeedExplicitTransactionFormat	bool	Angabe ob das Transaktionsformat beim Abruf der Umsätze explizit angegeben werden muss.
RedirectURIForEmbeddedSCA	bool	Angabe ob auch bei einer embedded SCA eine Redirect URI angegeben werden muss
PreAuthWSO2Token	bool	Benötigt der TPP ein WSO2 Token im Rahmen der Pre Authentification? Verwendung z.B. bei der DKB
PreAuthWSO2TokenURL	string	URL für den Bezug des WSO2 Access Tokens
PreAuthOAuthToken	bool	Wird ein OAuth2 Token im Rahmen der Pre Authentification benötigt?
PreAuthOAuthTokenURL	string	URL für den Bezug des OAuth2 Access Tokens. Verwendung z.B. bei der DKB
PreAuthSendAppTAN	bool	Wird eine App TAN im Rahmen der Pre Authentification benötigt?
PreAuthOAuthSendAppTAN	string	URL für die Übermittlung der App TAN. Verwendung z.B. bei der DKB
PreAuthSSOToken	bool	Benötigt der TPP ein SSO Token im Rahmen der Pre Authentification? Verwendung z.B. bei den Raiffeisenbanken Österreich
PreAuthSSOTokenURL	string	URL für den Bezug des SSO Access Tokens
AlwaysRequirePSUIDForAIS	bool	Wird die PSU ID für alle AIS Aufrufe benötigt? Beispiel: Sparda
AlwaysRequirePSUIDForPIS	bool	Wird die PSU ID für alle PIS Aufrufe benötigt? Beispiel: Sparda
JWTAccessTokenContainsPSUID	bool	Beinhaltet das zurückgelieferte JWT Access Token die PSU ID? Beispiel: Sparda
ReplaceCodeChallengeInRedirectURL	bool	Muss innerhalb einer Redirect URL die Code Challenge ersetzt werden? Beispiel: Sparda
ScopeNameAIS	string	Angabe des zu verwendenden Scope Namens, z.B. AIS, AISP, ais
ScopeNamePIS	string	Angabe des zu verwendenden Scope Namens, z.B. PIS, PISP, pis
UseClientSecretForOAuth	bool	Muss für OAuth2 Aufrufe ein spezielles Clientsecret angegeben werden? Default: Code Challenge
CertificateNamePasswordEncryption	string	Name des Zertifikats, das für die Passwortverschlüsselung verwendet wird

SCA-Varianten

Der BankAccessServer unterstützt die folgenden SCA Varianten:

- Redirect Berlin Group
- Redirect OAuth2
- Embedded
- Decoupled

In beiden Redirect Varianten erfolgt die SCA durch den bzw. auf den Seiten des ASPSPs. Der jeweilige Kunde (PSU) gibt dabei seine Zugangs- und Freigabedaten direkt auf der Bankseite ein.

Beim Redirect Berlin Group Ansatz wird bei der Consent Erstellung bzw. der Einreichung einer Zahlung ein Redirect Link zurückgegeben an den der Kunde weitergeleitet werden kann, um die SCA durchzuführen.

Beim Redirect OAuth sind folgende Profilangaben für den weiteren Flow wichtig:

- "ScaOAuth2LinkAuthorise": true | false -> gibt an, ob die zurückgelieferte URL bereits eine Authorisierungs-URL darstellt, an die der Kunde weitergeleitet werden kann.
- "ScaOAuth2LinkMetadata": true | false -> handelt sich bei der zurückgelieferten URL um Metadaten für die Erzeugung der Authorisierungs-URL.
- "ScaOAuth2AuthoriseLinkFi": true | false -> müssen die Parameter in der Authorize URL in der Schreibweise responseType/clientId (true) oder response_type/client_id (false) angegeben werden. Ferner muss bei zwischen Scope und ConsentID/PaymentID immer ein Leerzeichen angegeben werden.

PIS Beispiele

Redirect consorsbank

Einstellen der Zahlung -> Kunde wird an die redirectURL weitergeleitet -> Kunde gibt die Zahlung frei -> Rücksprung zum TPP -> Abfrage des Payment Status durch TPP

OAuth2 Commerzbank / Comdirect

Einstellen der Zahlung -> TPP generiert Authorisierungs URL anhand der scaOAuthURL -> Kunde gibt die Zahlung frei -> Rücksprung zum TPP -> Abfrage des Payment Status durch TPP

OAuth2 Sparkassen

Einstellen der Zahlung -> TPP ermittelt über die scaOAuthURL die Authorisierungs-URL -> TPP generiert Authorisierungs URL -> Kunde wird an die Authorisierungs-URL weitergeleitet - -> Kunde gibt die Zahlung frei -> Rücksprung zum TPP -> Abfrage des Payment Status durch TPP

Die weiteren im Berlin Group Standard vorgesehenen SCA Verfahren embedded und decoupled beziehen den jeweiligen TPP wesentlich stärker ein. So werden beispielsweise beim Embedded Verfahren alle Details direkt durch den TPP angestoßen, angefangen bei der Übermittlung der PSU Daten, der Auswahl des verwendeten Sicherheitsverfahren bis hin zum Versenden einer eventuell benötigten TAN.

Exemplarisch wird der Ablauf einer Zahlung unter Verwendung der Embedded Variante dargestellt.

Volksbanken

Einstellen der Zahlung -> Starten der PSU Authentication (Übermittlung der PIN) -> Auswahl des Sicherungsverfahrens -> Übermittlung der TAN -> Abfrage des Payment Status durch TPP

Sparkassen

Einstellen der Zahlung -> Starten der PSU Authentication (Übermittlung der PIN) -> Auswahl des TAN-Mediums -> Übermittlung der TAN -> Abfrage des Payment Status durch TPP

TPP Daten

Jede Order, die an einen ASPSP (Bank) geschickt wird, muss die TPP-Daten / Zertifkatsinformation beinhalten. Dabei handelt es sich um folgende Informationen:

Feldname	**Typ**	**Beschreibung**
ASPSPId	string	Eindeutige ID des ASPSPs, in der Regel entspricht diese der BIC. Eine ASPSPId kann für die serverseitige Ermittlung der TPP Profildaten verwendet werden.
ASPSPHost	string	Basis URL der PSD2 Sandbox- oder Produktionsumgebung - die effektive Verbindungs-URL setzt sich aus ASPSPHost + ASPSPPath zusammen.
ASPSPPath	string	URL Pfad der PSD2 Sandbox- oder Produktionsumgebung - die effektive Verbindungs-URL setzt sich aus ASPSPHost + ASPSPPath zusammen.
ASPSPUrl	string	Sofern angegebe, werden die angegebenen Werte für ASPSPHost/Path ignoriert. Sollte verwendet werden wenn z.B. die Url aus einer vorherigen Response verwendet wird.
TransportCertificateId	string	ID unter der das zu verwendende QWAC Zertifikat in den appsettings aufgelistet ist.
TransportCertificatePassword	string	Zertifikatspasswort, sofern erforderlich.
TransportCertificate	string	Base64 encoded QWAC Zertifikat, das alternativ zu einer Zertifikats-ID verwendet werden kann.
SignatureCertificateId	string	ID unter der das zu verwendende QSEAL Zertifikat in den appsettings aufgelistet ist.
SignatureCertificatePassword	string	Zertifikatspasswort, sofern erforderlich
SignatureCertificate	string	Base64 encoded QSEAL Zertifikat, das alternativ zu einer Zertifikats-ID verwendet werden kann.
ClientId	string	Angabe einer spezifischen Client ID, die z.B. beim Onboarding des TPPs vergeben wird.
CustomerHeaders	array	Array an Zusatzinformationen, die beim Request mitgeschickt werden. Bei OAuth Requests kann hier z.B. das Bearer Token

		mitgegeben werden (Name=”Authorization” / Value=”Bearer <access_token>”
Psd2Variant	string	Definiert die zu verwendende API Variante. Mögliche Werte sind: BerlinGroup, OpenBanking, ING, BankNorwegian Sofern kein Wert angegeben wird, ist als Default Value BerlinGroup eingestellt
DisableRemoteCertificateValidation	bool	Gibt an, ob für API Requests die Zertifikatsvalidierung deaktiviert werden soll. Bei Bedarf kann mit ASPSPs kommuniziert werden, die ungültige Zertifikate verwenden.

Somit können die TPP Zertifikatsdaten in der Praxis in 3 Varianten übermittelt werden.

Serverseitige Verwaltung der TPP Zertifikate - appsettings.json

Sofern in den Requests Zertifikats-Ids angegeben werden, müssen diese in den Appsettings des BankAccessServers konfiguriert sein.

```
BerlinGroupConfig": {
    ...
    "Certificates": {
      "qwac": "Certs/qwacCertificateWithPassword.pfx",
      "qseal": "Certs/qsealCertificateWithPassword.pfx"
    },
    "CertificatesWithoutPassword": {
      "qwacNoPW": "Certs/qwacCertificateNoPassword.pfx",
      "qsealNoPW": "Certs/qsealCertificateNoPassword.pfx"
    },...
  }
```

Zertifikat mit Passwort

Bei einem Berlin Group Request ist neben der Zertifikats-ID auch das Zertifikatspasswort anzugeben, die Zertifikatsverwaltung erfolgt serverseitig. Beispiel:

```
"TPPData": {
        "ASPSPHost": "https://api.sparda.de:443/xs2a",
```

```
        "ASPSPPath": "/v1",
        "TransportCertificateId": "qwac",
        "TransportCertificatePassword": "certificatePW",
```

Zertifikat ohne Passwort

Bei einem Berlin Group Request wird lediglich die Zertifikats-ID benötigt, die Zertifikatsverwaltung erfolgt serverseitig. Beispiel:

```
"TPPData": {
        "ASPSPHost": "https://api.sparda.de:443/xs2a",
        "ASPSPPath": "/v1",
        "TransportCertificateId": "qwacNoPW",
```

Base64 encoded Zertifikat

Bei einem Berlin Group Request ist das Base64 encoded Zertifikat anzugeben. Der BankAccessServer verwaltet in diesem Fall keine Zertifikate, sondern verwendet das angegebene Zertifikat ohne weitere Validierung/Verwaltung. Beispiel:

```
"TPPData": {
        "ASPSPHost": "https://api.sparda.de:443/xs2a",
        "ASPSPPath": "/v1",
        "TransportCertificate": "<Base64encodedCert>",
```

Verwendung der ASPSPId

Eine ASPSPId kann für die serverseitige Ermittlung der TPP Profildaten verwendet werden. Somit können im Clientrequest die Angabe von Host/URL vermieden werden. Beispiel:

```
"TPPData": {
                "ASPSPId": "{{BGASPSPId}}",
                "TransportCertificateId": "qwac",
                "SignatureCertificateId": "qseal",
                "ClientId": "{{ClientId}}"
            },
```

PreAuth Requests

Einige Banken verlangen vor der eigentlichen Verwendung eine PreAuthentication. In einigen Fällen, wie z.B. bei den Raiffeisenbanken Österreich, reicht eine einfache Authentifizierung des TPPs anhand von im Rahmen eines Onboardings generierten Credentials aus. Andere Institute, wie z.B. die DKB oder die Spardabanken, fordern zusätzlich umfangreiche und individuelle kundenseitige freigaben für den Zugriff auf die Berlin Group API. In diesem Fall lassen sich leider mehrfache SCAs für einen Kontozugriff nicht vermeiden. Im Sinne einer einheitlichen Pre-Authentifizierung empfehlen wir die Verwendung des universell einsetzbaren BGOAuth2AuthorizationRequest.

AuthCode erhalten

Ein typischer Workflow zum Erhalten eines AuthCodes sieht wie folgt aus:

1. Aufruf der Authorization URL des ASPSP
2. Weiterleitung des Endkunden an die zurückgeliefert URL
3. Nach Kundenbestätigung Weiterleitung an die TPP URL ergänzt um dem Auth Code und den State

Im Anschluss kann das AccessToken unter Angabe von AuthCode/CodeVerifier abgerufen werden. Der standarisierte BGOAuth2AuthorizationRequest hat folgenden Aufbau:

```
{
    "Type": "BGOAuth2AuthorisationRequest",
    "TPPData": {
        {{$tppData}}
    },
    "SecurityParameters": {
        "XRequestID": "{{$guid}}"
    },
    "OAuth2AuthorisationParameters": {
        "ResponseType": "CODE",
        "ClientId": "{{BGClientId}}",
        "State": "{{$guid}}",
        "Scope": "{{$scope}}",
        "RedirectUri": "{{tppRedirectUri}}",
        "CodeChallenge": "{{$codeChallenge}}"
    }
}
```

WSO2 Token abrufen - BGPreAuthWSO2TokenRequest

Abruf eines Access Tokens im Rahmen der WSO2 PreAuthentication.

Request:

```
"Orders": [
    {
      "PreAuthWSO2TokenParameters": {
        "Authorization": "<BasicAuth>"
      },
      "PreAuthWSO2Token": {
        "GrantType": "client_credentials"
      },
      "TPPData": {
        "ASPSPUrl": "https://api.dkb.de/oauth2/token",
        "TransportCertificateId": "<qwac>"
      },
      "SecurityParameters": {
        "XRequestID": "1a91e234-403f-4e6a-a9db-4f02e6119cbf"
      },
      "Type": "BGPreAuthWSO2TokenRequest"
    }
  ]
```

Response:

```
"Responses": [
    {
      "AccessToken": "<accessToken>",
      "Scope": "am_application_scope default",
      "TokenType": "Bearer",
      "ExpiresIn": 3600, ...
```

OAuth2 Token abrufen - BGPreAuthOAuthTokenRequest

Abruf eines Access Tokens im Rahmen der OAuth2 PreAuthentication.

Request:

```
"Orders": [
    {
      "PreAuthOAuthTokenParameters": {
        "Authorization": "Bearer <WSO2 AccessToken>",
        "PSD2AUTHORIZATION": "<BasicAuth> "
      },
      "PreAuthOAuthToken": {
        "Username": "<user>",
        "Password": "<password>"
      },
      "TPPData": {
        "ASPSPUrl": "https://api.dkb.de/pre-auth/psd2-auth/v1/auth/token",
        "TransportCertificateId": "<qwac>"
      },
      "SecurityParameters": {
        "XRequestID": "fc762198-fc50-4c2f-ae3d-3088fcbc05a9"
      },
      "Type": "BGPreAuthOAuthTokenRequest"
    }
  ]
```

Response:

```
"Responses": [
    {
      "ReturnCode": "CORRECT",
      "ActionCode": "PROMPT_FOR_TAN",
      "AuthTypeSelected": "SML",
      "Challenge": "<challenge>####Bestätigen sie den Login...[Pixel 4].",
      "ChallengeText": "#Bestätigen sie den Login ... [Pixel 4].",
      "TAN": "<TAN>",
      "XXSRFTOKEN": "<XXSRFTOKEN>",
      "SetCookie": "JSESSIONID=<id>",
```

App TAN übermitteln - BGPreAuthSendAppTANRequest

Übermittlung eine App TAN im Rahmen der PreAuthentication.

Request:

```
"Orders": [
    {
      "PreAuthSendAppTANParameters": {
        "Authorization": "Bearer <WSO2 AccessToken>",
        "XXSRFTOKEN": "<XXSRFTOKEN>",
        "Cookie": "JSESSIONID=<id>"
      },
      "PreAuthSendAppTAN": {
        "TAN": "<TAN>"
      },
      "TPPData": {
        "ASPSPUrl": "https://api.dkb.de/pre-auth/psd2-auth/v1/challenge?XSRF-TOKEN=",
        "TransportCertificateId": "<qwac>"
      },
      "SecurityParameters": {
        "XRequestID": "db5eb4ee-d0dc-4420-b26a-e5a4eddc31ff"
      },
      "Type": "BGPreAuthSendAppTANRequest"
    }
  ]
Response:
"Responses": [
    {
      "ReturnCode": "CORRECT",
      "ActionCode": "REDIRECT",
      "AccessToken": "<AccessToken>"
```

SSO Token abrufen - BGPreAuthSSOTokenRequest

Abruf eines Access Tokens im Rahmen der SSO PreAuthentication.

