Implementierungsleitfaden API ennoxx.banking

Version: 1.4
Datum: 04.05.2022

ementexx GmbH
Berner Str. 109
D-60437 Frankfurt am Main

info@ementexx.com
www.ementexx.com

Versionshistorie

Version	Datum	Beschreibung
1.2	Bis 23.11.2021	Finalisierung der Public-Version
1.3	Bis 08.04.2022	Erweiterung um die ab Version 2.1.1 enthaltenen neuen Funktionalitäten bzgl. Auslandszahlungen (DTAZV) und Cash Forecasts
1.4	Bis 04.05.2022	Erweiterung um die ab Version 2.2.1 enthaltene neue Funktionalität „Aktivieren des Sendens nach Erstunterschrift (VEU)“, sowie um das Kapitel „Automatisierte API-Client Erstellung mittels NSwagStudio“

Inhalt

1 Einleitung 7

2 EBICS im Überblick 8

2.1 Was ist EBICS? 8
2.1.1 Historie 8
2.1.2 Hintergrund 8
2.1.3 Vorteile der EBICS Kommunikation 9
2.1.4 Übersicht der EBICS Versionen 9
2.1.5 Voraussetzungen 10
2.2 EBICS-Schlüssel und -management 11
2.2.1 Schlüsselmanagement 11
2.2.2 Elektronische Signaturen 11
2.2.3 Teilnehmer Initialisierung 12
2.2.3.1 Beispiel INI Brief 12
2.2.3.2 Verifizierung durch Abholung der Bankschlüssel 13
2.3 Auftragsarten 14
2.3.1 Sendeaufträge 14
2.3.1.1 Senden an die Bank (u.a. Zahlungsdateien) 14
2.3.2 Abholaufträge 15
2.3.2.1 Abholung von der Bank (u.a. Protokolle und Kontoinformationen) 15
2.3.2.2 Beispiel Kundenprotokolle: PTK/HAC 16

3 Ablaufbeschreibung 17

3.1 Phase 1: Vorbereitung 18
3.2 Phase 2: Initiierung 18
3.3 Phase 3: Teilnehmer-Initialisierung 19
3.4 Phase 4: Zahlungen 19
3.5 Phase 5: Kontoinformationen 20

4 Anbindung der REST-API 21

4.1 URL & Swagger-UI 21
4.2 Automatisierte API-Client Erstellung mittels NSwagStudio 22
4.2.1 Generierung des Quellcodes 22
4.3 Autorisierung 23
4.3.1 Möglichkeiten der Anmeldung 23
4.3.2 Technischer Benutzer 23
4.3.3 Benutzer des Kunden 23

5 Initiierung von SEPA-Überweisungen (SCT) 24

5.1 Variante 1 – Fertige XML-Datei 24
5.1.1 Übermittlung der XML-Zahlungsdatei 24
5.1.1.1 Aufbau des Http-Requests 24
5.1.1.2 Aufbau der Http-Response 25
5.1.1.3 Code-Beispiel (C#) 27
5.2 Variante 2 – Einzelzahlungen über ennoxx.banking 28
5.2.1 Anlage von SEPA-Überweisungen 28
5.2.1.1 Aufbau des Http-Requests 28
5.2.1.2 Aufbau der Http-Response 30
5.2.1.3 Code-Beispiel (C#) 32
5.2.2 Bereitstellen von SEPA- Überweisungen 33
5.2.2.1 Aufbau des Http-Requests 33
5.2.2.2 Aufbau der Http-Response 34
5.2.2.3 Code-Beispiel (C#) 36
5.3 Aktivieren des Sendens nach Erstunterschrift (VEU) 37
5.3.1 Aufbau des Http-Requests 37
5.3.2 Aufbau der Http-Response 38

5.3.3 Code-Beispiel (C#) 42
5.4 Autorisierung von SEPA-Überweisungen 43
5.4.1 Aufbau des Http-Requests 43
5.4.2 Aufbau der Http-Response 44
5.4.3 Code-Beispiel (C#) 48
5.5 Ermittlung des Status eines EBICS-Auftrages mit SEPA-Überweisungen 49
5.5.1 Variante 1 – Abfrage mittels REST-API 49
5.5.1.1 Aufbau des Http-Requests 49
5.5.1.2 Aufbau der Http-Response 50
5.5.1.3 Code-Beispiel (C#) 54
6 Initiierung von SEPA-Lastschriften (SDD) 55
6.1 Variante 1 – Fertige XML-Datei 55
6.1.1 Übermittlung der XML-Zahlungsdatei 55
6.2 Variante 2 – Einzelzahlungen über ennoxx.banking 56
6.2.1 Anlage von SEPA-Lastschriften 56
6.2.1.1 Aufbau des Http-Requests 56
6.2.1.2 Aufbau der Http-Response 58
6.2.1.3 Code-Beispiel (C#) 60
6.2.2 Bereitstellen von SEPA-Lastschriften 61
6.2.2.1 Aufbau des Http-Requests 61
6.2.2.2 Aufbau der Http-Response 62
6.2.2.3 Code-Beispiel (C#) 64
6.3 Autorisierung von SEPA-Lastschriften 65
6.4 Abfrage des Status eines EBICS-Auftrages mit SEPA-Lastschriften 65
6.4.1 Variante 1 – Abfrage mittels REST-API 65
7 Initiierung von Auslandszahlungen (DTAZV) 66
7.1 Variante 1 – Fertige DTAZV-Datei 66
7.1.1 Übermittlung der DTAZV-Zahlungsdatei 66
7.2 Variante 2 – Einzelzahlungen über ennoxx.banking 67
7.2.1 Anlage von Auslandszahlungen (DTAZV) 67
7.2.1.1 Aufbau des Http-Requests 67
7.2.1.2 Aufbau der Http-Response 70
7.2.1.3 Code-Beispiel (C#) 73
7.2.2 Bereitstellen von Auslandszahlungen (DTAZV) 74
7.2.2.1 Aufbau des Http-Requests 74
7.2.2.2 Aufbau der Http-Response 75
7.2.2.3 Code-Beispiel (C#) 77
7.3 Autorisierung von Auslandszahlungen (DTAZV) 78
7.4 Abfrage des Status eines EBICS-Auftrages mit Auslandszahlungen 78
7.4.1 Variante 1 – Abfrage mittels REST-API 78
8 Abholung von Kontoinformationen 79
8.1 Regelmäßige Abholung von Kontoinformationen von der Bank 79
8.1.1 Aufbau des Http-Requests 79
8.1.2 Aufbau der Http-Response 80
8.1.3 Code-Beispiel (C#) 84
8.2 Variante 1 – Originaldatei der Bank 85
8.2.1 Per FileExchange bereitgestellte Dateien abrufen 85
8.2.1.1 Aufbau des Http-Requests 86
8.2.1.2 Aufbau der Http-Response 87
8.2.1.3 Code-Beispiel (C#) 90
8.2.2 Erfolgreichen Abruf der Dateien bestätigen 91
8.2.2.1 Aufbau des Http-Requests 91
8.2.2.2 Aufbau der Http-Response 92
8.2.2.3 Code-Beispiel (C#) 93
8.3 Variante 2 – Einzeltransaktionen abrufen 94
8.3.1 Aufbau des Http-Requests 94
8.3.2 Aufbau der Http-Response 95