Request:

```
"Orders": [
    {
      "PreAuthSSOToken": {
        "GrantType": "client_credentials",
        "ClientId": "API-<id>",
        "Scope": "apic-psd2"
      },
      "TPPData": {
        "ASPSPUrl": "https://sso-psd2.raiffeisen.at/as/token.oauth2",
        "TransportCertificateId": "qwac"
      },
      "SecurityParameters": {
        "XRequestID": "5d42af89-8a44-4973-bf32-45456304c06a"
      },
      "Type": "BGPreAuthSSOTokenRequest"
    }
  ]
```

Response:

```
"Responses": [
    {
      "AccessToken": "<AccessToken>",
      "TokenType": "Bearer",
      "ExpiresIn": 600, ...
```

IDP Link abrufen - BGPreAuthIDPAccessRequest

Abruf eines Redirect Links im Rahmen der IDP PreAuthentication.

Requests:

```
 "Orders": [
    {
      "PreAuthIDPAccessParameters": {
        "IDPUrl": "https://idp.sparda-m.de/oauth2/authorize",
        "BIC": "GENODEF1S04",
        "ResponseType": "code",
        "ClientId": "PSDDE-BAFIN-XXXXXX",
        "Scope": "ais",
        "State": "5da85946-b3d7-4023-8c40-85e3c9cbb786",
        "RedirectUri": "https://<url>"
      },
      "Type": "BGPreAuthIDPAccessRequest"
    }
  ]
```

Response:

```
"Responses": [
    {
      "Link":
"https://idp.sparda-m.de:443/oauth2/authorize?bic=GENODEF1S04&client_id=PSDDE
-BAFIN-XXXXXX&redirect_uri=https%3a%2f%2fxxx.com&scope=ais&response_type=code
&code_challenge=hD8XmK0ctJfeUB8G8vWc7ImrfiziwDuo_1zA3RwkBoo&code_challenge_me
thod=S256&state=5da85946-b3d7-4023-8c40-85e3c9cbb786",
      "Type": "BGPreAuthIDPAccessResponse"
    }
  ]
```

OAuth2 Requests

Authorisierungsserver ermitteln - BGOAuth2DetermineMetadataRequest

Ermittlung des Authorisierungsservers und Token Endpoints anhand der scaOAuth URL bei Banken (Profil: ScaOAuth2LinkMetadata = true). Aktuell betrifft das z.B. alle deutschen Sparkassen.


```
{
      "TPPData": {
        "ASPSPUrl":
"https://xs2a.f-i-apim.de:8443/fixs2aop-env/oauth/70150000/.well-known",
        "TransportCertificateId": "qwac"
      },
      "SecurityParameters": {
        "XRequestID": "674da0da-ba24-49b7-8ed0-b1785093f011"
      },
      "Type": "BGOAuth2DetermineMetadataRequest"
    }
```


Die Antwort enthält die Token- / Authorisierungs-Endpoints. Der Authorisierungsendpoint wird für die Generierung des Authorsierungs-Links benötigt, der Token-Endpoint für den Abruf von Access und Refresh Tokens.


```
{
      "Issuer": "https://xs2a.f-i-apim.de:8443",
      "AuthorizationEndpoint": "https://www.sskm.de/services/xs2a/authorize",
      "TokenEndpoint":
"https://xs2a.f-i-apim.de:8443/fixs2aop-env/oauth/70150000/token",
      "XRequestID": null,
      "Type": "BGOAuth2DetermineMetadataResponse"
    }
```

Authorisierungs Link generieren - BGOAuth2GenerateAuthoriseRequest

Mit dem folgenden Request kann der Authorsierungs Link für den Kunden generiert werden, über den dieser die Zahlung freigeben oder einen Consent erteilen kann. Sofern bei der bei der Consenterteilung/Initiierung der Zahlung eingestellte Link bereits als URL für den Kunden gedacht ist, so ist der scaOAuth Wert zu verwenden, andernfalls das Ergebnis (Authorization Endpoint) der BGOAuth2DetermineMetadataResponse. Über den Parameter FiLInk wird das Format des erzeugten Links gesteuert. Dieser Wert kann den Profildaten entnommen werden.

```
{
      "OAuth2GenerateAuthoriseParameters": {
        "AuthorisationEndpoint":
"https://www.sskm.de/services/xs2a/authorize",
        "ClientId": "PSDDE-BAFIN-XXXXXX",
        "Scope": "AIS: 87e5380a-5a94-4810-9b8e-811a467882c4",
        "RedirectUri": "https://xxx.com",
        "FiLink": true
      },
      "Type": "BGOAuth2GenerateAuthoriseRequest"
    }
```

Die Response enthält nunmehr den Link, an der Kunde weitergeleitet werden soll:

```
      {
          "Link": <URL>,
          "CodeVerifier": <CodeVerifier>,
          "Type": "BGOAuth2GenerateAuthoriseResponse"
      }
```

An die im Request angegebene Redirect URL wird der URL Parameter “code” angehängt, der gegen ein Access Token getauscht werden kann.

Beispiel:
<tppRedirectURri>?code=7c977baafa3046cba09515e6d72afbd4&state=6872d54e-2bb5-4a85-89bd-51fdcfbd664a

Access / Refresh Token abrufen - BGOAuth2TokenGeneralRequest

Der Abruf von Access und Refresh Tokens kann mit dem BGOAuth2TokenGeneralRequest vorgenommen werden, der sowohl für unterschiedlich Standard OAuth2 Implementierungen als auch für ASPSP spezifische Erweiterungen verwendet werden kann.

BGOAuth2TokenGeneral Parameter

Feldname	**Typ**
ClientId	string
ClientSecret	string
Code	string
GrantType	string
RedirectUri	string
RefreshToken	string
Scope	string
Parameters	array

Beispiel Abruf Access Token:

```
{
            "Type": "BGOAuth2TokenGeneralRequest",
            "OrderId": "{{$guid}}",
            "TPPData": {
                {{$tppData}}
            },
            "OAuth2TokenGeneral": {
                "GrantType": "authorization_code",
                "Code": "{{$code}}",
                "RedirectUri": "{{tppRedirectUri}}",
                "Parameters": [
                    {
                        "Name": "request_id",
                        "Value": "{{$requestId}}"
```

```
                },
                {
                    "Name": "code_verifier",
                    "Value": "{{$codeVerifier}}"
                }
            ]
        }
```

Beispiel Abruf Refresh Token:

```
{
    "Type": "BGOAuth2TokenGeneralRequest",
    "OrderId": "{{$guid}}",
    "TPPData": {
        {{$tppData}}
    },
    "OAuth2TokenGeneral": {
        "GrantType": "refresh_token",
        "RefreshToken" : "{{refresh_token}}"
    }
}
```

Refresh Token widerrufen

Der Widerruf eines Refresh Tokens kann mit dem BGOAuth2RevokeTokenRequest vorgenommen werden. Bei der spezifischen Impementierung der ING kann durch den Widerruf des Refresh Tokens ein aktiver Consent gelöscht werden.

BGOAuth2RevokeToken Parameter

Feldname	**Typ**
Token	string
TokenTypeHint	string

Beispiel:

```
{
    "Type": "BGOAuth2RevokeTokenRequest",
    "TPPData": {
        {{$tppData}}
    },
    "SecurityParameters": {
        "XRequestID": "{{$guid}}"
    },
    "OAuth2RevokeToken": {
        "Token": "{{refresh_token}}"
    }
}
```

Extrahieren der (PSU) Daten aus einem JWT Access Token - BGDecodeJWTRequest

Extrahieren der Daten aus einem JWT Access Token.

Request:

```
"Orders": [
    {
      "DecodeJWTParameters": {
        "JWT": "<JWTAccessToken>"
      },
      "Type": "BGDecodeJWTRequest"
    }
  ]
```

Response:

```
"Responses": [
    {
      "Sub": "<PSU-ID>",
      "Aud": "PSDDE-BAFIN-XXXXX",
      "Azp": "TPP Name",
      "Iss": "https://idp.sparda.de",
      "Exp": 1584357790,
      "Iat": 1584357490,
      "Jti": "da7982dc-26c2-42d6-94f0-xxxxxxxxxxx",
      "Type": "BGDecodeJWTResponse"
    }
  ],
```

Kontoinformationen (AIS)

Redirect SCA

Consent erteilen

Voraussetzung für den Zugriff auf ein Zahlungsverkehrskonto ist das Vorliegen der Kundenzustimmung, die einem BGCreateConsentRequest gestartet wird. Aus dem Profil ist ersichtlich, welche SCA Methoden, Consent-Typen unterstützt und welche Zusatzparameter/Formate benötigt werden. Ein einfacher Zugriff am Beispiel der Consorsbank (Redirect) sieht wie folgt aus:

```
{
      "CreateConsentParameters": {
        "TPPRedirectPreferred": true,
        "TPPRedirectURI": "https://xxx.com",
        "TPPNokRedirectURI": "https://xxx.com",
        "TPPExplicitAuthorisationPreferred": false
      },
      "Consents": {
        "Access": {
          "Accounts": [
            {
              "Iban": "<IBAN>",
              "Currency": "EUR"
            }
          ],
          "Balances": [
            {
              "Iban": "<IBAN>",
              "Currency": "EUR"
            }
          ],
          "Transactions": [
            {
              "Iban": "<IBAN>",
              "Currency": "EUR"
            }
          ]
        },
```

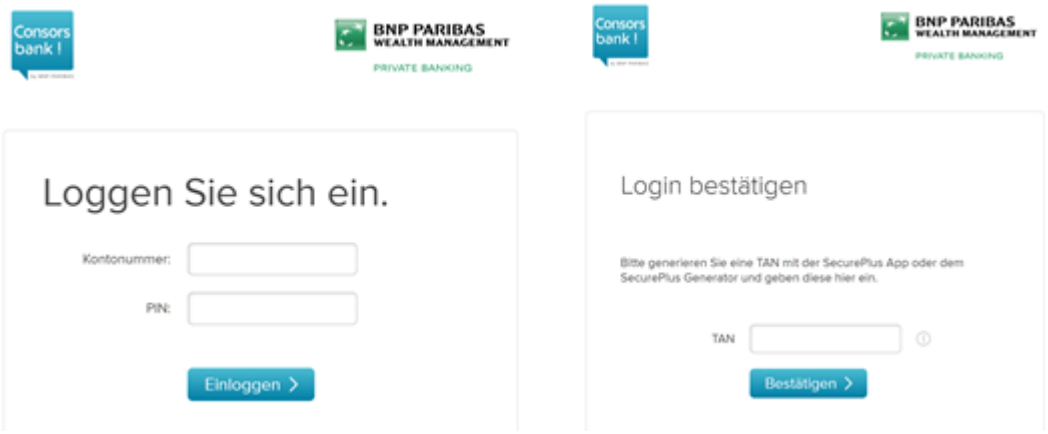
```
      "RecurringIndicator": true,
      "ValidUntil": "2020-06-14",
      "FrequencyPerDay": 4,
      "CombinedServiceIndicator": false
    },
    "PSUParameters": {
      "PSUIPAddress": "192.168.178.44"
    },
    "TPPData": {
      "ASPSPHost": "https://xs2a.consorsbank.de",
      "ASPSPPath": "/v1",
      "TransportCertificateId": "qwac"
    },
    "SecurityParameters": {
      "XRequestID": "3818013a-0e4c-464b-80fd-f82784e739d1"
    },
    "Type": "BGCreateConsentRequest"
  }
```

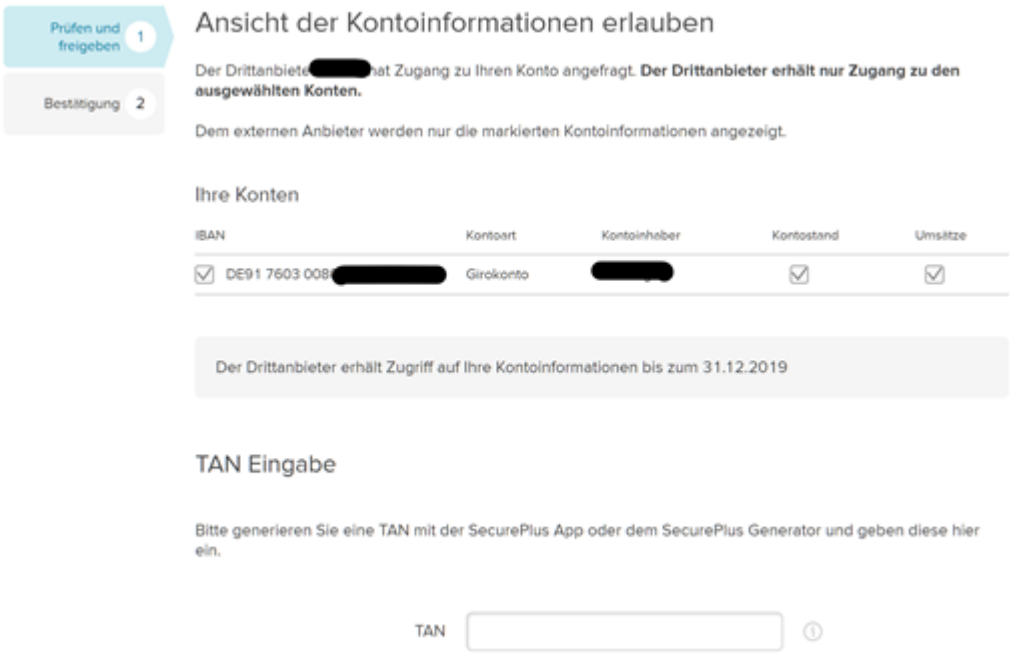

Die Auftragsparameter sind in dem jeweils aktuellen openapi File erklärt. Die Response würde in unserem Beispiel wie folgt aussehen:


```
{
    "AuthorisationId": "2ad73f3b-9481-4935-81c6-ab75cd2f051c",
    "ConsentStatus": "received",
    "ConsentId": "<consentId>",
    "Links": {
      "scaRedirect": {
        "href": "https://redirect.url"
      },
      "self": {
        "href": "https://self.url"
      },
      "status": {
        "href": "https://status.url"
      },
```

```
      "scaStatus": {
        "href": "https://scastatus.url"
      }
    },
    "XRequestID": "3818013a-0e4c-464b-80fd-f82784e739d1",
    "Type": "BGConsentsResponse"
  }
```

Der Kunde wird in diesem Fall an die SCA Redirect URL weitergeleitet und kann dort die Consent Freigabe vornehmen. Nach Abschluss der Freigabe kann über die Consent-ID im festgelegten Umfang auf die Kontoinformationen zugegriffen werden.





Embedded / Decoupled SCA

im Gegensatz zum Redirect ist es bei den SCA Varianten Embedded / Decoupled vorgesehen, dass die komplette oder Teile der Kundenauthentifizierung und Auftragsfreigabe durch den TPP vorgenommen werden. Nach Erstellung eines Consents sind in Abhängigkeit der angesteuerten Bank und der beim Kunden konfigurierten Sicherheitsverfahren / Medien ggfs. folgende Requests auszuführen.

Der erforderliche Request ist anhand der Links, die in der BGCreateConsentsResponse zurückgeliefert werden, ersichtlich. Ein typischer Link zum Starten des Authorisierungsprozesses und der Übermittlung der PSU Daten könnte wie folgt aussehen:

```
"Links": {
        "startAuthorisationWithPsuAuthentication": {
          "href": "/v1/consents/<consentId>/authorisations"
        }
      }
```

Consent erteilen

Nachstehend wird die Erteilung eines Consents anhand eines an die Fiducia angeschlossenen Instituts dargestellt:

```
{
      "CreateConsentParameters": {
        "PSUID": "<PUSID>",
        "TPPRedirectPreferred": false,
        "TPPExplicitAuthorisationPreferred": false
      },
      "Consents": {
        "Access": {
          "Accounts": [
            {
              "Iban": "IBAN",
              "Currency": "EUR"
            }
          ],
          "Balances": [
            {
              "Iban": "IBAN",
              "Currency": "EUR"
            }
```

```
        ],
        "Transactions": [
          {
            "Iban": "IBAN",
            "Currency": "EUR"
          }
        ]
      },
      "RecurringIndicator": true,
      "ValidUntil": "2020-06-14",
      "FrequencyPerDay": 4,
      "CombinedServiceIndicator": false
    },
    "PSUParameters": {
      "PSUIPAddress": "192.168.178.44"
    },
    "TPPData": {
      "ASPSPHost": "https://www.sparda-sw.de/",
      "ASPSPPath": "/services_xs2a/v1",
      "TransportCertificateId": "qwac",
      "SignatureCertificateId": "qseal"
    },
    "SecurityParameters": {
      "XRequestID": "1dac7ac9-0e10-489b-a0b5-fc73e810aa7e"
    },
    "Type": "BGCreateConsentRequest"
  }
```

Response:

```
{
    "ConsentStatus": "received",
    "ConsentId": "consentId",
    "Links": {
      "startAuthorisationWithPsuAuthentication": {
        "href": "..."
      }
    },
    "ASPSPSCAApproach": "EMBEDDED",
```

```
    "XRequestID": "1dac7ac9-0e10-489b-a0b5-fc73e810aa7e",

    "Type": "BGConsentsResponse"
  }
```

Starten den Authorisierungsprozesses

Anlegen einer neuen Authorisierungs-Ressource, die in der Antwort den Link zum weiteren Vorgehen enthält, z.B. zur Auswahl des Authentifizierungsverfahrens

```
"Links": {
      "startAuthorisationWithPsuAuthentication": {
        "href": "/v1/consents/<consentId>/authorisations"
      }
    }
```

BGStartConsentAuthorisationUpdatePsuAuthenticationRequest

Angabe der StartConsentAuthorisationParameters (ConsentId, PSUID …) und der PsuData (Passwort / Verschlüsseltes Passwort)

Beispiel:

```
{
    "StartConsentAuthorisationParameters": {
      "ConsentId": "consentId",
      "PSUID": "PSUID"
    },
    "UpdatePsuAuthentication": {
      "PsuData": {
        "Password": "Password"
      }
    },
    "PSUParameters": {
      "PSUIPAddress": "192.168.178.44"
    },
    "TPPData": {
      "ASPSPHost": "https://www.sparda-sw.de/",
      "ASPSPPath": "/services_xs2a/v1",
      "TransportCertificateId": "qwac",
      "SignatureCertificateId": "qseal"
    },
```

```
    "SecurityParameters": {
      "XRequestID": "47f573ca-15ad-4cb2-9c36-51a1bb2897a7"
    },
    "Type": "BGStartConsentAuthorisationUpdatePsuAuthenticationRequest"
  }
```


In der Response werden nunmehr der neue ScaStatus zurückgeliefert und die erforderlichen nächsten Schritte, z.B. Auswahl der Authentication Methode, angezeigt.


```
{
    "ScaStatus": "psuAuthenticated",
    "AuthorisationId": "AuthorizationId",
    "ScaMethods": [
      {
        "AuthenticationType": "SMS_OTP",
        "AuthenticationMethodId": "942",
        "Name": "mobile TAN"
      },
      {
        "AuthenticationType": "PHOTO_OTP",
        "AuthenticationMethodId": "982",
        "Name": "Smart-TAN photo"
      }
    ],
    "Links": {
      "selectAuthenticationMethod": {
        "href": "/v1/consents/consentID/authorisations/authId"
      }
    },
    "ASPSPSCAApproach": "EMBEDDED",
    "Type": "BGStartScaProcessResponse"
  }
```


BGStartConsentAuthorisationSelectPsuAuthenticationMethodRequest

Angabe der StartConsentAuthorisationParameters (ConsentId, PSUID …) und der SelectPsuAuthenticationMethod (Id der Authentifizierungsmethode) durch die die Authorisierungsressource im Rahmen der Übermittlung der Authentifizierungsmethode angelegt wird

BGStartConsentAuthorisationTransactionAuthorisationRequest

Angabe der StartConsentAuthorisationParameters (ConsentId, PSUID …) und der TransactionAuthorisation (Authentifizierungsdaten, z.B. eine TAN)

BGStartConsentAuthorisationRequest

Legt eine Authorisierungsressource an, ohne dass weitere Authorisierungsparameter angegeben werden müssen.

Aktualisieren der PSU Daten

Sofern die Authorisierungs-Resource schon angelegt wurde, z.B. automatisch bei der Consenterteilung oder durch expliziten Start der Authorisierung, so können die Authentifizierungs-/Autorisierungsdaten aktualisiert werden. Zum Beispiel ist auf folgende Antwort die Authentication Methode festzulegen:

```
"Links": {
        "selectAuthenticationMethod": {
          "href": "/v1/consents/consentID/authorisations/authId"
        }
      }
```

BGUpdateConsentsPsuDataUpdatePsuAuthenticationRequest

Angabe der UpdateConsentsPsuDataParameters (ConsentId, AuthorisationId, PSUID …) und der PsuData (Passwort / Verschlüsseltes Passwort).

BGUpdateConsentsPsuDataSelectPsuAuthenticationMethodRequest

Angabe der UpdateConsentsPsuDataParameters (ConsentId, AuthorisationId, PSUID …) und der SelectPsuAuthenticationMethod (Id der Authentifizierungsmethode)

Beispiel:

```
{
      "UpdateConsentsPsuDataParameters": {
        "ConsentId": "consentId",
        "AuthorisationId": "authorizationId",
        "PSUID": "xxx"
      },
      "SelectPsuAuthenticationMethod": {
        "AuthenticationMethodId": "942"
      },
```

```
      "PSUParameters": {
        "PSUIPAddress": "192.168.178.44"
      },
      "TPPData": {
        "ASPSPHost": "https://www.sparda-sw.de/",
        "ASPSPPath": "/services_xs2a/v1",
        "TransportCertificateId": "qwac",
        "SignatureCertificateId": "qseal"
      },
      "SecurityParameters": {
        "XRequestID": "54bc6b60-35b8-4230-954c-201459af7eff"
      },
      "Type": "BGUpdateConsentsPsuDataSelectPsuAuthenticationMethodRequest"
    }
```


Die Antwort beinhaltet nunmehr den aktualisierten SCA Status sowie die Daten der TAN Challenge:


```
{
      "ChallengeData": {
        "OtpMaxLength": null,
        "AdditionalInformation": "Ihre TAN wurde als SMS an die Nummer XXX
gesendet."
      },
      "Links": {
        "authoriseTransaction": {
          "href": "/v1/consents/consentId/authorisations/authId"
        }
      },
      "ScaStatus": "started",
      "ASPSPSCAApproach": "EMBEDDED",
      "XRequestID": "54bc6b60-35b8-4230-954c-201459af7eff",
      "Type": "BGUpdatePsuAuthenticationResponse"
    }
```


BGUpdateConsentsPsuDataTransactionAuthorisationRequest

Angabe der UpdateConsentsPsuDataParameters (ConsentId, AuthorisationId, PSUID …) und der TransactionAuthorisation (Authentifizierungsdaten, z.B. eine TAN)

Beispiel:

```
{
      "UpdateConsentsPsuDataParameters": {
        "ConsentId": "consentId",
        "AuthorisationId": "authorisationId",
        "PSUID": "xxx"
      },
      "TransactionAuthorisation": {
        "ScaAuthenticationData": "123456"
      },
      "PSUParameters": {
        "PSUIPAddress": "192.168.178.44"
      },
      "TPPData": {
        "ASPSPHost": "https://www.sparda-sw.de/",
        "ASPSPPath": "/services_xs2a/v1",
        "TransportCertificateId": "qwac",
        "SignatureCertificateId": "qseal"
      },
      "SecurityParameters": {
        "XRequestID": "f1994fc4-73e2-4ad7-8d7d-0c479e289594"
      },
      "Type": "BGUpdateConsentsPsuDataTransactionAuthorisationRequest"
    }
```


Nach einer erfolgreichen Authorisierung wird wiederum der aktualisierte SCA Status geliefert.


```
{
      "ScaStatus": "finalised",
      "XRequestID": "f1994fc4-73e2-4ad7-8d7d-0c479e289594",
      "Type": "BGUpdatePsuIdenticationResponse"
}
```


BGUpdateConsentsPsuDataRequest

Aktualisiert eine Authorisierungsressource, ohne dass weitere Authorisierungsparameter angegeben werden müssen.

Passwort verschlüsseln - BGEncryptPasswordWithAspspCertificateLinkRequest

Einige Institute verlangen im Rahmen der Embedded SCA die verschlüsselte Übermittlung des Benutzerpassworts. Der folgende Request am Beispiel der Postbank zeigt die Verschlüsselung des Passworts unter Verwendung der zuvor gelieferten Zertifikats-URL.

```
{
	"Type": "BGEncryptPasswordWithAspspCertificateLinkRequest",
	"TPPData": {
		"ASPSPUrl": "{{BGAspspCertificates}}",
		"TransportCertificateId": "{{BGTransportCertificateId}}"
	},
	"SecurityParameters": {
		"XRequestID": "{{$guid}}"
	},
	"EncryptPasswordWithAspspCertificateLinkParameters": {
		"Password": "password"
	}
}
```

Consent löschen

Löscht einen zuvor erteilten Consent:

```
{
	"DeleteConsentParameters": {
	  "ConsentId": "consentId"
	},
	"PSUParameters": {
	  "PSUIPAddress": "192.168.178.44"
	},
	"TPPData": {
	  "ASPSPHost": "https://www.sparda-sw.de/",
	  "ASPSPPath": "/services_xs2a/v1",
	  "TransportCertificateId": "qwac",
	  "SignatureCertificateId": "qseal"
	},
	"SecurityParameters": {
	  "XRequestID": "6e748fc7-9af2-4b6b-977a-bbe9450eae48"
	},
	"Type": "BGDeleteConsentRequest"
    }
```

Consent SCA Status abfragen

Abfrage des SCA Status für die angegebene Consent-ID/Authorization-ID:

```
{
    "Type": "BGGetConsentScaStatusRequest",
    "TPPData": {
        "ASPSPHost": "{{BGHost}}",
        "ASPSPPath": "{{BGAISPath}}",
        "TransportCertificateId": "{{BGTransportCertificateId}}"
    },
    "SecurityParameters": {
        "XRequestID": "{{$guid}}"
    },
    "PSUParameters": {
        "PSUIPAddress": "{{psuIpAddress}}"
    },
    "GetConsentScaStatusParameters": {
        "ConsentId": "{{consentId}}",
        "AuthorisationId": "{{authorisationId}}"
    }
}
```

Die Antwort enthält den SCA Status:

```
"Responses": [
    {
        "ScaStatus": "psuIdentified",
        "Type": "BGGetConsentScaStatusResponse"
    }
]
```

Gültige Werte für den SCA-Status sind:

"received", "psuIdentified", "psuAuthenticated", "scaMethodSelected", "started", "finalised", "failed", "exempted".

Consent Status abfragen

Abfrage des Consent Status für die angegebene Consent-ID:

```
{
      "GetConsentStatusParameters": {
        "ConsentId": "consentId"
      },
      "PSUParameters": {
        "PSUIPAddress": "192.168.178.44"
      },
      "TPPData": {
        "ASPSPHost": "https://www.sparda-sw.de/",
        "ASPSPPath": "/services_xs2a/v1",
        "TransportCertificateId": "qwac",
        "SignatureCertificateId": "qseal"
      },
      "SecurityParameters": {
        "XRequestID": "6e748fc7-9af2-4b6b-977a-bbe9450eae48"
      },
      "Type": "BGGetConsentStatusRequest"
    }
```

Die Antwort beinhaltet den Consent Status:

```
{
      "ConsentStatus": "valid",
      "XRequestID": "6e748fc7-9af2-4b6b-977a-bbe9450eae48",
      "Type": "BGConsentStatusResponse"
    }
```

Gültige Werte für den Consent-Status sind:

"received", "rejected", "valid", "revokedByPsu", "expired", "terminatedByTpp"

Consent Informationen abfragen

Abfrage des Consent Informationen für die angegebene Consent-ID:

```
{
    "Type": "BGGetConsentInformationRequest",
    "TPPData": {
        "ASPSPHost": "{{BGHost}}",
        "ASPSPPath": "{{BGAISPath}}",
        "TransportCertificateId": "{{BGTransportCertificateId}}",
        "CustomHeaders": [
            {{BGCustomHeader}}
        ]
    },
    "SecurityParameters": {
        "XRequestID": "{{$guid}}"
    },
    "PSUParameters": {
        "PSUIPAddress": "{{psuIpAddress}}"
    },
    "GetConsentInformationParameters": {
        "ConsentId": "{{consentId}}"
}
```

Die Antwort beinhaltet neben dem Consent Status auch alle Informationen zum erzeugten Consent, z.B. die Gültigkeitsdauer, Abfragefrequenz und die beinhalteten Konten:

```
"Access": {
        "Accounts": [
            {
                "Iban": "DE9628020050XXXXXXXXXX",
                "Bban": "XXXXXXXXXX",
                "Currency": "EUR"
            }
        ],
        "Balances": [
```

```
            {
                "Iban": "DE96280200 50XXXXXXXXXX",
                "Bban": "XXXXXXXXXXX",
                "Currency": "EUR"
            }
        ],
        "Transactions": [
            {
                "Iban": "DE96280200 50XXXXXXXXXX",
                "Bban": "XXXXXXXXXXX",
                "Currency": "EUR"
            }
        ],
        "AdditionalInformation": {
            "OwnerName": [
                {
                    "Iban": "DE96280200 50XXXXXXXXXX",
                    "Bban": "XXXXXXXXXXX",
                    "Currency": "EUR"
                }
            ]
        }
    },
    "RecurringIndicator": true,
    "ValidUntil": "2021-12-05T00:00:00",
    "FrequencyPerDay": 4,
    "LastActionDate": "2021-10-05T00:00:00",
    "ConsentStatus": "received"
```


Gültige Werte für den Consent-Status sind:

"received", "rejected", "valid", "revokedByPsu", "expired", "terminatedByTpp"

Kontoliste lesen

Lesen der Kontoliste mit oder ohne Saldeninformation in Abhängigkeit des erteilten Consents. Es ist zu beachten das nicht alle Institute das gleichzeitige Lesen der Salden erlaubt. Diese Information ist in den Profiledaten enthalten.