8.3.3 Code-Beispiel (C#) 97
9 Cash Forecasts (Finanzplandaten) 98
9.1 Anlage von Cash Forecasts 98
9.1.1 Aufbau des Http-Requests 98
9.1.2 Aufbau der Http-Response 100
9.1.3 Code-Beispiel (C#) 102
9.2 Cash Forecasts abrufen 103
9.2.1.1 Aufbau des Http-Requests 103
9.2.1.2 Aufbau der Http-Response 104
9.2.1.3 Code-Beispiel (C#) 106
9.3 Forecast Transaktionen abrufen 107
9.3.1.1 Aufbau des Http-Requests 107
9.3.1.2 Aufbau der Http-Response 108
9.3.1.3 Code-Beispiel (C#) 109
10 On-Boarding 110
10.1 Anlage Unternehmen 110
10.1.1 Aufbau des Http-Requests 110
10.1.2 Aufbau der Http-Response 111
10.1.3 Code-Beispiel (C#) 112
10.2 Anlage Benutzer 113
10.2.1 Aufbau des Http-Requests 113
10.2.2 Aufbau der Http-Response 115
10.2.3 Code-Beispiel (C#) 116
10.3 Passwortänderung eines Benutzers 117
10.3.1 Aufbau des Http-Requests 117
10.3.2 Aufbau der Http-Response 118
10.3.3 Code-Beispiel (C#) 119
10.4 Zuweisung von Benutzerrollen zu einem Benutzer 120
10.4.1 Aufbau des Http-Requests 120
10.4.2 Aufbau der Http-Response 121
10.4.3 Code-Beispiel (C#) 122
10.5 Zuweisung von Unternehmen zu einem Benutzer 123
10.5.1 Aufbau des Http-Requests 123
10.5.2 Aufbau der Http-Response 124
10.5.3 Code-Beispiel (C#) 125
10.6 Erstellung Auth Token für einen Benutzer 126
10.6.1 Aufbau des Http-Requests 126
10.6.2 Aufbau der Http-Response 127
10.6.3 Code-Beispiel (C#) 128
10.7 Anlage Konten 129
10.7.1 Aufbau des Http-Requests 129
10.7.2 Aufbau der Http-Response 131
10.7.3 Code-Beispiel (C#) 132
10.8 Anlage EBICS-Zugang 133
10.8.1 Aufbau des Http-Requests 133
10.8.2 Aufbau der Http-Response 135
10.8.3 Code-Beispiel (C#) 137
10.9 EBICS-Teilnehmer zu EBICS-Zugang hinzufügen 138
10.9.1 Aufbau des Http-Requests 138
10.9.2 Aufbau der Http-Response 140
10.9.3 Code-Beispiel (C#) 142
10.10 Erstellung EBICS-Transportschlüssel 143
10.10.1 Aufbau des Http-Requests 143
10.10.2 Aufbau der Http-Response 144
10.10.3 Code-Beispiel (C#) 145

10.11 Erstellung EBICS-Unterschriftsschlüssel 146
10.11.1 Aufbau des Http-Requests 146
10.11.2 Aufbau der Http-Response 147
10.11.3 Code-Beispiel (C#) 148
10.12 Änderung Passwort des Unterschriftsschlüssels 149
10.12.1 Aufbau des Http-Requests 149
10.12.2 Aufbau der Http-Response 150
10.12.3 Code-Beispiel (C#) 151
10.13 Initialisierung der Benutzerschlüssel bei der Bank 152
10.13.1 Aufbau des Http-Requests 152
10.13.2 Aufbau der Http-Response 153
10.13.3 Code-Beispiel (C#) 157
10.14 Abholung der Bankschlüssel 158
10.14.1 Aufbau des Http-Requests 158
10.14.2 Aufbau der Http-Response 159
10.14.3 Code-Beispiel (C#) 163

1 Einleitung

Der Partner von ementexx möchte seine Plattform um die Möglichkeit einer direkten Anbindung an unterschiedliche Banken erweitern. Zukünftig sollen per EBICS Zahlungen ausgelöst und Kontoinformationen abgeholt werden können.

Diese Möglichkeit soll durch die Anbindung der ennoxx.banking REST-API geschaffen werden. Folgende Themen sind hierbei prozessual zu beachten/notwendig:

On-Boarding
- Anlage der Kundendaten und EBICS-Zugänge
- Durchführung der technischen EBICS-Initialisierung

Initiierung von Zahlungen
- Bereitstellung von Zahlungen (SEPA-Überweisungen (SCT) & SEPA-Lastschriften (SDD))
- Autorisierung der Zahlungen durch den Kunden

Abholung von Kontoinformationen
- automatisierte Abholung von Kontoinformationen bei der Bank
- Übermittlung der Kontoinformationen

2 EBICS im Überblick

2.1 Was ist EBICS?

EBICS ist ein Kommunikationsstandard, der 2007 eingeführt wurde:

- Eine gemeinsame, offene Standardschnittstelle für alle Unternehmen und Banken in Frankreich, Deutschland und der Schweiz zur Anbindung
- Höchste Nachrichtensicherheit durch "Electronic Distributed Signatures" (EDS) mit Verschlüsselung auf Transport- und Anwendungsebene
- Einheitlicher Zugang für alle Geschäftstransaktionen, einschließlich Lastschriften, Überweisungen, Cash Management, Kontoauszüge usw.
- Flexibilität bei der Unterstützung verschiedener Dateiformate wie XML, HTTPS, TLS und ZIP

2.1.1 Historie

- 2003 Beginn der Spezifikationsphase zur Etablierung von Folgeprodukten zu X.25-Protokollen
- 2007 Start in Deutschland
- 2008 Mandatierung in Deutschland
- 2011 Frankreich hat die Migration abgeschlossen
- 2015 Beitritt der Schweiz
- 2018 EBICS 3.0 Spezifikation als harmonisierte Version
- 2020 Österreich beigetreten
- 2021 EBICS 3.0 wird verpflichtend

2.1.2 Hintergrund

Seit 2007 ist es für deutsche Banken verpflichtend, EBICS (Electronic Banking Internet Communication Standard) zu unterstützen, ein offenes IP-basiertes Kommunikationsprotokoll für die Verbindung zwischen Unternehmen und Banken. Durch die Verwendung eines gemeinsamen, bankneutralen Protokolls können Unternehmen mit allen Banken in Deutschland einheitlich und kostengünstig kommunizieren.

EBICS zielt darauf ab, die Einführung dieses Standards auf internationaler Ebene zu beschleunigen, so dass Banken EBICS als einheitlichen Kommunikationsstandard für den sicheren Austausch von Nachrichten nutzen können.

EBICS verfügt auch über eine "Multi-Bank-Fähigkeit", die es Firmenkunden generell ermöglicht, jede Bank anzusprechen, die den Standard unterstützt. Der EBICS-Standard ist heute in Deutschland, Frankreich, Österreich und der Schweiz weit verbreitet. Der EBICS-Standard unterstützt SEPA und kann als sicherer Kommunikationskanal genutzt werden, um SEPA-Lastschriften und SEPA-Überweisungen über das Internet einzuleiten.