Die Antwort beinhaltet neben den Kontoinformationen IDs, die für die Abfrage weiterer Informationen (Kontodetails, Salden, Umsätze) benötigt werden:

```
{
      "GetAccountListParameters": {
        "ConsentId": "consentid",
        "WithBalance": false
      },
      "PSUParameters": {
        "PSUIPAddress": "192.168.178.44"
      },
      "TPPData": {
        "ASPSPHost": "https://www.sparda-sw.de/",
        "ASPSPPath": "/services_xs2a/v1",
        "TransportCertificateId": "qwac",
        "SignatureCertificateId": "qseal"
      },
      "SecurityParameters": {
        "XRequestID": "ed5ec6b8-6098-408f-89a9-cbe29def3d74"
      },
      "Type": "BGGetAccountListRequest"
    }
```

Die Antwort enthält die Kontenliste:

```
    {
      "Accounts": [
        {
          "ResourceId": "resourceId",
          "Iban": "DE6555090500000XXXXXX",
          "Currency": "EUR",
          "Name": "Kontokorrent",
          "OwnerName": "Klaus Igel",
```

```
            "Bic": "GENODEF1S01"
          }
        ],
        "XRequestID": "ed5ec6b8-6098-408f-89a9-cbe29def3d74",
        "Type": "BGAccountListResponse"
      }
```

Kontodetails abfragen

Kontodetails mit oder ohne Salden für die angegebene Resource-ID abfragen.

```
{
    "Type": "BGReadAccountDetailsRequest",
    "TPPData": {
        "ASPSPHost": "{{BGHost}}",
        "ASPSPPath": "{{BGAISPath}}",
        "TransportCertificateId": "{{BGTransportCertificateId}}",
    },
    "SecurityParameters": {
        "XRequestID": "{{$guid}}"
    },
    "PSUParameters": {
        "PSUIPAddress": "{{psuIpAddress}}"
    },
    "ReadAccountDetailsParameters": {
        "ConsentId": "{{consentId}}",
        "AccountId": "{{resourceId}}",
        "WithBalance": "{{withBalance}}"
    }
}
```

Die Antwort kann neben den Kontodetails auch die Links zur Abfrage der Salden und Transaktionen beinhalten.

```
"Responses": [
    {
        "ResourceId": "39e5c2fec02056fad774756a4c276805124XXXX",
        "Iban": "DE917603008002501XXXXX",
        "Currency": "EUR",
```

```
            "CashAccountType": "CACC",
            "Status": "enabled",
            "Bic": "CSDBDE71XXX",
            "Balances": [
                {
                    "BalanceAmount": {
                        "Currency": "EUR",
                        "Amount": "59.99"
                    },
                    "BalanceType": "interimAvailable"
                },
                {
                    "BalanceAmount": {
                        "Currency": "EUR",
                        "Amount": "54.99"
                    },
                    "BalanceType": "closingBooked"
                }
            ],
            "Links": {
                "balances": {
                    "href": "balancesUrl"
                },
                "transactions": {
                    "href": "transactionsUrl"
                }
            },
            "XRequestID": "e53f2b2d-d845-4c2f-857f-b783a7ee170c",
            "Type": "BGAccountDetailsResponse"
        }
    ],
```

Saldenabfrage

Saldenabfrage für die angegebene Resource-ID.


```
{
      "GetBalancesParameters": {
        "AccountId": "accountId",
        "ConsentId": "consentId"
      },
      "PSUParameters": {
        "PSUIPAddress": "192.168.178.44"
      },
      "TPPData": {
        "ASPSPHost": "https://www.sparda-sw.de/",
        "ASPSPPath": "/services_xs2a/v1",
        "TransportCertificateId": "qwac",
        "SignatureCertificateId": "qseal"
      },
      "SecurityParameters": {
        "XRequestID": "381437e3-6eb3-4fbc-83a5-7dd71238cad7"
      },
      "Type": "BGGetBalancesRequest"
    }
```


Die Response beinhaltet die verfügbaren Saldoinformationen:


```
{
      "Balances": [
        {
          "BalanceAmount": {
            "Currency": "EUR",
            "Amount": "58.12"
          },
          "BalanceType": "closingBooked"
        }
      ],
      "XRequestID": "381437e3-6eb3-4fbc-83a5-7dd71238cad7",
      "Type": "BGBalancesResponse"
    }
```

Umsatzabfrage

Umsatzabfrage für die angegebene Resource-ID.


```
{
      "GetTransactionListParameters": {
        "AccountId": "accountId",
        "DateFrom": "2019-12-19",
        "BookingStatus": "booked",
        "ConsentId": "consentId"
      },
      "PSUParameters": {
        "PSUIPAddress": "192.168.178.44"
      },
      "TPPData": {
        "ASPSPHost": "https://www.sparda-sw.de/",
        "ASPSPPath": "/services_xs2a/v1",
        "TransportCertificateId": "qwac",
        "SignatureCertificateId": "qseal"
      },
      "SecurityParameters": {
        "XRequestID": "88376ec9-b70b-4282-b478-4c43f2e92ccb"
      },
      "Type": "BGGetTransactionListRequest"
    }
```


Neben dem Datum, ab dem die Umsätze abgefragt werden sollen muss noch der Buchungsstatus (booked, pending, both) mit angegeben werden.

In der Antwort ist nunmehr eine Liste der verfügbaren Umsätze enthalten. In welchem Format die Umsätze zurückgeliefert werden, kann den Profilinformationen entnommen werden.


```
{
      "Account": {
        "Iban": "IBAN",
        "Currency": "EUR"
      },
```

```
      "Transactions": {
        "Booked": [
          {
            "EndToEndId": "NOTPROVIDED",
            "BookingDate": "2020-01-15T00:00:00",
            "ValueDate": "2020-01-16T00:00:00",
            "TransactionAmount": {
              "Currency": "EUR",
              "Amount": "-0.01"
            },
            "CreditorName": "Klaus Igel",
            "CreditorAccount": {
              "Iban": "IBAN"
            },
            "DebtorName": "Klaus Igel",
            "DebtorAccount": {
              "Iban": "IBAN"
            },
            "RemittanceInformationUnstructured": "TAN1:Auftrag nicht
TAN-pflichtig",
            "ProprietaryBankTransactionCode": "NTRF+177+00804"
          }
        ],
        "Links": {
          "account": {
            "href": "/v1/accounts/id"
          }
        }
      },
      "Balances": [
        {
          "BalanceAmount": {
            "Currency": "EUR",
            "Amount": "xxx"
          },
          "BalanceType": "openingBooked",
          "ReferenceDate": "2020-01-15"
        },
        {
```

```
        "BalanceAmount": {
          "Currency": "EUR",
          "Amount": "xxx"
        },
        "BalanceType": "closingBooked",
        "ReferenceDate": "2020-03-09"
      }
    ],
    "Type": "BGAccountsTransactionsResponse"
  }
```


Wichtig: Es ist zu beachten das die Formate, in denen die Umsatzdaten bereitgestellt werden, für die einzelnen Banken unterschiedlich sein können. Neben dem oben dargestellten JSON Format können Umsätze auch im CAMT oder MT940 Format bereitgestellt werden. Diese Information kann den Profildaten entnommen werden.

Zahlungsauslösung (PIS)

Zahlung initiieren

Über den folgende Request kann am Beispiel der consorsbank eine Zahlung eingestellt werden. Die unterstützten Payment Services und Payment Types sind in den Profildaten enthalten:

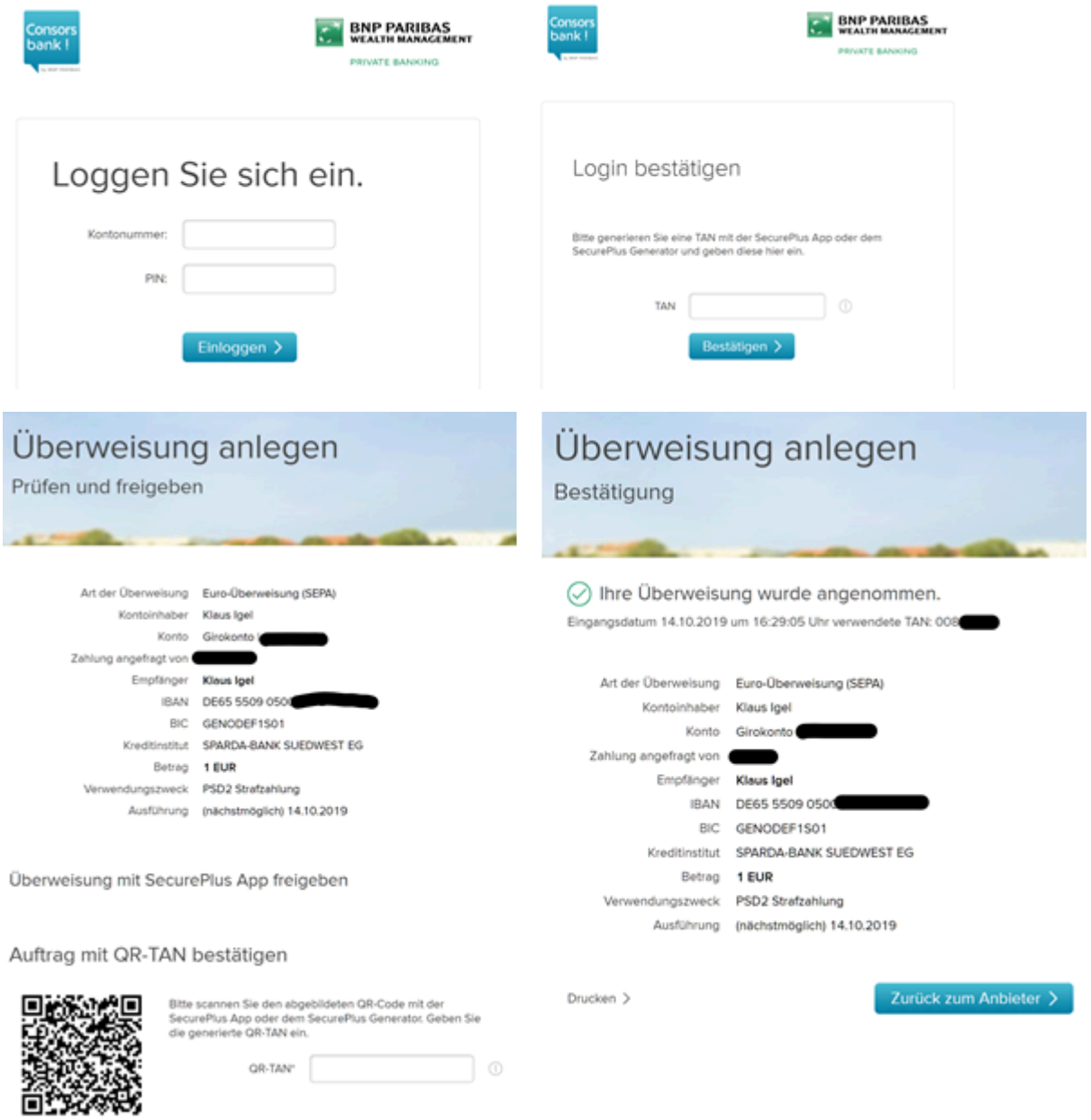
```
{
            "Type": "BGInitiatePaymentRequest",
            "TPPData": {
                "ASPSPHost": "{{BGHost}}",
                "ASPSPPath": "{{BGPISPath}}",
                "TransportCertificateId": "{{BGTransportCertificateId}}"
            },
            "SecurityParameters": {
                "XRequestID": "{{$guid}}"
            },
            "PSUParameters": {
                "PSUIPAddress": "{{psuIpAddress}}"
            },
            "InitiatePaymentParameters": {
                "TPPNokRedirectURI": "{{tppNokRedirectUri}}",
                "TPPRedirectURI": "{{tppRedirectUri}}",
                "TPPRedirectPreferred": "{{tppRedirectPreferred}}",
                "PaymentService": "{{paymentService}}",
                "PaymentProduct": "{{paymentProduct}}"
            },
            "PaymentInitiation": {
                "debtorAccount": {
                   "currency": "EUR",
                    "iban": "{{BGIban}}"
                },
                "instructedAmount": {
                    "currency": "EUR",
                    "amount": {{instructedAmount}}
                },
                "creditorAccount": {
                   "currency": "EUR",
                   "iban":"{{creditorAccount}}"
```

```
                },
                "creditorName": "{{creditorName}}",
                "remittanceInformationUnstructured":
"{{remittanceInformationUnstructured}}"
            }
        }
```

Die Antwort beinhaltet neben den Links immer auch die Payment-ID, die in allen Folgerequests angegeben werden muss.

```
{
            "TransactionStatus": "RCVD",
            "PaymentId": <PaymentID>
            "TransactionFeeIndicator": false,
            "Links": {
                "scaRedirect": {
                    "href": <redirectURL>
                },
                "self": {
                    "href": <selfURL>
                },
                "status": {
                    "href": <statusURL>
                },
                "scaStatus": {
                    "href": <scaStatusURL>
                }
            },
            "Type": "BGInitiatePaymentResponse"
        }
```

Der Kunde wird an die Redirect URL weitergeleitet und gibt hierüber die Zahlung frei



Embedded / Decoupled SCA

im Gegensatz zum Redirect ist es bei den SCA Varianten Embedded / Decoupled vorgesehen, dass die komplette oder Teile der Kundenauthentifizierung und Auftragsfreigabe durch den TPP vorgenommen werden. Nach Initiierung einer Zahlung sind in Abhängigkeit der angesteuerten Bank und der beim Kunden konfigurierten Sicherheitsverfahren / Medien ggfs. folgende Requests auszuführen.

Der erforderliche Request ist anhand der Links, die in der BGPaymentInitiationResponse zurückgeliefert werden, ersichtlich. Ein typischer Link zum Starten des Authorisierungsprozesses und der Übermittlung der PSU Daten könnte wie folgt aussehen:

```
"Links": {
                "startAuthorisationWithPsuAuthentication": {
                    "href":
"/v1/payments/pain.001-sepa-credit-transfers/consentId/authorisations"
                }
            }
```


Starten den Authorisierungsprozesses

Anlegen einer neuen Authorisierungs-Ressource, die in der Antwort den Link zum weiteren Vorgehen enthält, z.B. zur Auswahl des Authentifizierungsverfahrens


```
"Responses": [
        {
            "ScaStatus": "psuAuthenticated",
            "ScaMethods": [
                {
                    "AuthenticationType": "CHIP_OTP",
                    "AuthenticationVersion": "1.4",
                    "AuthenticationMethodId": "972",
                    "Name": "SmartTAN optic/USB HHD 1.4"
                },
                {
                    "AuthenticationType": "CHIP_OTP",
                    "AuthenticationVersion": "1.4",
                    "AuthenticationMethodId": "962",
                    "Name": "SmartTAN plus HHD 1.4"
                },
                {
                    "AuthenticationType": "PUSH_OTP",
                    "AuthenticationMethodId": "944",
                    "Name": "apoTAN"
                }
            ],
            "Links": {
                "selectAuthenticationMethod": {
                    "href":
"/v1/payments/pain.001-sepa-credit-transfers/paymentId/authorisations/authorisation
Id"
```

```
            }
        },
        "XRequestID": "68482186-0be0-4dbc-8e36-a83fc611c6d4",
        "ASPSPSCAApproach": "EMBEDDED",
        "Type": "BGStartScaProcessResponse"
    }
```


BGStartPaymentAuthorisationUpdatePsuAuthenticationRequest

Angabe der StartPaymentAuthorisationParameters (PaymentId, PSUID …) und der PsuData (Passwort / Verschlüsseltes Passwort)

BGStartPaymentAuthorisationSelectPsuAuthenticationMethodRequest

Angabe der StartPaymentAuthorisationParameters (PaymentId, PSUID …) und der SelectPsuAuthenticationMethod (Id der Authentifizierungsmethode)

BGStartPaymentAuthorisationTransactionAuthorisationRequest

Angabe der StartPaymentAuthorisationParameters (PaymentId, PSUID …) und der TransactionAuthorisation (Authentifizierungsdaten, z.B. eine TAN)

BGStartPaymentAuthorisationRequest

Legt eine Authorisierungsressource an, ohne dass weitere Authorisierungsparameter angegeben werden müssen.

Aktualisieren der PSU Daten

Sofern die Authorisierungs-Resource schon angelegt wurde, wird ein nachfolgend erforderlicher Schritt im Rahmen der Aktualisierung der Authentifizierungs-/Autorisierungsdaten durchgeführt.

BGUpdatePaymentPsuDataUpdatePsuAuthenticationRequest

Angabe der UpdatePaymentPsuDataParameters (PaymentId, AuthorisationId, PSUID …) und der PsuData (Passwort / Verschlüsseltes Passwort)

BGUpdatePaymentPsuDataSelectPsuAuthenticationMethodRequest

Angabe der UpdatePaymentPsuDataParameters (PaymentId, AuthorisationId, PSUID …) und der SelectPsuAuthenticationMethod (Id der Authentifizierungsmethode)

BGUpdatePaymentPsuDataTransactionAuthorisationRequest

Angabe der UpdatePaymentPsuDataParameters (PaymentId, AuthorisationId, PSUID …) und der TransactionAuthorisation (Authentifizierungsdaten, z.B. eine TAN).