2.1.3 Vorteile der EBICS Kommunikation

- Sichere Kommunikation zwischen Bank und Unternehmen
  - Verschlüsselt
  - Authentifiziert
- Auftragsautorisierung mit elektronischer(n) Unterschrift(en)
- Austausch beliebiger Datenformate
- Austausch großer Daten-"Dateien" / Massen-"Dateien"
- Rationalisierung der Finanzprozesse auf Unternehmensseite
- Stapelverarbeitung möglich, keine Benutzerinteraktion im Kommunikationsprozess erforderlich

2.1.4 Übersicht der EBICS Versionen

EBICS 2.3 (2007)		Schlüssellänge	Hashverfahren
Protokoll	H002	-	-
Authentifikation	X001	1.024 bis 16.384	SHA1
Verschlüsselung	E001	1.024 bis 16.384	SHA1
Unterschrift	A004	1.024	RIPEMD160

EBICS 2.4 (2008)		Schlüssellänge	Hashverfahren
Protokoll	H003	-	-
Authentifikation	X002	1.024 bis 16.384	SHA256
Verschlüsselung	E002	1.024 bis 16.384	SHA256
Unterschrift	A004	1.024	RIPEMD160
	A005	1.536 bis 4.096	SHA256
	A006	1.536 bis 4.096	SHA256

EBICS 2.5 (2011)		Schlüssellänge	Hashverfahren
Protokoll	H004	-	-
Authentifikation	X002	1.024 bis 16.384	SHA256
Verschlüsselung	E002	1.024 bis 16.384	SHA256
Unterschrift	A004	1.024	RIPEMD160
	A005	1.536 bis 4.096	SHA256
	A006	1.536 bis 4.096	SHA256

EBICS 3.0 (2017)		Schlüssellänge	Hashverfahren
Protokoll	H005	-	-
Authentifikation	X002	2.048 bis 16.384	SHA256
Verschlüsselung	E002	2.048 bis 16.384	SHA256
Unterschrift	A005	2.048 bis 4.096	SHA256
	A006	2.048 bis 4.096	SHA256

2.1.5 Voraussetzungen

- EBICS-Vertrag mit der Bank
  - Kunden-ID
  - Teilnehmer-ID
  - Vereinbarte Auftragsarten
- EBICS-Server auf Bankenseite
  - Schlüssel für Verschlüsselung und Authentifizierung
- EBICS-Client auf Kundenseite
  - Schlüssel für Verschlüsselung, Authentifizierung und Signatur

2.2 EBICS-Schlüssel und -management

2.2.1 Schlüsselmanagement

Das EBICS-Protokoll sieht für jeden Teilnehmer drei RSA-Schlüsselpaare vor. Diese werden für die folgenden Zwecke verwendet:

- Banktechnische/technische elektronische Signatur (ES) der Auftragsdaten, die das Teilnehmer-/Kundensystem an das Banksystem sendet (A00x)
- Identifikation und Authentifizierung des Teilnehmers durch das Banksystem mittels Identifikations- und Authentifizierungssignatur (X00x)
- Entschlüsselung des (symmetrischen) Transaktionsschlüssels zur Verschlüsselung der Auftragsdaten, die der Teilnehmer vom Banksystem abruft (E00x)

2.2.2 Elektronische Signaturen

Die elektronischen Signaturen sind in verschiedene Signaturklassen unterteilt. Je nach Art der Erstsignatur kann somit eine weitere Signatur notwendig sein (siehe nachfolgende Tabelle).

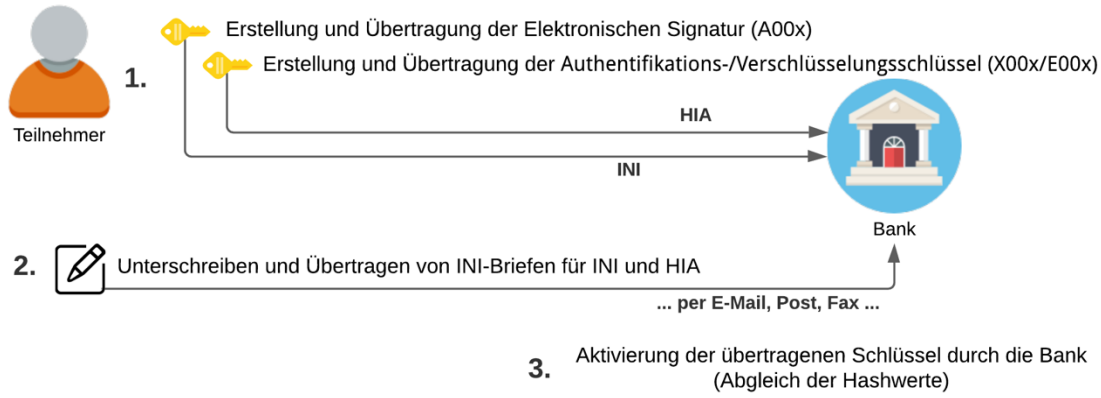
Signaturklassen:

- Einzelsignatur (Typ E)
- Erste Unterschrift (Typ A)
- Zweite Unterschrift (Typ B)
- Transportsignatur (Typ T)

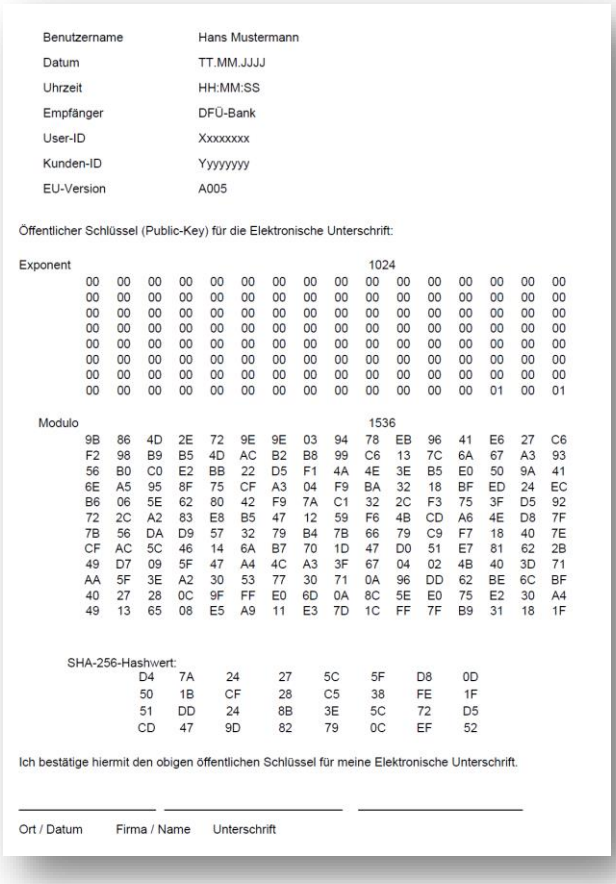
Erste EU ► ▼ Zweite EU	E	A	B	T
E	√	√	√	
A	√	√	√	
B	√	√		
T				

2.2.3 Teilnehmer Initialisierung

Für die EBICS Initialisierung werden die beiden Auftragsarten INI und HIA zur Übermittelung des öffentlichen Schlüssels des Teilnehmers genutzt. Wobei der INI Auftrag zur Übermittlung der Elektronischen Signatur und der HIA Auftrag für die Übermittlung der Authentifikations-/Verschlüsselungsschlüssel dient. Die Authentifizierung wird ebenfalls durch einen zweiten Mechanismus sichergestellt, indem das initiierende Kundensystem ein Initialisierungsprotokoll erstellt, welches zum Abgleich an die Bank übermittelt werden muss.



2.2.3.1 Beispiel INI Brief



Benutzername	Hans Mustermann
Datum	TT.MM.JJJJ
Uhrzeit	HH:MM:SS
Empfänger	DFÜ-Bank
User-ID	Xxxxxxxx
Kunden-ID	Yyyyyyyy
EU-Version	A005

Öffentlicher Schlüssel (Public-Key) für die Elektronische Unterschrift:

Exponent 1024

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 01 00 01
```

Modulo 1536

```
9B 86 4D 2E 72 9E 9E 03 94 78 EB 96 41 E6 27 C6
F2 98 B9 B5 4D AC B2 B8 99 C6 13 7C 6A 67 A3 93
56 B0 C0 E2 BB 22 D5 F1 4A 4E 3E B5 E0 50 9A 41
6E A5 95 8F 75 CF A3 04 F9 BA 32 18 BF ED 24 EC
B6 06 5E 62 80 42 F9 7A C1 32 2C F3 75 3F D5 92
72 2C A2 83 E8 B5 47 12 59 F6 4B CD A6 4E D8 7F
7B 56 DA D9 57 32 79 B4 7B 66 79 C9 F7 18 40 7E
CF AC 5C 46 14 6A B7 70 1D 47 D0 51 E7 81 62 2B
49 D7 09 5F 47 A4 4C A3 3F 67 04 02 4B 40 3D 71
AA 5F 3E A2 30 53 77 30 71 0A 96 DD 62 BE 6C BF
40 27 28 0C 9F FF E0 6D 0A 8C 5E E0 75 E2 30 A4
49 13 65 08 E5 A9 11 E3 7D 1C FF 7F B9 31 18 1F
```

SHA-256-Hashwert:

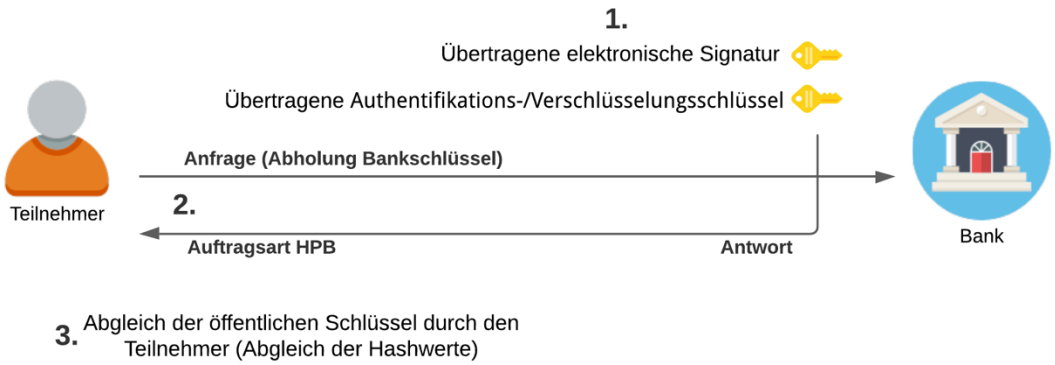
```
D4 7A 24 27 5C 5F D8 0D
50 1B CF 28 C5 38 FE 1F
51 DD 24 8B 3E 5C 72 D5
CD 47 9D 82 79 0C EF 52
```

Ich bestätige hiermit den obigen öffentlichen Schlüssel für meine Elektronische Unterschrift.

Ort / Datum Firma / Name Unterschrift

2.2.3.2 Verifizierung durch Abholung der Bankschlüssel

Voraussetzung für die Übermittlung von Aufträgen per EBICS ist der Download der Bankschlüssel, welche über die Auftragsart HPB abgerufen und anschließend abgeglichen werden müssen (Abgleich der Hashwerte).



3. Abgleich der öffentlichen Schlüssel durch den Teilnehmer (Abgleich der Hashwerte)

2.3 Auftragsarten

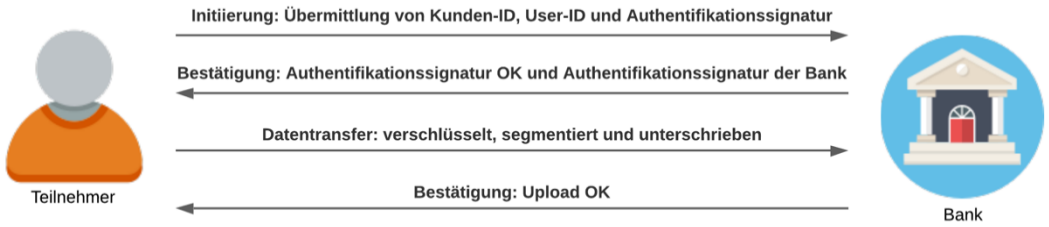
2.3.1 Sendeaufträge

Die nachfolgende Tabelle beinhaltet die gängigsten Sendeauftragsarten in Deutschland.

Auftragsart	Beschreibung	Format
AZV	Senden Auslandszahlungen im Diskettenformat	DTAZV
CCT	Senden SEPA Überweisung (Customer Credit Transfer)	pain.001
CCU	Senden Euro-Eil-Überweisung	pain.001
CIP	Senden eines Sammlers mit (terminierten) SEPA-Echtzeitüberweisungen: Instant Payment	pain.001
CDB	Senden SEPA Firmenlastschrift (B2B)	pain.008
CDD	Senden SEPA Basislastschrift (CORE)	pain.008
RFT	Senden Request for Transfer	MT101

2.3.1.1 Senden an die Bank (u.a. Zahlungsdateien)

Zum Versand von Daten per EBICS muss die Konnektivität seitens des Kundensystems initiiert werden. Hierbei übermittelt der Teilnehmer bei der Anfrage die entsprechende Authentifikationssignatur, Kunden- und User-ID, Auftragsart, Signatur und Daten. Die Bank prüft die Anfrage, die Berechtigung auf die Auftragsart, Gültigkeit der Signatur und Daten. Der Vorgang ist abgeschlossen, nachdem das Banksystem die erfolgreiche Datenübermittlung bestätigt hat.



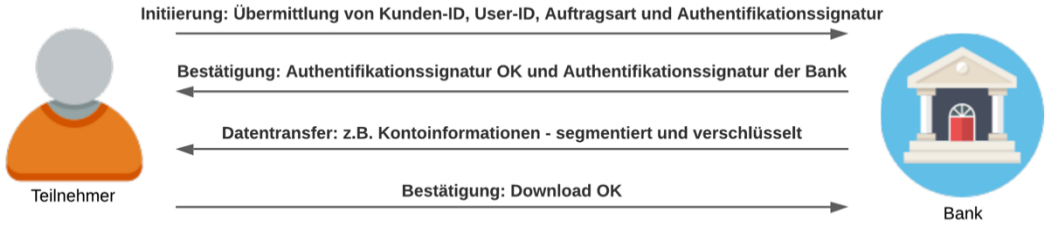
2.3.2 Abholaufträge

Die nachfolgende Tabelle beinhaltet die gängigsten Abholauftragsarten in Deutschland.

Auftragsart	Beschreibung	Format
C52	Abholen camt-Vormerkposten (VMK)	camt.052
C53	Abholen camt-Tagesauszug (STA)	camt.053
C54	Abholen Sammelbuchungsinformationen (DTI)	camt.054
CBC	Abholen Payment Status Report für SEPA Lastschriften via XML-Container	pain.002
CDZ	Abholen Payment Status Report für SEPA Lastschriften	pain.002
CRC	Abholen Payment Status Report für SEPA Überweisungen via XML-Container	pain.002
CRZ	Abholen Payment Status Report für SEPA Überweisungen	pain.002
STA	Abholen Swift-Tagesauszüge	MT940
VMK	Abholen kurzfristige Vormerkposten	MT942

2.3.2.1 Abholung von der Bank (u.a. Protokolle und Kontoinformationen)

Zur Abholung von Daten per EBICS muss die Konnektivität seitens des Kundensystems initiiert werden. Hierbei übermittelt der Teilnehmer bei der Anfrage die entsprechende Auftragsart, Authentifikationssignatur, Kunden- und User-ID. Die Bank prüft die Anfrage, ebenfalls die Berechtigung auf die Auftragsart, und übermittelt die angeforderten Daten nach erfolgreicher Prüfung zurück. Der Vorgang ist abgeschlossen, nachdem das Kundensystem die erfolgreiche Datenübermittlung bestätigt hat.



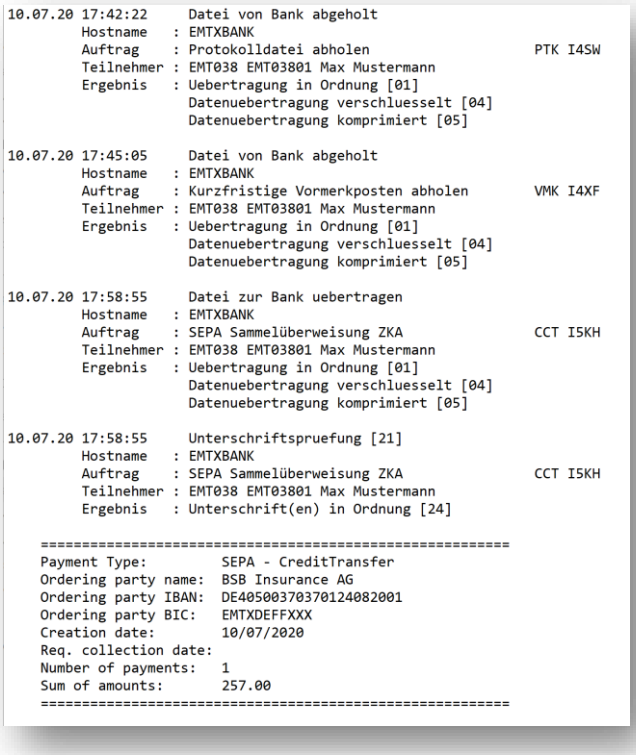
2.3.2.2 Beispiel Kundenprotokolle: PTK/HAC

Kundenprotokolle dokumentieren die folgenden Prozesse im Zusammenhang mit Kundenaufträgen:

- Übermittlung von Auftragsdaten zum und vom Banksystem
- Übermittlung von elektronischen Unterschriften
- Nachbearbeitung von Aufträgen, soweit es sich um die Unterschriftenprüfung, die Anzeige von Auftragsdaten oder Fehler bei der Dekomprimierung handelt

Verwendete Auftragsarten:

- PTK (Textformat)
- HAC (pain.002 XML-Format)