Löschen einer zuvor eingestellen Zahlung

Löschen einer zuvor eingestellten Zahlung. Je nach Umsetzung beim ASPSP kann eine SCA des Kunden erforderlich sein. Ferner ist zu berücksichtigen, dass bestimmte Zahlungsarten nicht mehr gelöscht werden können.

```
{
      "Type": "BGCancelPaymentRequest",
      "TPPData": {
          "ASPSPHost": "{{BGHost}}",
          "ASPSPPath": "{{BGPISPath}}",
          "TransportCertificateId": "{{BGTransportCertificateId}}"
      },
      "SecurityParameters": {
          "XRequestID": "{{$guid}}"
      },
      "PSUParameters": {
          "PSUIPAddress": "{{psuIpAddress}}"
      },
      "CancelPaymentParameters": {
          "PaymentId": "{{paymentId}}",
          "PaymentService": "{{paymentService}}",
          "PaymentProduct": "{{paymentProduct}}"
      }
  }
```

Zahlung SCA Status abfragen

Abfrage des SCA Status für die angegebene Payment-ID/Authorization-ID:

```
{
            "Type": "BGGetPaymentInitiationScaStatusRequest",
            "TPPData": {
                "ASPSPHost": "{{BGHost}}",
                "ASPSPPath": "{{BGPISPath}}",
                "TransportCertificateId": "{{BGTransportCertificateId}}",
                "TransportCertificatePassword":
"{{BGTransportCertificatePassword}}"
            },
            "SecurityParameters": {
                "XRequestID": "{{$guid}}"
            },
            "PSUParameters": {
                "PSUIPAddress": "{{psuIpAddress}}"
            },
            "GetPaymentInitiationScaStatusParameters": {
                "PaymentId": "{{paymentId}}",
                "AuthorisationId": "{{authorisationId}}",
                "PaymentService": "{{paymentService}}",
                "PaymentProduct": "{{paymentProduct}}"
            }
        }
```

Die Antwort enthält den SCA Status:

```
{
    "ScaStatus": "finalised",
    "Type": "BGGetPaymentInitiationScaStatusResponse"
}
```

Gültige Werte für den SCA-Status sind:

"received", "psuIdentified", "psuAuthenticated", "scaMethodSelected", "started", "finalised", "failed", "exempted"

Zahlung Status abfragen

Abfrage des Zahlungs-Status für die angegebene Payment-ID:

```
{
            "Type": "BGGetPaymentInitiationStatusRequest",
            "TPPData": {
                "ASPSPHost": "{{BGHost}}",
                "ASPSPPath": "{{BGPISPath}}",
                "TransportCertificateId": "{{BGTransportCertificateId}}"
            },
            "SecurityParameters": {
                "XRequestID": "{{$guid}}"
            },
            "PSUParameters": {
                "PSUIPAddress": "{{psuIpAddress}}"
            },
            "GetPaymentInitiationStatusParameters": {
                "PaymentId": "{{paymentId}}",
                "PaymentService": "{{paymentService}}",
                "PaymentProduct": "{{paymentProduct}}"
            }
        }
```

Der Transaktionsstatus in der Response wird mit den Codes aus der ISO 20022 Tabelle gefüllt.

```
{
        "TransactionStatus": "ACSC",
        "Type": "BGGetPaymentInitiationStatusResponse"
    }
```

Auszug aus der Dokumentation der Berlin Group:

The transaction status is filled with codes of the ISO 20022 data table:

- 'ACCC': 'AcceptedSettlementCompleted' - Settlement on the creditor's account has been completed.
- 'ACCP': 'AcceptedCustomerProfile' - Preceding check of technical validation was successful. Customer profile check was also successful.
- 'ACSC': 'AcceptedSettlementCompleted' - Settlement on the debtor◆s account has been completed.
  Usage: this can be used by the first agent to report to the debtor that the transaction has been completed.
  Warning: this status is provided for transaction status reasons, not for financial information. It can only be used after bilateral agreement.
- 'ACSP': 'AcceptedSettlementInProcess' - All preceding checks such as technical validation and customer profile were successful and therefore the payment initiation has been accepted for execution.
- 'ACTC': 'AcceptedTechnicalValidation' - Authentication and syntactical and semantical validation are successful.
- 'ACWC': 'AcceptedWithChange' - Instruction is accepted but a change will be made, such as date or remittance not sent.
- 'ACWP': 'AcceptedWithoutPosting' - Payment instruction included in the credit transfer is accepted without being posted to the creditor customer◆s account.
- 'RCVD': 'Received' - Payment initiation has been received by the receiving agent.
- 'PDNG': 'Pending' - Payment initiation or individual transaction included in the payment initiation is pending. Further checks and status update will be performed.
- 'RJCT': 'Rejected' - Payment initiation or individual transaction included in the payment initiation has been rejected.
- 'CANC': 'Cancelled' Payment initiation has been cancelled before execution Remark: This codeis accepted as new code by ISO20022.
- 'ACFC': 'AcceptedFundsChecked' - Preceding check of technical validation and customer profile was successful and an automatic funds check was positive . Remark: This code is accepted as new code by ISO20022.
- 'PATC': 'PartiallyAcceptedTechnical' Correct The payment initiation needs multiple authentications, where some but not yet all have been performed. Syntactical and semantical validations are successful. Remark: This code is accepted as new code by ISO20022.
- 'PART': 'PartiallyAccepted' - A number of transactions have been accepted, whereas another number of transactions have not yet achieved 'accepted' status. Remark: This code may be used only in case of bulk payments. It is only used in a situation where all mandated authorisations have been applied, but some payments have been rejected.

Zahlungsinformationen abrufen

Abruf der Zahlungsinformationen für die angegebene Payment-ID:

```
{
            "Type": "BGGetPaymentInformationRequest",
            "TPPData": {
                "ASPSPHost": "{{BGHost}}",
                "ASPSPPath": "{{BGPISPath}}",
                "TransportCertificateId": "{{BGTransportCertificateId}}"
            },
            "SecurityParameters": {
                "XRequestID": "{{$guid}}"
            },
            "PSUParameters": {
                "PSUIPAddress": "{{psuIpAddress}}"
            },
            "GetPaymentInformationParameters": {
                "PaymentId": "{{paymentId}}",
                "PaymentService": "{{paymentService}}",
                "PaymentProduct": "{{paymentProduct}}"
            }
        }
```

Die Antwort enthält die Zahlungsdaten und deren Transaktionsstatus.


```
{
    "DebtorAccount": {
        "Iban": "DE9176030080025XXXXXXX",
        "Currency": "EUR"
    },
    "InstructedAmount": {
        "Currency": "EUR",
        "Amount": "1"
    },
    "CreditorAccount": {
        "Iban": "DE65550905000000XXXXXXX",
        "Currency": "EUR"
    },
```

```
        "CreditorName": "Klaus Igel",
        "CreditorAddress": {
            "Country": ""
        },
        "RemittanceInformationUnstructured": "PSD2 Strafzahlung",
        "TransactionStatus": "ACSC",
        "Type": "BGGetPaymentInformationResponse"
    }
```

Confirmation of Funds (PIIS)

Die Ausführung der PIIS Aufträge richtet sich an Anbieter von Zahlungsinstrumenten und ermöglicht die Abfrage, ob auf einem Konto der angefragte Betrag verfügbar ist. Beispiel:

```
{
        "Type": "BGCheckAvailabilityOfFundsRequest",
        "TPPData": {
            "ASPSPHost": "{{BGHost}}",
            "ASPSPPath": "{{BGAISPath}}",
            "TransportCertificateId": "{{BGTransportCertificateId}}",
        },
        "SecurityParameters": {
            "XRequestID": "{{$guid}}"
        },
        "PSUParameters": {
            "PSUIPAddress": "{{psuIpAddress}}"
        },
        "ConfirmationOfFunds": {
            "Account": {
               "currency": "EUR",
                "iban": "{{BGIban}}"
            },
            "instructedAmount": {
                "currency": "EUR",
                "amount": {{instructedAmount}}
            }
        }
```

Allgemeine Aufträge

Insbesondere im Bereich der PreAuthentication oder beim Onboarding kann es vorkommen das Requests verlangt werden, die nicht standardisiert sind, aber dennoch z.B. die Verwendung der TPP Zertifikaten erfordern. Hierfür kann ein generischer Execute Request verwendet werden, mit dem quasi beliebige Aufträge ausgeführt werden können.

Beispiel - Abfrage des Consentstatus auf Basis des Execute Requests:

```
{
	"Type": "BGExecuteRequest",
	"TPPData": {
		"ASPSPUrl":
"https://xs2a-api.comdirect.de/berlingroup/v1/consents/{{id}}/status",
		"TransportCertificateId": "qwac"
	},
	"SecurityParameters": {
		"XRequestID": "5e3bbfbc-6e50-4dd2-8734-650ae299a113"
	},
	"ExecuteParameters": {
		"Method" : "GET",
		 "ContentType": "application/json"
	}
}
```

Die Response auf den obigen Request sieht wie folgt aus:

```
{
	"Content": "{\n  \"consentStatus\" : \"received\"\n}",
	"XRequestID": "5e3bbfbc-6e50-4dd2-8734-650ae299a113",
	"Type": "BGExecuteResponse"
}
```

SEPA Orders

SEPA Orders benötigen keine Benutzeranmeldung und können autonom ausgeführt werden.

Beispiele für SEPA Orders sind z.B. Tools / Konverter / Validatoren, die die Arbeit mit SEPA-Dokumenten und deren Integration in bestehende Anwendungsszenarien erleichtern.

SEPA-XML Dokumente nach JSON konvertieren

Konvertiert ein gültiges SEPA XML Dokument in die entsprechende JSON-Notation.

Beispiel:

{

"OrderId": "1",

"Type": "SepaXmlToJsonRequest",

"Xml" : "<base64 encoded SEPA Dokument>"

}

Die Response enthält dann das SEPA Dokument in der JSON Notation.

```
"SepaDocument": {
      "Message": {
        "InitiatingParty": {
            ...
        },
        "PaymentInformations": [
            {
              ...
            }
        ],
        "Type": "...",
        ...
      },
      "MessageInfo": {
        "PainIdentifier": "...",
        ...
      },
      ...
}
```

SEPA-Dokumente im JSON Format nach XML konvertieren

Konvertiert ein SEPA-Dokument im JSON Format in ein XML-Dokument, das z.B. mit weiteren Anwendungen verarbeitet werden kann.

Beispiel:

```
{
      "OrderId": "1",
      "Type": "SepaJsonToXmlRequest",
      "SepaDocument": {
            "Message": {
              "InitiatingParty": {
                  ...
              },
              "PaymentInformations": [
                  {
                    ...
                  }
              ],
              "Type": "...",
              ...
            },
            "MessageInfo": {
              "PainIdentifier": "...",
              ...
            },
            ...
      }
}
```

Die Antwort enhält nun das SEPA XML Dokument im Base64 Format.

```
{
  "Xml": "<base64 encoded SEPA Dokument>",
  "OrderId": "1"
}
```

Info Orders

Info Orders benötigen keine Benutzeranmeldung und können autonom ausgeführt werden.

Beispiele für Info Orders sind inhaltliche Konvertierungen (z.B. Umwandlung nationaler Kontonummern in IBANs), Validierungen (z.B. IBAN Prüfung) oder der Abruf von Bank- und Zugangsinformationen.

Bankinformationen / Zugangsdaten abrufen

Der GetBankInfoRequest liefert alle Für die angegebene Bankleitzahl werden Informationen für den FinTS und EBICS Zugang abgerufen. Ferner ist in der Antwort auch ein

Request:

```
{
   "Type": "GetBankInfoRequest",
   "BankCode": "51390000"
}
```

Die GetBankInfoResponse enthält fünf logische Bereiche:

1. Allgemeine Informationen - Bankleitzahl, BIC, Name, Banklogo und einen Hinweis ob SEPA Echtzeitüberweisungen möglich sind.
2. Informationen zum FinTS Endpoint, z.B. Hinweise für die Benutzeranmeldung und ob eine spezielle FinTS Bankleitzahl verwendet werden muss (UseBLZ)
3. Informationen zum EBICS Endpoint (URL und Hostname)
4. Informationen zum XS2A Endpoint - beinhaltet z.B. Hinweise zur Anmeldung und den unterstützten Geschäftsvorfällen.
5. Hinweise ob die Bank am giropay Verfahren teilnimmt.

Response:

```
{
   "BankCode": "51390000",
   "BIC": "VBMHDE5FXXX",
   "BankName": "VB Mittelhessen",
   "BankNameFull": "Volksbank Mittelhessen",
   "CardImage": "<base64 encoded image>"
   "SCTInstReachable": false,
```

```
"FinTSInfo": {
    "CommAddress": "https://hbci11.fiducia.de/cgi-bin/hbciservlet",
    "FinTSVersion": "300",
    "Pin1": false,
    "Pin2": true,
    "UserID": "Alias/VR-NetKey",
    "Notice": "",
    "TestMode": false,
    "UseBlz": "",
    "FinTSSupported": true,
    "TanConsumeSupported": false
},
"EbicsInfo": {
    "HostId": "EBICSVB",
    "HostUrl": "https://ebics.zv-mittelhessen.de/"
},
"Xs2aInfo": {
    "Contact": {
        "AccountServiceTypes": "Bank",
        "Authentication": "UserPassword",
        "BankCode": "51390000",
        "Capabilities": [
            "ImplementsBalance",
            "ImplementsStatement"
        ],
        "LoginURL": "https://hbci11.fiducia.de/cgi-bin/hbciservlet",
        "ProductName": "Volksbank Mittelhessen FinTS",
        "Profile": {
            "UserIDLabel": "Alias/VR-NetKey",
            "PasswordLabel": "PIN"
        },
        "Xs2aName": "FinTS"
    }
```

```
    },
    "GiropayInfo": {
        "Giropay": true,
        "GiropayId": true
    },
    "Type": "GetBankInfoResponse"
}
```


IBAN Validierung

IBAN Validierung IBANs.

Request:


```
{
      "Type":"ValidateIBANRequest",
      "OrderId" : "3",
      "IBAN": "<IBAN>"
}
```


Response:


```
{
      "Type": "ValidateIBANResponse",
      "OrderId": "3",
      "IBAN": "<IBAN>",
      "CountryCode": "e.g. DE",
      "ValidationCode": 0,
      "ValidationMessage": "OK"
}
```


Validierungscodes:

```
CODE_OK = 0; //OK
CODE_UNSUPPORTED_COUNTRY = 8; //Land nicht unterstützt
CODE_BEST_GUESS = 9; //IBAN Ermittlung bei mehrdeutigen
Kontonummern
CODE_INVALID_IBAN = 1001; //Ungültige IBAN
CODE_UNKNOWN_BANKCODE = 1002; //Ungültige Bankleitzahl
```

Ermittlung von Konto-/Bankinfos - IBAN

IBAN Validierung und Bereitstellung weiterer Kontoinformation für IBANs in folgenden Ländern:

- Deutschland
- Österreich
- Schweiz
- Niederlande

Request:

```
{
      "Type":"GetAccountInfoFromIBANRequest",
      "OrderId" : "2",
      "IBAN": "<IBAN>"
}
```

Response:

```
{
      "Type": "GetAccountInfoFromIBANResponse",
      "OrderId": "2",
      "BIC": "<BIC>",
      "AcctNo": "<nationale Kontonummer>",
      "BankCode": "<Bankleitzahl>",
      "BankName": "<Name der Bank>",
      "IBAN": "<IBAN>",
      "CountryCode": "e.g. DE",
      "ValidationCode": 0,
      "ValidationMessage": "OK"
}
```

Ermittlung von Konto-/Bankinfos - IBAN

IBAN Validierung und Bereitstellung weiterer Kontoinformation unter Angabe der Kontonummer / Bankleitzahl in folgenden Ländern:

- Deutschland - DE
- Österreich - AT
- Schweiz - CH
- Niederlande - NL

Request:

```
{
      "Type":"GetAccountInfoFromNationalAccountRequest",
      "OrderId" : "1",
      "Country": "DE",
      "AcctNo": "<nationale Kontonummer>",
      "BankCode" : "<Bankleitzahl>"
}
```

Response:

```
 {
      "Type": "GetAccountInfoFromNationalAccountResponse",
      "OrderId": "1",
      "BIC": "<BIC>",
      "AcctNo": "<nationale Kontonummer>",
      "BankCode": "<Bankleitzahl>",
      "BankName": "<Name der Bank>",
      "IBAN": "<IBAN>",
      "CountryCode": "e.g. DE",
      "ValidationCode": 0,
      "ValidationMessage": "OK"
}
```

XS2A Zugangsdaten abrufen

Abfrage der allgemeinen Informationen zum XS2A Zugang. Die Abfrage kann wahlweise auf Basis des Account-Objects oder der eindeutigen Xs2aId erfolgen.

Beispiel Request auf Basis der Xs2aId:

```
{
    "Orders": [
         {
            "Type": "GetXs2aContactInfoRequest",
            "Xs2aId":"N26"
        }
    ]
}
```

Beispiel Request auf Basis der Account Angaben:

```
{
    "Orders": [
         {
            "Type": "GetXs2aContactInfoRequest",
              "Account": {
              "AcctTpCd": "CACC",
              "AcctCcy": "EUR",
              "AcctBankCode": "10011001"
            },
        }
    ]
}
```

Beispiel Response:


```
{
   "Contact": {
      "AccountServiceTypes": "Bank",
      "Authentication": "UserPassword",
      "BankCode": "10011001",
      "Capabilities": [
           "ImplementsBalance",
           "ImplementsStatement"
      ],
      "LoginURL": "https://api.tech26.de/",
      "ProductName": "N26",
      "Profile": {
         "UserIDLabel": "E-Mail",
         "PasswordLabel": "Passwort"
      },
      "Xs2aName": "N26 API"
   },
   "Type": "GetXs2aContactInfoResponse"
}
```


Aus der Response lassen sich wichtige Informationen zum angefragten XS2A Zugang entnehmen, wie z.B. die Authentication Methode, welche Geschäftsvorfälle unterstützt werden und Hinweise für die Benutzeranmeldung.

Bank- und Kartenlogos abrufen

Abrufen eines Base64-kodierten Images für die angegebene Bank oder Kreditkarte. Der anzugebende Imagename entspricht dem Dateinamen der Grafik.

Beispiel Request:

```
{
      "Orders": [
      {
            "Type":"GetCardImageRequest",
            "OrderId" : "1",
            "ImageName": "N26"
      }]
}
```

Beispiel Response:

```
{
      "CardImage": "<base64encode image>",
      "Type": "GetCardImageResponse",
      "OrderId": "1"
}
```

Ermittlung der unterstützten EBICS Versionen

Anonymer Request zur Ermittlung der vom EBICS Zielsystem unterstützten EBICS Versionen. Dieser Request kann vorab verwendet werden um zu erfahren, welche EBICS-Versionen das gewünschte EBICS Zielsystem unterstützt.

Nebenbei wird sichergestellt, dass unter der angegebenen Host-ID und Host-URL auch ein EBICS Zielsystem erreichbar ist.

Beispiel Request:

```
{
      "Orders": [
          {
            "Type": "GetSupportedEbicsVersionsInfoRequest",
            "HostID": "APO",
            "HostUrl": "https://multicash.apobank.de/"
         }
     ]
}
```

Beispiel Response:

```
{
      "SystemReturnCode": {
            "ReportText": "OK",
            "ReturnCode": "000000",
            "SymbolicName": "EBICS_OK"
      },
      "Versions": [
            {
                  "VersionNumber": "02.30",
                  "ProtocollVersion": "H002"
            },
              {
                  "VersionNumber": "02.40",
                  "ProtocollVersion": "H003"
            },
```

```
            {
                  "VersionNumber": "02.50",
                  "ProtocollVersion": "H004"
            }
      ],
      "Type": "GetSupportedEbicsVersionsInfoResponse"
}
```

Erstellung von EPC-069 Codes

Der BankAccessServer unterstützt die Generierung von EPC-069 kompatiblen QR-Codes. Der QR-Code enthält dabei alle relevanten Daten für eine SEPA-Überweisung und kann in einer Vielzahl von Banking Apps (z.B. Banking 4I/4A) direkt eingelesen werden.

Für die Generierung der EPC Codes können folgende Felder spezifiziert werden:

Feldname	**Typ**		**Beschreibung**
IBAN	string	M	IBAN des Empfängers
BIC	string	O	BIC der Empfängerbank
Receiver	string	M	Name des Empfängers / Kontoinhabers
Amount	Number	O	Betrag
Purpose	string	O	Geschäftscode
Reference	string	C	Zahlungsreferenz, darf nicht zusammen mit dem Text/Verwendungszweck angegeben werden
Text	string	C	Verwendungszweck, darf nicht zusammen mit der Reference/Zahlungsreferenz angegeben werden
Display	string	O	Benutzerhinweis

Neben den inhaltlichen Daten der SEPA-Überweisung kann auch die Größe des QR-Codes in px über das Feld “imageSize” spezifiziert werden.

Beispiel Request für die Generierung eines EPC-Codes (200px):

```
{
      "Orders": [
          {
            "Type":"CreateEpcCodeRequest",
                 "OrderId" : "1",
                 "data" : {
                      "IBAN" : "XXXX",
                       "Receiver" : "Klaus Igel",
                      "Amount" : 102.50,
                       "Reference": "Test"
                 },
             "imageSize" : 200
          }
     ]
}
```

Beispiel Response:

```
{
      "EpcCode": "<base64encode image>",
      "Type": "CreateEpcCodeResponse",
      "OrderId": "1"
}
```

Push Benachrichtigungen

Mit der Version BankAccessServer Version 3.5 haben wir die Möglichkeit eingeführt, Nachrichten über einen zwischengeschalteten PushServer an registrierte Apps zu versenden.

Als erste praktische Lösung haben wir eine Kopplung mit der Subsembly EBICS VEU App umgesetzt. In der App können Benachrichtigungen für hinterlegte Konten aktiviert werden, sofern hier Aufträge zur Freigabe vorliegen. Der BankAccessServer bietet nun die Möglichkeit für die mit einer Transportberechtigung eingereichten Zahlungen die passenden VEU Apps zu benachrichtigen.

InfoPingAcctsRequest

Feldname	**Typ**	**Beschreibung**
PingTokenId	string	Über die Token Id wird aus den appsettings das zu verwendende JWT Token ermittelt.
ProductId	string	Produktkennung
ProductVer	string	Produktversion
PingAccts	Array InfoPingAcct	Array von InfoPingAcct Objekten, bestehend aus dem EventType und Iban

Beispiel:

```
"Orders": [
      {
            "OrderId": "{{$randomInt}}",
            "Type": "InfoPingAcctsRequest",
            "PingTokenId" : "EbicsVeuToken",
            "ProductId": "BAS",
            "ProductVer": "3.5",
            "PingAccts": [
                  {
                        "iban" : "xxx"
                  }
            ]
      }
]
```

Beispiel Response:


```
{
    "Responses": [
        {
            "HttpStatus": 200,
            "HttpSuccess": true,
            "Type": "InfoSupaErrorResponse",
            "OrderId": "950"
        }
    ]
}
```

Swagger basierte API

Die API Beschreibung liegt komplett im JSON-Format vor. Auf Grundlage des mitgelieferten Swagger-Files sind Entwickler in der Lage, eine Codegenerierung anhand der API Spezifikation vorzunehmen. Ein entsprechender Client kann somit für die verschiedensten Sprachen in kürzester Zeit generiert werden, so z.B. für Java, PHP, Node.js, Javascript oder C#.

Die komplette Doku kann u.A. mit dem Swagger Editor gelesen werden. Den Editor, Code Generator und weitere Informationen zu Swagger finden Sie unter http://swagger.io

Konzepte / Best Practices

FinTS

Bankdaten und FinTS Support ermitteln

Der Subsembly BankAccessServer führt intern eine Tabelle mit Informationen zu den erforderlichen Zugangsdaten für alle Banken in Deutschland. Die Zugangsdaten können über die Info Order “GetBankInfoRequest” unter Angabe der Bankleitzahl ermittelt werden. Fragt man z.B. die Zugangsdaten für die Bankleitzahl 12030000 ab, so erhält man folgende Informationen:

Request

```
{
    "RequestOptions": {
        "RequestId": "{{$guid}}",
        "AccessToken": "{{AccessToken}}"
    },
    "Orders": [
        {
            "OrderId": "{{$randomInt}}",
            "Type": "GetBankInfoRequest",
            "BankCode": "12030000"
        }
    ]
}
```

Response

```
{
      "BankCode": "12030000",
      "BIC": "BYLADEM1001",
      "BankName": "Deutsche Kreditbank Berlin",
      "BankNameFull": "Deutsche Kreditbank Berlin",
      "CardImage": "<base64encode image>",
      "FinTSInfo": {
            "CommAddress": "https://banking-dkb.s-fints-pt-dkb.de/fints30",
            "FinTSVersion": "300",
            "Pin1": false,
            "Pin2": true,
            "UserID": "Anmeldename",
```

```
            "Notice": "Bitte verwenden Sie nur den Anmeldenamen und nicht
            Ihre Legitimations-ID! Für pushTAN ist ein spezieller
            Anmeldename erforderlich.",
            "TestMode": false
      },
      "Xs2aInfo": {
            "Contact": {
            "AccountServiceTypes": "Bank",
                  "Authentication": "UserPassword",
                  "BankCode": "12030000",
                  "Capabilities": [
                  "ImplementsBalance",
                        "ImplementsStatement"
                  ],
                  "LoginURL":
      "https://banking-dkb.s-fints-pt-dkb.de/fints30",
                  "ProductName": "Deutsche Kreditbank Berlin FinTS",
                  "Profile": {
                  "UserIDLabel": "Anmeldename",
                        "PasswordLabel": "PIN",
                        "AdviceText": "Bitte verwenden Sie nur den
      Anmeldenamen und nicht Ihre Legitimations-ID! Für pushTAN ist ein
      spezieller Anmeldename erforderlich."
                  },
                  "Xs2aName": "FinTS"
      }
      },
      "EbicsInfo": {
            "HostId": "D88-ELKO",
            "HostUrl":
            "https://by8ebics.sparkasse-banking.de/ebicsweb/ebicsweb"
      },
      "Type": "GetBankInfoResponse",
      "OrderId": "1"
}
```


Um nachfolgende Dialoge mit dem Kunden individuell und passend darzustellen sind insbesondere auch die Felder “benutzer” und “hinweis” interessant.

FinTS Verbindungsaufbau

Anmeldung über Benutzername/PIN, ohne dass das Sicherungsverfahren bekannt ist.

```
"Connection": {
        "LogOn": {
                "Contact": {
                        "BankCode": "{{BankCode}}",
                        "UserID": "{{UserID}}"
                },
                "Pin": "{{Pin}}"
        }
        "Suspend": true
}
```

Sofern der Benutzer mehr als ein TAN Verfahren besitzt, muss dieses nachfolgend spezifiziert werden, um eine eindeutige Auswahl zu treffen. Die aktuelle Session wird für folgende Aufrufe weiterverwendet und über den Service dargestellt. Alle weiteren Aufrufe müssen wiederum die zuvor gesendeten Orders beinhalten.

Die Anmeldung enthält in diesem Fall in der Response das Feld **NeedSecurityFunction**, das wiederum alle Sicherheitsverfahren des Kunden enthält.

Der Anwender kann nunmehr die entsprechende Security Function auswählen und den Request wie folgt absenden:

```
"Connection": {
         "SetSecurityFunction": {
                 "Service": "{{Service}}",
                "SecurityFunction": {{SelectedSecurityFunction}}
         },
        "Suspend": true
}
```

Die Auswahl der Security Function erfolgt über den dreistelligen numerischen Wert, z.B. 901.

Weiterhin kann es notwendig sein ein TAN Medium zu spezifizieren. Das kann z.B. notwendig sein wenn der Kunde ein SMS oder App TAN Verfahren auf mehreren Endgeräten nutzt. In diesem Fall enthält die Antwort auf die Anmeldung das Feld **NeedTanMediaName**, das die verfügbaren TAN-Medien beinhaltet.

Das zu verwendende TAN Medium wird wie folgt festgelegt und muss wiederum die zuvor gesendeten Orders beinhalten.

```
"Connection": {
         "SetTanMediaName": {
                 "Service": "{{Service}}",
                "TanMediaName": {{SelectedTanMediaName}}
         },
        "Suspend": true
}
```

Sofern im Rahmen der Anmeldung eine TAN notwendig ist, enthält die Antwort das Feld **NeedTan** mit der zugehörigen TAN-Challenge.

```
"Challenge": "Die mobileTAN zu diesem Auftrag wird als SMS an Ihre
Mobil-Telefonnummer gesendet, die bei Ihrer Bank registriert ist.",
```

Eine TAN kann wie folgt übertragen werden und muss wiederum die zuvor gesendeten Orders beinhalten.:

```
"Connection": {
        "SendTan": {
                "Service": "{{Service}}",
                "Tan": {{TAN}}
        },
        "Suspend": true
}
```

Wie dargestellt, kann eine SCA basierte Anmeldung im Worst Case aus vier Schritten bestehen. Somit sollte bei wiederkehrenden Benutzer Anmeldungen immer mit einem vollständigen Kontakt vorgenommen werden.

Beispiel:

```
"Connection": {
        "LogOn": {
                "ContactSerialized": "{{SerializedContact}}",
                    "Pin": "{{Pin}}"
                },
        "Suspend": true
}
```

Der serialisierte FinTS Kontakt kann in der Response zurückgegeben werden, in dem in den Response Options der Wert Contact auf true gesetzt wird.

```
"ResponseOptions": {
        "Trace": false,
        "Contact": true
}
```

Bei Instituten, die nicht bei jedem Login eine SCA verlangen, können über diesen Weg unnötige Schritte vermieden werden.

Nach einer Anmeldung ist im Verlauf der Session immer sinnvoll, über den Service (entspricht dem aktiven FinTS Dialog) zu gehen und diesen am Ende der Session sauber zu beenden. Der Service wird immer dann zurückgeliefert, wenn in der Connection **Suspend=true** angegeben wird.

```
"Connection": {
        "Resume": {
                "Service": "{{Service}}"
        },
        "Suspend": true
},
```

Der FinTS Dialog wird beendet, indem Suspend auf false gesetzt wird.

Weiterhin ist zu beachten, dass alle FinTS Anfragen mit einer gültigen Produktkennung / Version vorgenommen werden.

Hintergrund: Zur Erfüllung der PSD2-Vorgaben bzgl. der Transparenz über die kundenseitig eingesetzte Software hat die Deutsche Kreditwirtschaft einen **Prozess zur Registrierung von FinTS-Produkten** etabliert, um gegenüber den Kunden Informationen zur FinTS-Nutzung nachweisen zu können. Ab dem **01.12.2018 sollen** zugeteilte FinTS-Registrierungsnummern in der Dialoginitialisierung im Element Produktbezeichnung gesendet werden. **Vor dem 1.12.2018 ist die Verwendung nicht erlaubt**.

Weitere Informationen zur FinTS Produktregistrierung finden Sie auf:
https://www.hbci-zka.de/register/register_faq.htm

Hierzu kann, sofern der BankAccessServer immer mit derselben Kennung arbeitet, der in den appsettings konfigurierte Wert verwendet werden.

```
"BasConfig": {
        ...
        "ProductName": "",
        "ProductVersion": ""
}
```

Sofern dynamische Produktkennungen verwendet werden sollen, können diese innerhalb der Connection -> LogOn spezifiziert werden.