```
10.07.20 17:42:22     Datei von Bank abgeholt
         Hostname   : EMTXBANK
         Auftrag    : Protokolldatei abholen                    PTK I4SW
         Teilnehmer : EMT038 EMT03801 Max Mustermann
         Ergebnis   : Uebertragung in Ordnung [01]
                      Datenuebertragung verschluesselt [04]
                      Datenuebertragung komprimiert [05]

10.07.20 17:45:05     Datei von Bank abgeholt
         Hostname   : EMTXBANK
         Auftrag    : Kurzfristige Vormerkposten abholen        VMK I4XF
         Teilnehmer : EMT038 EMT03801 Max Mustermann
         Ergebnis   : Uebertragung in Ordnung [01]
                      Datenuebertragung verschluesselt [04]
                      Datenuebertragung komprimiert [05]

10.07.20 17:58:55     Datei zur Bank uebertragen
         Hostname   : EMTXBANK
         Auftrag    : SEPA Sammelüberweisung ZKA                CCT I5KH
         Teilnehmer : EMT038 EMT03801 Max Mustermann
         Ergebnis   : Uebertragung in Ordnung [01]
                      Datenuebertragung verschluesselt [04]
                      Datenuebertragung komprimiert [05]

10.07.20 17:58:55     Unterschriftspruefung [21]
         Hostname   : EMTXBANK
         Auftrag    : SEPA Sammelüberweisung ZKA                CCT I5KH
         Teilnehmer : EMT038 EMT03801 Max Mustermann
         Ergebnis   : Unterschrift(en) in Ordnung [24]

    ============================================================
    Payment Type:          SEPA - CreditTransfer
    Ordering party name:   BSB Insurance AG
    Ordering party IBAN:   DE40500370370124082001
    Ordering party BIC:    EMTXDEFFXXX
    Creation date:         10/07/2020
    Req. collection date:
    Number of payments:    1
    Sum of amounts:        257.00
    ============================================================
```

3 Ablaufbeschreibung

Die nachfolgende Tabelle beschreibt die verschiedenen Phasen und Schritte der Implementierung eines neuen EBICS-Zugangs sowie deren Zuständigkeit.

Phase	Beschreibung	Zuständigkeit
1. Vorbereitung	Beantragung der EBICS Zugänge bei den Banken	Kunde
	Übermittlung der EBICS-Zugangsdaten an Partner oder ementexx	Kunde
2. Initiierung	Übermittlung der Informationen an ennoxx.banking (Anlage Kunde, Benutzer, Unternehmen, EBICS-Zugänge etc.)	Partner, Kunde oder ementexx
3. Teilnehmer-Initialisierung	Schlüsselerzeugung und Erzeugung der EBICS-Initialisierungsaufträge (INI und HIA)	Partner oder Kunde
	Übermittlung der unterschriebenen INI-Briefe an die Bank	Kunde
	Abruf der Bankschlüssel mittels HPB nach Teilnehmeraktivierung durch die Bank	Partner oder Kunde
	Ggf. Änderung des Passworts für die EBICS Signatur	Kunde
4. Zahlungen	Übermittlung von Zahlungen an ennoxx.banking (Zahlungsdateien oder Einzeltransaktionen) und Erzeugung des EBICS-Sendeauftrags	Partner
	Optional: Aktivieren des Sendens nach Erstunterschrift zur Nutzung der VEU	Kunde
	Freigabe des EBICS-Senderauftrags durch Elektronische Unterschrift	Kunde
	Statusauswertung des EBICS-Sendeauftrags	Partner
5. Kontoinformationen	Automatischer Abruf der Kontoinformationen durch wiederkehrenden EBICS-Abholauftrag	ennoxx.banking
	Abruf der Kontoinformationen (Originaldatei oder Einzeltransaktionen)	Partner oder Kunde

3.1 Phase 1: Vorbereitung

In dieser Phase beantragt der jeweilige Kunde den EBICS-Zugang direkt bei seiner Bank und übergibt nach erfolgreicher Beantragung die entsprechenden Zugangsinformationen an den Partner oder an ementexx. Für die Beantragung muss der Kunde u.a. festlegen welche Konten, Auftragsarten, Benutzer (inkl. Autorisierungsberechtigung) für den angeforderten EBICS-Zugang hinterlegt werden sollen.

3.2 Phase 2: Initiierung

Nachdem der Kunde die Zugangsdaten in Phase 1 übermittelt hat, werden diese Daten in ennoxx.banking angelegt. Bei einer Implementierung durch den Partner in dessen Lösung kommen die folgenden Endpunkte der REST-API zum Einsatz:

Beschreibung	Endpunkt
Anlage des Unternehmens (Identity)	10.1
Anlage des ennoxx.banking Benutzers für die Anmeldung an die REST-API, sowie zur Zuordnung zum EBICS-Zugang	10.2
Einmalige Änderung des Initialpassworts des angelegten Benutzers	10.3
Zuweisung von Benutzerrollen zum angelegten Benutzer um dessen Zugriff auf funktionaler Ebene entsprechend einzuschränken	10.4
Zuweisung der Unternehmen zum angelegten Benutzer um dessen Zugriff datenbezogen einzuschränken	10.5
Erstellung eines Auth-Tokens für die Anmeldung des angelegten Benutzers an die REST-API	10.6
Anlage der Konten des jeweiligen Unternehmens (optional: dieser Schritt könnte auch automatisch nach erfolgreicher EBICS-Benutzerinitialisierung erfolgen)	10.7
Anlage des EBICS-Zugangs mit den vom Kunden übermittelten Zugangsdaten	10.8
Anlage des EBICS-Teilnehmers zum angelegten EBICS-Zugang	10.9

3.3 Phase 3: Teilnehmer-Initialisierung

Im Anschluss an die Anlage des EBICS-Zugangs in Phase 2, wird der Teilnehmer bei der Bank initialisiert. Hierbei kommen die folgenden Endpunkte der REST-API zum Einsatz:

Beschreibung	Endpunkt
Erstellung der EBICS-Transportschlüssel (X00x für Authentifikation und E00x für Verschlüsselung)	10.10
Erstellung des EBICS-Unterschriftschlüssels (A00x)	10.11
Änderung des Passworts für den EBICS-Unterschriftsschlüssel durch den Kunden (optional: nur relevant falls Partner den Unterschlüssel vorgibt)	10.12
Initialisierung der Benutzerschlüssel bei der Bank (Die Initialisierungsbriefe müssen anschließend vom Kunden unterschrieben und an die Bank übermittelt werden)	10.13
Abholung der Bankschlüssel nach Freischaltung des Teilnehmers durch die Bank	10.14

3.4 Phase 4: Zahlungen

Sobald der EBICS-Zugang mit Phase 3 aktiviert wurde, können auf zwei verschiedenen Wegen SEPA-Überweisungen an ennoxx.banking übermittelt und ausgeführt werden. Hierbei kommen die folgenden Endpunkte der REST-API zum Einsatz:

Beschreibung	Endpunkt
Variante 1: Übermittlung einer fertigen XML-Datei und Erzeugung des EBICS Auftrags	5.1.1
Variante 2: Anlage einer SEPA-Überweisung (für Erzeugung der XML-Datei über ennoxx.banking)	5.2.1
Variante 2: Bereitstellung der SEPA-Überweisung (für Erzeugung der XML-Datei über ennoxx.banking) und Erzeugung des EBICS Auftrags	5.2.2
Beide Varianten: Aktivieren des Sendens nach Erstunterschrift (optional: nur relevant falls die VEU genutzt werden soll)	5.3
Beide Varianten: Autorisierung des zuvor erzeugten EBICS-Auftrags (Übermittlung an die Bank erfolgt nach vollständiger Autorisierung automatisch)	5.4
Beide Varianten: Überprüfung des Status des EBICS-Auftrags	5.5

Hinweis: Für SEPA-Lastschriften siehe analog Kapitel 6, für Auslandszahlungen (DTAZV) siehe Kapitel 7.

3.5 Phase 5: Kontoinformationen

Sobald der EBICS-Zugang mit Phase 3 aktiviert wurde, können Kontoinformationen von ennoxx.banking abgerufen werden. Hierbei kommen die folgenden Endpunkte der REST-API zum Einsatz:

Beschreibung	Endpunkt
Erstellung des werktäglichen Abrufs der Kontoinformationen (muss einmal pro EBICS-Zugang angelegt werden und wird dann automatisch durch ennoxx.banking wiederholt)	8.1
Variante 1: Abruf von Originaldateien per File-Exchange (müsste zuvor manuell eingerichtet werden)	8.2.1
Variante 1: Bestätigung des Dateiabrufs	8.2.2
Variante 2: Abruf der Einzeltransaktionen	8.3

4 Anbindung der REST-API

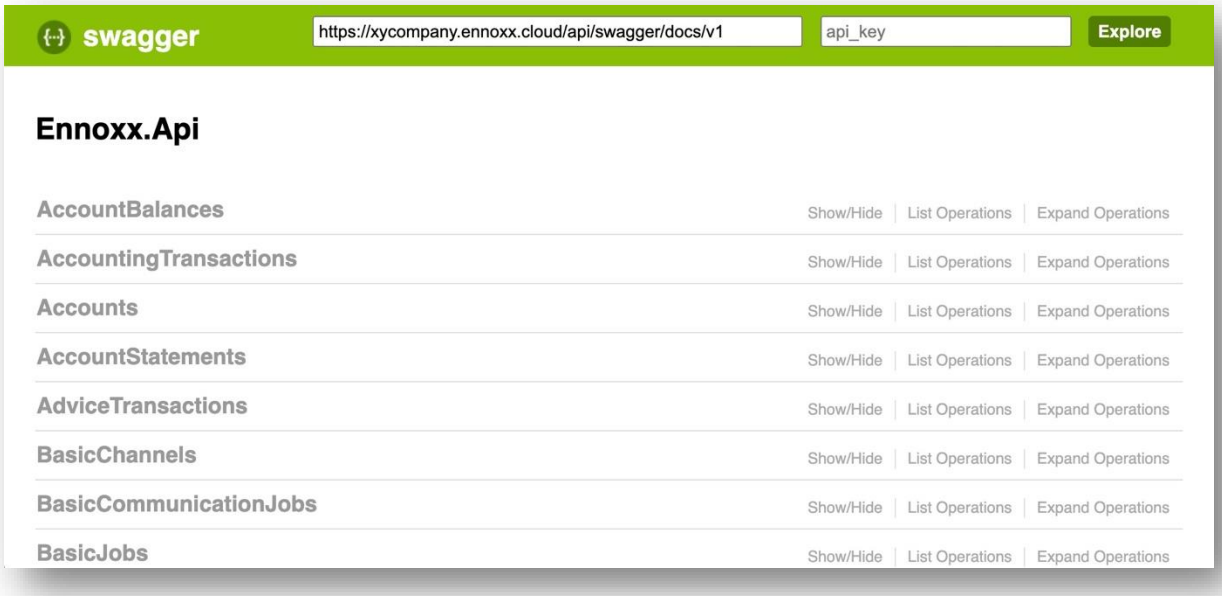
4.1 URL & Swagger-UI

Die ennoxx.banking REST-API kann über folgende URL erreicht werden:

https://xycompany.ennoxx.cloud/api

Mit der API wird zusätzlich ein Swagger-UI zur Verfügung gestellt, welches alle vorhandenen API-Controller, Methoden und die dabei verwendeten Dto-Objekte beschreibt. Zudem lassen sich alle Methoden ohne weiteren Aufwand direkt ausführen, es wird allerdings eine Benutzeranmeldung mit Benutzernamen und Passwort vorausgesetzt. Das Swagger-UI kann über folgenden Link erreicht werden:

https://xycompany.ennoxx.cloud/api/swagger/ui/index



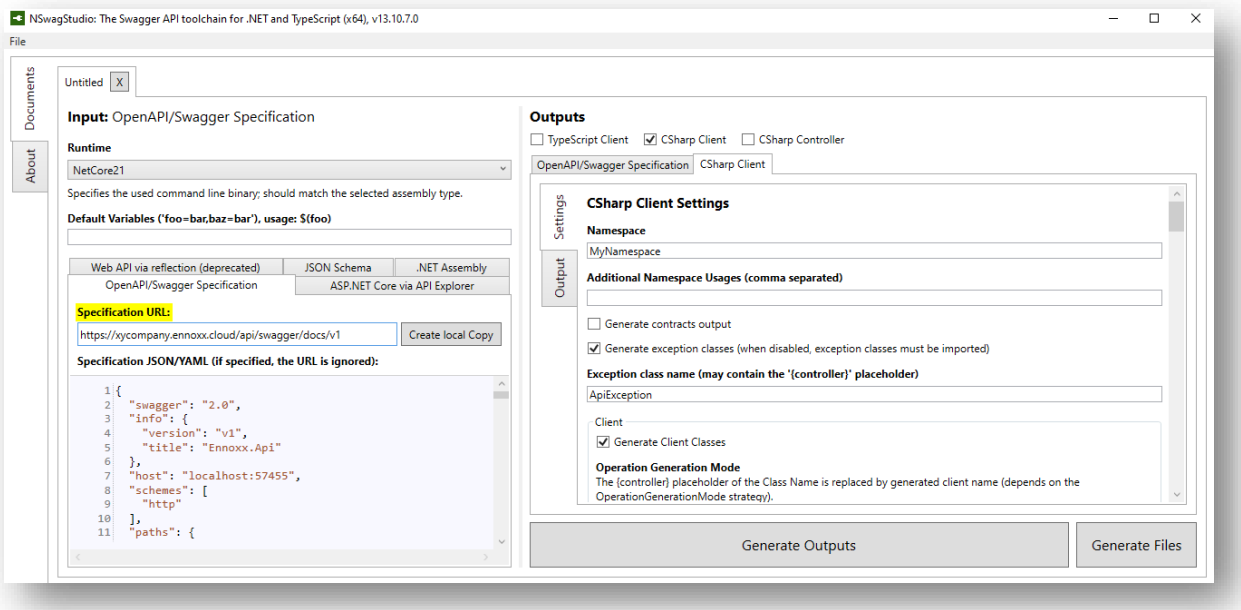
4.2 Automatisierte API-Client Erstellung mittels NSwagStudio

Auf Basis der bereitgestellten Swagger-UI können Tools für eine automatisierte Quellcode Erzeugung genutzt werden. Für die in diesem Dokument angegebenen Code-Beispiele wurde hierzu der mit den Standardeinstellungen erzeugte CSharp-Client aus NSwagStudio verwendet. NSwagStudio kann in der hier genutzten Version 13.10.7 unter dem nachfolgenden Link heruntergeladen werden:

https://github.com/RicoSuter/NSwag/releases/tag/NSwag-Build-1128

4.2.1 Generierung des Quellcodes

Im NSwagStudio kann unter „Specification URL“ die angezeigte URL aus der Swagger-UI angegeben und damit eine lokale Kopie der JSON Struktur erzeugt werden. Im Bereich „Outputs“ kann der gewünschte Ausgabeclient ausgewählt werden, zum Beispiel „CSharp Client“. Mittels des Knopfes „Generate Outputs“ wird der Quellcode erzeugt, den man anschließend in seiner eigenen Implementierung zur Anbindung der ennoxx.banking REST-API verwenden kann.



Weiterführende Informationen zur Nutzung von NSwagStudio können der Microsoft Dokumentation Get started with NSwag and ASP.NET Core oder dem Blog des Entwicklers NSwag Tutorial: Integrate the NSwag toolchain into your ASP.NET Web API project entnommen werden.

4.3 Autorisierung

4.3.1 Möglichkeiten der Anmeldung

Alle zur Verfügung gestellten Methoden der ennoxx.banking API können aus Sicherheitsgründen nur mit einem angemeldeten Benutzer aufgerufen werden. Für die Benutzeranmeldung stehen die beiden folgenden Möglichkeiten zur Auswahl. Die jeweiligen Anmeldedaten müssen dabei über den Authorization Header des Http-Requests mitgegeben werden. Welcher Anmeldetyp von welchem Benutzer verwendet werden kann, lässt sich in ennoxx.banking konfigurieren. Ein Auth Token muss zudem vom jeweiligen Benutzer selbst erstellt werden und kann zeitlich begrenzt werden.

Anmeldung mit Benutzernamen und Passwort (Basic-Authentication):