Spezielle Testeinstellungen für die starke Kundenauthentifizierung:

Innerhalb des Connection->LogOn->Contact-StrongCustomerAuthentication kann das SCA Verhalten definiert werden. Folgende Werte sind zulässig:

- Default - keine SCA bis 14.09.2019, danach mit SCA
- RequestSCA - SCA aktiviert (hilfreich um im Vorfeld gegen Server mit SCA testen zu können)
- NoSCA - ggfs. sinnvoll wenn nach dem 14.09.2019 Server noch kein SCA aktiviert haben

```
"Connection": {
        "LogOn": {
            "Contact": {
                "BankCode": "{{BankCode}}",
                "UserID": "{{UserID}}",
                "StrongCustomerAuthentication":"RequestSCA"
            },
            "Pin": "{{Pin}}"
        },
        "Suspend": true
    }
```

Verwendung von FinTS-Session Tokens

Vielfach werden nach einem erfolgreichen Verbindungsaufbau innerhalb einer Session weitere Aktionen durchgeführt, z.B.

- Auslesen der Bankkonten
- Abfragen von Kontosalden
- Umsatzabfragen
- Ausführen von Zahlungen

Es wäre ziemlich ineffizient wenn hierfür jeweils separate Anmeldungen erfolgen würden. Stattdessen kann ein sogenanntes Session Token verwendet werden und die laufende Dialog-Session offen gehalten werden. Die Verwendung bringt einen erheblichen Performancegewinn. Verwendungsbeispiel:

1) Session-Token zurückliefern

```
"Connection": {
      "LogOn": {
            ...
      },
      "Suspend" : true
}
```

2) Session-Token für weitere Anfragen verwenden

```
"Connection": {
      "Resume": {
            "Service": "<encoded/encrypted session>"
      }
}
```

Unterstützte Geschäftsvorfälle

Je nach Kreditinstitut und Bankkonten des Kunden sind bestimmte Geschäftsvorfälle verfügbar bzw. erlauben unterschiedliche Parameter. So ist z.B. für ein Sparkonto keine SEPA-Überweisung oder der Umsatzabruf bei Bank XY für 360 Tage rückwirkend möglich. Im Rahmen des FinTSContactInfoRequests kann man alle über den BankAccessServer verfügbaren Geschäftsvorfälle und deren Parametrisierung erhalten. Somit kann eine effektive Ablaufsteuerung gut und einfach umgesetzt werden.

Ausschnitt aus einer FinTSContactInfoResponse:

```
"FinOrderBuilderProperties": [
{
      "Type": "FinAcctBalBuilderProperties",
      "TanRequired": false,
      ...
},
{
      "Type": "FinAcctMvmtsSpecifiedPeriodBuilderProperties",
      "AcctMvmtDataCutoff": 360,
      "AllAcctAllowed": false,
      "MaxEntriesAllowed": true,
      "TanRequired": false
      ...
},
{
      "Type": "FinSepaSingRemittBuilderProperties",
      "TanRequired": true,
      ...
}
```

Interessant ist in diesem Zusammenhang auch der TanRequired Wert über den ersichtlich ist, ob der Auftrag mit einer TAN freigegeben werden muss.

FinTS Sicherheitsverfahren (TAN-Verfahren)

Grundsätzliches

Der FinTS Standard definiert verschiedene Sicherheitsverfahren zur Absicherung der Kommunikation zwischen Kunden- und Banksystemen. Der BankAccessServer unterstützt ausschließlich die Sicherheitsverfahren, die auf TAN-Nummern basieren. Hingegen können Sicherheitsverfahren, die auf Chipkarten oder Schlüsseldateien basieren, nicht verwendet werden.

Abfrage der verfügbaren TAN-Verfahren

Bei einer erfolgreiche Anmeldung im Rahmen eines FinTSContactInfoRequests erhält man eine Liste der verfügbaren TAN-Verfahren. Für die weitere Verwendung / Auswahl des TAN-Verfahrens ist die sogenannte TechnicalIdentification zu verwenden, die das Verfahren eindeutig identifiziert.

```
"FinTanProcInfos": [
      {
            "Name": "chipTAN manuell",
            "SecurityFunction": 910,
            "TanMediaNameReqd": 2,
            "TechnicalIdentification": "HHD1.3.0"
      },
      {
            "Name": "chipTAN optisch",
            "SecurityFunction": 911,
            "TanMediaNameReqd": 2,
            "TechnicalIdentification": "HHD1.3.0OPT"
      }
]
```

Auswahl des TAN-Verfahrens

Das gewünschte TAN-Verfahren kann im Contact-Objekt über die TanProcedure eingestellt werden. Der zu verwendende Wert ist die SecurityFunction des TAN-Verfahrens.

```
"Connection": {
      "LogOn": {
            "Contact": {
                  "BankCode": "12030000",
                  "UserID": "xxxxx",
                  "SecurityFunction" : 901
            },
            "Pin": "XXXXX"
      }
}
```

Hinweis: Erfolgt keine Angabe der TanProcedure, so wird das erste zurückgelieferte TAN-Verfahren für die Freigabe von TAN-pflichtigen Aufträgen verwendet. Lässt man sich den Contact in der Response zurückliefern, so enthält dieser bereits das ausgewählte Verfahren, so dass keine explizite Angabe mehr notwendig ist.

Bezeichnung des TAN-Mediums

Innerhalb eines TAN-Verfahrens können wiederum mehrere TAN-Medien verwendet werden. So können z.B. beim SMS-TAN Verfahren mehrere Telefonnummern oder beim Chip-TAN Verfahren mehrere Chipkarten hinterlegt sein.

Im Normalfall wird der Kunde nur ein TAN-Medium verwenden, das automatisch verwendet wird wenn keine explizite Angabe erfolgt. Die verfügbaren TAN-Medien können mit folgender Order abgefragt werden:

```
"Orders": [
      {
            "Type": "FinTanMediaListRequest"
      }
],
```

Die Response enthält dann die verfügbaren TAN-Medien:

```
"TanMedias": [
      {
            "AvailableTANCount": 0,
            "CardNumber": "519008XXXX",
            "CardType": 0,
            "SecurityFunction": 0,
            "TanMediaClass": "A",
            "TanMediaName": "Girokarte1",
            "TanMediaStatus": "Active"
      },
      {
            "AvailableTANCount": 0,
            "CardNumber": "519009XXXX",
            "CardType": 0,
            "SecurityFunction": 0,
            "TanMediaClass": "A",
            "TanMediaName": "Girokarte2",
            "TanMediaStatus": "Available"
      }
]
```

Die Auswahl des TAN-Mediums erfolgt durch die Angabe im Contact:

```
"Connection": {
      "LogOn": {
            "Contact": {
                  "BankCode": "12030000",
                  "UserID": "xxxxx",
                  "TanMediaName" : "Girokarte2"
            },
            "Pin": "XXXXX"
      }
}
```

Alternativ kann die Festlegung auch im Workflow selbst durch den FinTSSetTanMediaNameRequest erfolgen.

Freigabe von TAN-pflichtigen Aufträgen

Nach der Einreichung eines TAN-pflichtigen Auftrags muss dieser mittels einer TAN-Nummer freigegeben werden und wird durch den Response Type FinTSNeedTanResponse quittiert.


```
"Order": "orderToken",
"ChallengeInfo": {
      "ChallengeHHDUC": "28088815172910103526958604 1,00",
},
"TanProcess": {
      "ActiveTanListCount": 1,
      "ActiveTanMediaCount": 1,
      "CancelOrderAllowed": false,
      "ChallengeAmountReqd": false,
      "ChallengeClassReqd": false,
      "ChallengeLabel": "TAN-Nummer",
      "ChallengeStructd": false,
      "MaxChallengeLength": 3,
      "MaxTanLength": 6,
      "MultipleTanOrdersAllowed": false,
      "MultiTanAllowed": true,
      "Name": "chipTAN optisch",
      "OrderingAcctReqd": false,
      "PostponedTanAllowed": true,
      "SecurityFunction": 911,
      "SegmentVersion": 3,
      "SmsChargingAcctReqd": false,
      "TanFormat": 1,
      "TanListNoReqd": false,
      "TanMediaInitMode": "00",
      "TanMediaNameReqd": 2,
      "TanProcess": 2,
      "TechnicalIdentification": "HHD1.3.0OPT"
},
"Type": "FinTSNeedTanResponse"
```

Sofern noch die Auswahl eines TAN-Mediums erforderlich ist wird der Auftrag zuvor mit einer FinTSNeedTanMediaNameResponse und im Anschluss mit der FinTSNeedTanResponse quittiert. Die Festlegung des TAN Mediums erfolgt mit einem FinTSSetTanMediaNameRequest.

Der nachfolgende FinTSSendTanRequest gibt dann den Auftrag frei. Hierfür sind neben der TAN-Nummer auch das Service- und Order Token anzugeben, damit die Zuordnung zum eigentlichen Auftrag erfolgen kann.

```
{
    "Connection": {
        "Resume": {
            "Service": "<serviceToken>"
        }
    },
    "Orders": [
        {
            "Type": "FinTSSendTanRequest",
            "Tan": "123456",
            "Order": "<orderToken>"
        }
    ],
    "ResponseOptions": {
        "Trace": false,
        "Contact": false
    }
}
```

Die Antwort enthält im Positivfall dann auch den ursprünglichen Auftragstyp und weitere Details. Im Falle einer SEPA-Einzelüberweisung z.B.:

```
{
    "Responses": [
        {
            "Type": "FinSepaSingRemittResponse",
           ...
        }
    ]
}
```

Allgemeine Informationen

Mehrere Aufträge ausführen

Zur Vermeidung unnötiger Server Roundtrips und optimaler Performance gibt es die Möglichkeit mehrere Orders in einem Request zu schicken. Will man z.B. für bekannte Konten in einem Rutsch Salden und Umsätze abrufen, könnte man folgenden Request schicken:

```
"Orders": [
      {
            "OrderId": "1",
            "Type": "FinAcctBalRequest",
            "OrderAcct": {
                  ...
            }
      },
      {
            "Type": "FinAcctBalRequest",
            "OrderId": "2",
            ...
      },
      {
            "Type": "FinCamtStatementRequest",
            "OrderId": "3",
            ...,
            "StartDate": "2017-04-29",
            "EndDate": "2017-04-29"
      },
      {
            "Type": "FinCamtStatementRequest",
            "OrderId": "4",
            ...,
            "StartDate": "2017-04-29",
            "EndDate": "2017-04-29"
      }
]
```

Wichtig/empfehlenswert ist die Definition eindeutiger OrderIds, so dass die Response einfacher verarbeitet bzw. zugeordnet werden kann.

```
"Responses": [
      {
            ...
            "Type": "FinAcctBalResponse",
            "OrderId": "1"
      },
      {
            ...
            "Type": "FinAcctBalResponse",
            "OrderId": "2"
      },
      {
            ...
            "Type": "FinCamtStatementResponse",
            "OrderId": "3"
      },
      {
            ...
            "Type": "FinCamtStatementResponse",
            "OrderId": "4"
      }
]
```

Verarbeitung von SEPA-Dokumenten

Der BankAccessServer erlaubt die Verarbeitung unterschiedlichster SEPA-Aufträge, z.B. für Einzel-/Sammelüberweisungen, Lastschrifteinzüge.

Jedes SEPA-Dokument kann in zwei unterschiedlichen Formaten verarbeitet werden:

1. JSON-Repräsentation
2. Base64 encoded

Aufgrund der Verwendung des einheitlichen SEPA-Standards ergeben sich weitreichende Integrationsmöglichkeiten in bestehende Anwendungen, ohne auf proprietäre Formate setzen zu müssen.

JSON-Repräsentation des SEPA Dokuments

Das SEPA-Dokument wird in diesem Fall als JSON Objekt dargestellt und is am besten für die serverseitige AdHoc Generierung geeignet. Das SEPA Dokument für eine Einzelüberweisung würde dabei wie folgt aussehen:

```
"SepaDocument": {
	"Message": {
		"Type": "SepaCreditTransferPaymentInitiationRS",
		"InitiatingParty": {
			"Name": "KLAUS IGEL"
		},
	"PaymentInformations": [
	{
		"Type": "SepaCreditTransferPaymentInformationRS",
		"Debtor": {
			"Name": "KLAUS IGEL"
		},
		"DebtorAccountIBAN": "DEXXXXXXXXXX0000XXXXXX",
		"TransactionInformations": [
		{
			"Type": "SepaCreditTransferTransactionInformationRS",
			"Amount": "1",
			"RemittanceInformation": "BankAccessServer",
			"Creditor": {
				"Name": "KLAUS IGEL"
			},
			"CreditorAccountIBAN": "DEXXXXXXXXXX0000XXXXXX",...
		} ]
	} ]
}
```

Base64 encoded SEPA Dokument

Für den Fall, dass die Erzeugung von SEPA-Dokumenten bereits durch andere Anwendungen erfolgt kann die Verarbeitung wie folgt vorgenommen werden:

1. Erzeugen des SEPA XML-Dokuments
2. XML Konvertierung in Base64
3. Übergabe des Base64 kodierten Dokuments

Auch hierzu ein kurzes Besipiel anhand der SEPA-Einzelüberweisung.

```
"Orders": [
      {
            "Type": "FinSepaSingRemittRequest",
            "OrderAcct": {
                  "AcctNo": "kontonummer",
                  "IBAN": "IBAN",
                  "BIC": "BIC"
            },
            "SepaDocumentSerialized": "<base64enc>"
      }
]
```

EBICS

Sicherheit

Um zu gewährleisten, dass nur befugte Anwendungen und EBICS Teilnehmer auf die Dienste des Subsembly BankAccessServer EBICS zugreifen können, gibt es verschiedene Sicherheitsvorkehrungen.

Client Berechtigungen

Für die Ausführung der einzelnen Aufträge der BAS EBICS REST-Schnittstelle muss der Client (Mandant) bestimmte zusätzliche Berechtigungen in seinem Access-Token mitbringen. Die BAS EBICS REST-Schnittstelle definiert die folgenden drei unabhängigen Berechtigungen.

Berechtigung	**Beschreibung**
Admin	Berechtigung ist zur Ausführung von administrativen EBICS Orders
Download	Berechtigung zur Ausführung aller EBICS Download Orders, z.B. Umsatzabrufe
Upload	Berechtigung zur Ausführung aller EBICS Upload Orders, z.B. Einreichung von SEPA Zahlungsdateien

Für einen Mandanten können mehrere Client-Access-Tokens mit zwar unterschiedlichen Berechtigungen, aber der gleichen Client-ID generiert werden. Somit ist es z.B. möglich nur einen speziellen Arbeitsplatz eines Mandanten mit einem Client-Access-Token mit Admin Berechtigung auszustatten.

KeyPassword

Die Kommunikation zwischen EBICS Teilnehmer und EBICS Banksystem wird durch verschiedene RSA-Schlüssel gesichert. Es muss gewährleistet sein, dass die geheimen RSA-Schlüssel des EBICS-Teilnehmers zuverlässig geschützt sind. Dies wird erreicht indem die EBICS Teilnehmerschlüssel ausschließlich verschlüsselt gespeichert werden.

Die Verschlüsselung der EBICS Teilnehmerschlüssel basiert auf einem Schlüsselpasswort, das beim Erstellen von neuen EBICS Teilnehmerschlüsseln vom EBICS Teilnehmer selbst vergeben werden sollte. Es ist darauf zu achten, dass hier nur sehr starke Passwörter vom Teilnehmer akzeptiert werden. Ein schwach gewähltes Passwort kann keine ausreichende Sicherheit gewährleisten.

Das Schlüsselpasswort muss für die Ausführung von Requests eingestellt werden (Connection->KeyPassword).

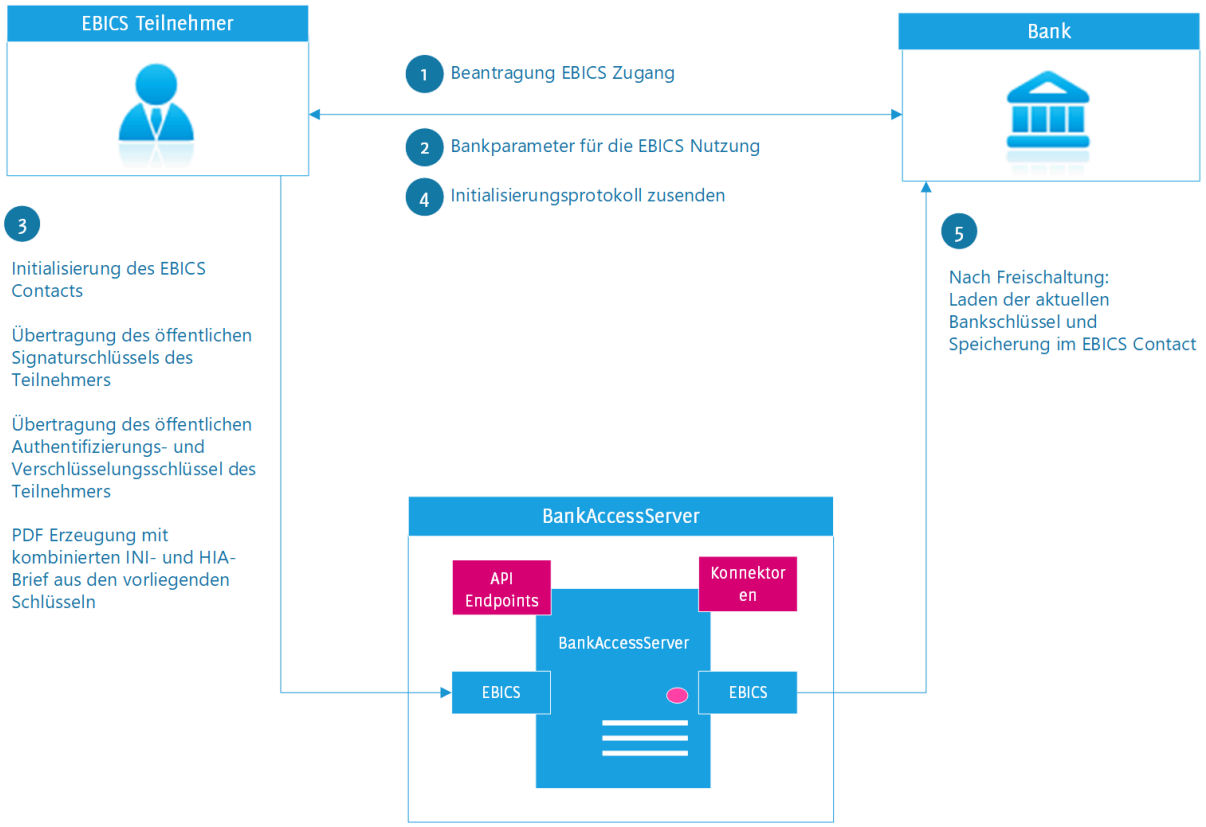
Der BankAccessServer entschlüsselt die RSA-Schlüssel des Teilnehmers immer nur unmittelbar vor deren Verwendung. Die entschlüsselten RSA-Schlüssel werden nicht gespeichert und werden nach jedem Request wieder aus dem Speicher gelöscht.

Für jeden Zugriff auf die EBICS Teilnehmerschlüssel wird deshalb das Schlüsselpasswort des Teilnehmers zur Entschlüsselung erneut benötigt und muss in jedem Request vom Client an den BankAccessServer mitgegeben werden.

EBICS Zugänge einrichten

Nach Beantragung des EBICS Zugangs erhält der Kunde die Zugriffsdaten/Bankparameter von seiner Bank/Sparkasse. Um EBICS Transaktionen durchzuführen muss noch eine Initialisierung durchgeführt werden, bei der das Initialisierungsprotokoll innerhalb einer bestimmten Frist an das Kreditinstitut übermittelt und von dem jeweiligen Kreditinstitut freigegeben werden muss. Der Initialisierungsflow wird vollständig vom BankAccessServer unterstützt.

Exemplarischer Ablauf für die Einrichtung eines EBICS Zugangs:



Weitere Details sind unter dem Abschnitt EBICS Admin Requests beschrieben.

Subsembly EBICS Testserver

Bei Interesse können Sie im Rahmen der Entwicklung gerne auf unseren Subsembly EBICS Testserver zugreifen. Der Subsembly EBICS Testserver kann ohne Kreditinstitut mit einem Dummy-Backend genutzt werden.

XS2A

Verwendung von Sessions

Sofern für einen XS2A Zugang weitere Orders durchgeführt werden sollen, die auf vorherigen Antworten basieren, ist die Verwendung von Sessions sinnvoll.

Ein typisches Beispiel ist die Abfrage einer Kontenliste und im Folgeschritt z.B. der Saldenabruf für jedes Konto.

Hierzu ist das Suspend-Flag in der Connection auf true zu setzen, wodurch die Session aktiv bleibt und in der Response an den Aufrufer zurückgeliefert wird.

Beispiel:

```
{
  "RequestOptions": {
      "RequestId" : "...",
      "AccessToken" : "..."
  },

  "Connection": {
      "Service": {
            "Account" : {
                  ...
            }
      },
    "Credentials": {
      "UserID": "...",
      "Password": "..."
    },
    "Suspend" : true
  },
```

```
  "Orders": [
      {
            "Type": "Xs2aContactInfoRequest"
      }
  ]
}
```

Die Response enthält dann Informationen zur Session, die im weiteren Verlauf verwendet werden kann.

```
"Session": "e5755fba-2847-4787-a497-ff2895560557"
```

Die Folge-Request kann dann wie folgt initiiert werden:

```
{
  "RequestOptions": {
      "RequestId" : "...",
      "AccessToken" : "..."
  },
  "Connection": {
      "Resume": {
           "Session": "e5755fba-2847-4787-a497-ff2895560557"
      },
      "Suspend" : true
  },
  "Orders": [
       ...
  ]
}
```

Das Format der zurückgelieferten Session kann in den App-Settings des BankAccessServers über folgendes Flag gesteuert werden:

RespondWithSessionObjectWhenSuspend - Sofern mit Sessions gearbeitet wird kann festgelegt werden ob das gesamte Session Objekt (true) oder nur die Session Id (false) in der Response zurückgeliefert wird.

Beim letzten Request für eine XS2A Verbindung ist es empfehlenswert, das Suspend-Flag mit false anzugeben, da hierdurch die Session freigegeben und eine explizite Abmeldung beim XS2A Service vorgenommen wird.

2 Faktor Anmeldungen

Im Rahmen der PSD2 und zur Erhöhung der Sicherheit bei Zugriffen auf Kontodaten ist ein weiterer Faktor (z.B. zusätzlich zu Username/Password) notwendig. Der XS2A Endpoint ist auf 2 Faktor Anmeldungen vorbereitet. Dabei wird in der Antwort auf die initiale Anmeldung eine Challenge zurückgeliefert.

Beispiel Request / Connection:

```
"Connection": {
    "Service": {
        "Account": {
          "AcctTpCd": "CACC",
          "AcctCcy": "EUR",
          "AcctNo": "9999999999",
          "AcctBankCode": "99000354",
          "AcctCtry": "DE"
        }

    },
    "Credentials" : {
      "UserID": "subsembly",
      "Password": "654321"
    }
}
```

Beispiel-Response:

```
{
      "ChallengeType": "Question",
      "ChallengeData": "TmFtZSBkZXMgU29mdHdhcmVoZXJzdGVsbGVycw==",
      "Session": "2588937a-ad82-4ada-afa4-672a7a4d9af9"
}
```

Die Challenge Daten sind dabei BASE64 encoded.

Im nächsten Schritt sind dann die Challenge Response Daten anzugeben um effektiv auf die Daten zugreifen zu können:

```
"Connection": {
```

```
    "Resume": {
        "Session": "2588937a-ad82-4ada-afa4-672a7a4d9af9"
    },
    "Credentials" : {
      "ChallengeResponse" : "subsembly"
    }
}
```

Testumgebung

Postman Collection / Environment

Für einen schnellen Einstieg und komfortablen Test des BankAccessServer stellen wir für Postman (https://www.getpostman.com/) ein entsprechendes Testenvironment und eine Collection mit Beispielrequests für die Endpoints FinTS/XS2A und Info bereit.

Hierzu sind die folgenden JSON Dateien im Postman zu importieren:

1.) Testenvironment
2.) Collection

Environment-Variablen

Variable	**Beschreibung**
URL	Base URL des BankAccessServers, z.B. http://localhost:5000
AccessToken	Zugriffstoken für den BankAccessServer
BankCode	Bankleitzahl für FinTS Requests
UserID	FinTS Anmeldename
Pin	FinTS PIN
Acct	Konto für FinTS Salden-/ CAMT Umsatzabfragen, z.B. { "AcctNo": "2000", "BankCode": "70199903","BIC": "SUBSDE03", "CountryCode": "280","Currency": "EUR", "HolderName": "Subsembly GmbH", "IBAN": "DE56701999030000002000" }
EPCData	Daten zur Generierung eines EPC069 QR-Codes einer SEPA-Einzelüberweisung {"IBAN" : "NL64BUNQ2288888888","Receiver" : "TEST","Amount" : 0.07,"Reference": "EPC Test"}
StartDate	Datum ab dem Umsätze abgerufen werden sollen, 2019-05-06
EndDate	Datum bis zu dem Umsätze abgerufen werden sollen, 2019-05-06
Xs2aIdCACC	Id zur Ermittlung eines XS2A Services für Bankkonten, z.B. n26
Xs2aIdCRDC	Id zur Ermittlung eines XS2A Services für Kreditkarten, z.B. LbbCreditCard
Xs2aUserIDCACC	Anmeldename für den XS2A Service (Bankkonten)
Xs2aPaswordCACC	Passwort für den XS2A Service (Bankkonten)

Xs2aUserIDCRDC	Anmeldename für den XS2A Service (Kreditkarten)
Xs2aPaswordCRDC	Passwort für den XS2A Service (Kreditkarten)
SepaDocBase64	Base64 encoded SEPA XML Dokument
FinTSKKAcct	Konto für FinTS Kreditkartenabfragen. z.B.: {"AcctNo": "XXXXXXXXXXXX2922", "BankCode": "XXXXXXXX", "CountryCode": "280", "Currency": "EUR", "HolderName": "Klaus Igel", "SubAcctCharacteristic": "Visa-Karte (Kreditkarte)"}
FinTSKKNo	Kreditkartennummer für FinTS Kreditkartenabfragen
FinTSMvmtsAcct	Konto für FinTS MT940 Umsatzabfragen, z.B. { "AcctNo": "2000", "BankCode": "70199903","BIC": "SUBSDE03", "CountryCode": "280","Currency": "EUR", "HolderName": "Subsembly GmbH", "IBAN": "DE56701999030000002000" }
SerializedContact	FinTS Kontakt, wird automatisch aktualisiert sofern in den Response Options aktiviert
Service	Aktuelle FinTS Session, wird automatisch aktualisiert sofern in den Connection aktiviert
Order	Aktueller FinTS Auftrag, wird automatisch aktualisiert

Requests:

Request-Name	**Endpoint**	**Beschreibung**
Contact Info	FinTS	Abrufen der Informationen eines Bankzugangs, z.B. Konten, TAN-Verfahren, erlaubte Geschäftsvorfälle
Contact Info (Serialized Contact)	FinTS	Abrufen der Informationen eines Bankzugangs auf Basis eines bekannten FinTS Kontakts
Contact Info (Resume)	FinTS	Abrufen der Informationen eines Bankzugangs auf Basis einer laufenden FinTS Session
Contact Info (Request SCA)	FinTS	Abrufen der Informationen eines Bankzugangs mit starker Kundenauthentifizierung
Contact Info (SetSecurityFunction)	FinTS	Abrufen der Informationen eines Bankzugangs, sofern noch eine Security Function benötigt wird (NeedSecurityFunction)
Contact Info (SetTanMediaName)	FinTS	Abrufen der Informationen eines Bankzugangs, sofern noch ein TAN Medium benötigt wird (NeedTanMediaName)
Contact Info	FinTS	Abrufen der Informationen eines

(SendTan)		Bankzugangs, sofern noch ein TAN Medium benötigt wird (NeedTan)
Balance	FinTS	Abrufen von Salden für ein Konto
Statement CAMT	FinTS	Abrufen von Umsätzen im CAMT Format
Statement MT940	FinTS	Abrufen von Umsätzen im MT940 Format
Sepa Bank Transfer	FinTS	Einreichen einer SEPA-Einzelüberweisung
Sepa Bank Transfer INSTANT	FinTS	Einreichen einer SEPA-Echtzeitüberweisung
Send TAN	FinTS	Freigabe eines TAN-pflichtigen Auftrags
Kreditkartensaldo	FinTS	Abruf des Kreditkartensaldos
Kreditkartenumsätze	FinTS	Abruf der Kreditkartenumsätze
TAN Media List	FinTS	Abrufen der TAN Medien
Logout	FinTS	Beenden einer FinTS Session
Get Bank Info	Info	Abrufen Bankinformationen anhand der angegebenen Bankleitzahl
Get IBAN from AcctNo/BankCode	Info	Berechnen der IBAN auf Basis von Kontonummer und Bankleitzahl
EPC Code	Info	Erstellung eines EPC069 QR-Codes
XS2A Contact Info	Info	Abrufen von Informationen für einen XS2A Service
Contact Info CACC Id	XS2A	Abrufen der Konten/Kontaktinformationen für einen XS2A Bankzugang anhand der ServiceId
Contact Info CRDC Id	XS2A	Abrufen der Konten/Kontaktinformationen für einen XS2A Kreditkartenservice anhand der ServiceId
Contact Info CACC Acct	XS2A	Abrufen der Konten/Kontaktinformationen für einen XS2A Bankzugang anhand der Kontinformationen
Contact Info CRDC Acct	XS2A	Abrufen der Konten/Kontaktinformationen für einen XS2A Kreditkartenservice anhand der Kreditkarteninfo
Balance CACC	XS2A	Abruf der Salden (Bankkonto)
Balance CRDC	XS2A	Abruf der Salden (Kreditkarte)

Statement CACC	XS2A	Abruf der Umsätze (Bankkonto)
Statement CRDC	XS2A	Abruf der Umsätze (Kreditkarte)
Health Check	HC	Echtzeit Health Check Abfrage

Weitere Informationen

Subsembly BankAccessServer

Webseite: https://subsembly.com/bank-access-server.html

OpenAPI Spezifikation: https://subsembly.com/apidoc/bas/

Produktinformationen: https://subsembly.com/download/BankAccessServer.pdf

API Spezifikation: https://subsembly.com/download/BankAccessServerClientInterface.pdf

Installation: https://subsembly.com/download/BankAccessServerInstallation.pdf

EBICS: https://subsembly.com/download/BankAccessServerEBICS.pdf

Subsembly Banking APIs / SDKs

FinTS API: https://subsembly.com/fints-api.html

Ebics API: https://subsembly.com/ebics-api.html

SEPA API: https://subsembly.com/xs2a-api.html

XS2A API: https://subsembly.com/xs2a-api.html

Spezifikationen

Subsembly Payments Datenformate (SUPA): https://subsembly.com/supa.html

Deutsche Kreditwirtschaft / EBICS:
https://die-dk.de/zahlungsverkehr/electronic-banking/dfu-verfahren-ebics/

Deutsche Kreditwirtschaft / FinTS:

https://die-dk.de/zahlungsverkehr/electronic-banking/fints/

Deutsche Kreditwirtschaft / PSD2 Kontoschnittstelle:

https://die-dk.de/zahlungsverkehr/electronic-banking/psd2-kontoschnittstelle/

NextGenPSD2 Access to Account Interoperability Framework

PSD2 Access to Bank Accounts

https://www.berlin-group.org/psd2-access-to-bank-accounts

Laufzeitumgebung

Microsoft .NET Core: https://dotnet.microsoft.com/en-us/download/dotnet

Codegenerierung

Swagger CodeGen: http://swagger.io/swagger-codegen/