```
var userName = "username1";
var password = "password1";
var httpClient = new HttpClient();
var credentials = new NetworkCredential(userName, password);
var handler = new HttpClientHandler { Credentials = credentials };
var httpClient = new HttpClient(handler);
```

Anmeldung mit Auth Token (Bearer-Authentication):

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
```

4.3.2 Technischer Benutzer

Für sämtliche Prozesse (z.B. On-Boarding eines neuen Kunden, Bereitstellung von Zahlungen, etc.) kann ein zentraler technischer Benutzer verwendet werden. Dieser muss lediglich den Zugriff auf die REST-API erhalten. Für eine Vereinfachung der Prozessschritte kann die Anmeldung dieses Benutzers mittels Auth Token erfolgen.

4.3.3 Benutzer des Kunden

Für den oder die erzeugten Benutzer des Kunden kann für den technischen Login in die REST-API ebenfalls ein Auth Token verwendet werden, damit der Kunde neben dem Unterschriftspasswort keine weiteren Zugangsdaten merken muss.

5 Initiierung von SEPA-Überweisungen (SCT)

5.1 Variante 1 – Fertige XML-Datei

5.1.1 Übermittlung der XML-Zahlungsdatei

Der API-Controller `BasicCommunicationJobs` stellt für die Übermittlung einer XML-Zahlungsdatei in ennoxx.banking die Methode `CreateFromFile` zur Verfügung. Da der zugehörige Kommunikationskanal anhand der in der Zahlungsdatei hinterlegten Auftraggeberdaten automatisch ermittelt wird, müssen neben der Datei keine weiteren Informationen übermittelt werden. Der in diesem Zuge erzeugte Kommunikationsauftrag wird anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/BasicCommunicationJobs/BasicCommunicationJobs_CreateFromFile

5.1.1.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/BasicCommunicationJobs/CreateFromFile
Method	POST
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
Optionaler URL-Parameter „permissionClass“	Enthält den Namen der Zugriffsklasse, sofern diese dem Auftrag für einen eingeschränkten Zugriff zugeordnet werden soll

Der Dateiname und der Dateiinhalt der Zahlungsdatei werden im Http-Request als Datenstrom mitgegeben. Das hierbei zu nutzende `FileDto`-Objekt ist eine JSON-Struktur mit dem nachfolgenden Aufbau. Für einen eingeschränkten Zugriff kann eine Zugriffsklasse optional als URL-Parameter mitgegeben werden.

```
{
  "FileName": "string",
  "FileSize": 0,
  "Data": "string"
}
```

Feldname	Bezeichnung	Datentyp	Pflichtfeld	Beschreibung
FileName	Dateiname	String	Ja	Enthält den Dateinamen inkl. Dateiendung (z.B. Payments.xml)
FileSize	Dateigröße in Bytes	Integer	Nein	
Data	Dateiinhalt	String	Ja	Enthält den Dateiinhalt als Base64-codiertes Byte-Array

5.1.1.2 Aufbau der Http-Response

Die Http-Response liefert ein `BasicCommunicationJobListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:


```
{
  "Objects": [
    {
      "Channel": {
        "Caption": "string",
        "ChannelType": "Communication",
        "Name": "string",
        "Description": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "SpecificDetail": "string",
      "State": "None",
      "Caption": "string",
      "CommunicationDirection": "Send",
      "DueTime": "2021-09-20T14:39:31.462Z",
      "IntervalType": "Single",
      "Interval": 0,
      "LastExecution": "2021-09-20T14:39:31.462Z",
      "PermissionClass": "string",
      "WebUrl": "string",
      "LastLog": {
        "State": "Success",
        "DateTime": "2021-09-20T14:39:31.462Z",
        "Description": "string",
        "Protocol": "string"
      },
      "Files": [
        {
          "OriginalFile": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ContentType": "Undefined",
          "ReportHtml": "string",
          "ReportSummaryHtml": "string",
          "ReportPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ReportSummaryPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "FileHash": {
            "Method": "None",
            "Value": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        }
      ],
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ BasicCommunicationJobDto	Enthält den angelegten Kommunikationsauftrag
Channel	Kanal	BasicChannelDto	Enthält Basisinformationen des Kanals
SpecificDetail	Spezifische Details	String	Enthält eine zusammenfassende Kurzinformation des Auftrags
State	Status	String	Aktueller Status des Auftrags
Caption	Bezeichnung	String	
CommunicationDirection	Richtung	String	Enthält die Richtung des Auftrags (in diesem Fall: Senden)
DueTime	Ausführungszeitpunkt	DateTime	
IntervalType	Art des Intervalls	String	Einmaliger oder wiederkehrender Auftrag mit dem angegebenen Intervall
Interval	Intervall	Integer	
LastExecution	Letzte Ausführung am	DateTime	Nur belegt bei wiederkehrenden Aufträgen, für die ein letztes Ausführungsdatum festgelegt wurde
PermissionClass	Zugriffsklasse	String	
WebUrl	Web URL	String	Enthält den Weblink mit dem der Auftrag im Web UI angezeigt werden kann
LastLog	Letztes Log	BasicLogDto	Enthält Informationen des letzten Logs
Files	Dateien	Array vom Typ ContextFileDto	Enthält bei Sendeaufträgen die zu übertragenden Nutzdaten
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des erstellten Kommunikationsauftrags, welche für weitere Auswertungen und Methodenaufrufe verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall wird im `Objects`-Array der erzeugte Kommunikationsauftrag mit dessen Details (z.B. Kanal, Status, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

5.1.1.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `CreateFromFile` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var basicCommunicationJobsClient = new BasicCommunicationJobsClient(httpClient);
basicCommunicationJobsClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var fileDto = new FileDto();
fileDto.FileName = "Payments.xml";
fileDto.Data = File.ReadAllBytes(@"C:\Payments.xml");
var task = basicCommunicationJobsClient.CreateFromFileAsync(fileDto, null);
var result = task.Result;
if (result.Result == BasicCommunicationJobListResultDtoResult.Success)
{
   log.Info(result.Message);
   var oid = result.Objects.First().Oid;
   //...
}
else
{
   throw new Exception(result.Message);
}
```

5.2 Variante 2 – Einzelzahlungen über ennoxx.banking

5.2.1 Anlage von SEPA-Überweisungen

Der API-Controller `SepaCreditTransferPayments` stellt für die Anlage von SEPA-Überweisungen in ennoxx.banking die Methoden `Create2` (für eine Einzelanlage) sowie `CreateBulk2` (für die Anlage mehrerer SEPA-Überweisungen in einem Rutsch) zur Verfügung. Die in diesem Zuge erzeugten Zahlungen werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/SepaCreditTransferPayments/SepaCreditTransferPayments_Create2

5.2.1.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/SepaCreditTransferPayments/Create2 oder https://xycompany.ennoxx.cloud/api/SepaCreditTransferPayments/CreateBulk2
Method	POST
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json

Die Daten der zu erstellenden SEPA-Überweisung werden im Http-Request als Datenstrom mitgegeben. Das hierbei zu nutzende `SepaCreditTransferPaymentCreateRequestDto`-Objekt ist eine JSON-Struktur mit dem nachfolgenden Aufbau. Bei Nutzung der Methode `CreateBulk2` wird die gleiche Dto-Struktur in einem Array verwendet.

```
{
  "Amount": 0,
  "RemittanceInformation": "string",
  "DueTime": "2021-09-20T14:39:31.851Z",
  "IntervalType": "Single",
  "Interval": 0,
  "OrderingPartyAccountOidOrIban": "string",
  "EndToEndReference": "string",
  "PaymentType": "CreditTransfer",
  "BeneficiaryName": "string",
  "BeneficiaryIban": "string",
  "BeneficiaryBic": "string"
}
```

Feldname	Bezeichnung	Datentyp	Pflichtfeld	Beschreibung
Amount	Betrag	Number	Ja	
RemittanceInformation	Verwendungszweck	String	Nein	
DueTime	Fälligkeitstermin	DateTime	Nein	Sofern nicht angegeben, wird der aktuelle Tag verwendet
IntervalType	Art des Intervalls	String	Nein	Einmalige oder wiederkehrende Zahlung
Interval	Intervall	Integer	Nein	
OrderingPartyAccount OidOrIban	Auftraggeberkonto	String	Ja	Wahlweise die Objekt-ID (Oid) oder die IBAN des Auftraggeberkontos
EndToEndReference	Ende-Zu-Ende-Referenz	String	Nein	Sofern nicht belegt, wird eine Ende-Zu-Ende-Referenz durch ennoxx.banking erzeugt
PaymentType	Zahlungsart	String	Nein	• CreditTransfer • Urgent • Salary • CapitalBuilding • Charity • TreasuryPayment • InstantCreditTransfer • InstantSalary Sofern nicht belegt, wird „CreditTransfer“ verwendet
BeneficiaryName	Empfänger Name	String	Ja	
BeneficiaryIban	Empfänger IBAN	String	Ja	
BeneficiaryBic	Empfänger BIC	String	Nein	Zahlungen innerhalb Deutschlands sind IBAN only möglich

5.2.1.2 Aufbau der Http-Response

Die Http-Response liefert ein `SepaCreditTransferPaymentListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:


```
{
  "Objects": [
    {
      "EndToEndReference": "string",
      "PaymentType": "CreditTransfer",
      "BeneficiaryName": "string",
      "BeneficiaryIban": "string",
      "BeneficiaryBic": "string",
      "Amount": 0,
      "AmountInEur": 0,
      "CurrencyIsoCode": "string",
      "RemittanceInformation": "string",
      "DueTime": "2021-09-20T14:39:31.849Z",
      "IntervalType": "Single",
      "Interval": 0,
      "Status": "Open",
      "Description": "string",
      "OrderingPartyAccount": {
        "Description": "string",
        "Iban": "string",
        "Bic": "string",
        "AccountNumber": "string",
        "NationalBankCode": "string",
        "AccountIdentifier": "string",
        "AccountIdentifier2": "string",
        "AccountIdentifier3": "string",
        "AccountType": "OpenAccount",
        "AccountCategory": "string",
        "CurrencyIsoCode": "string",
        "AccountHolder": {
          "CompanyName": "string",
          "CreditorIdentification": "string",
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "PaymentProperties": [
        "string"
      ],
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ SepaCreditTransfer PaymentDto	Enthält die angelegten SEPA-Überweisungen
EndToEndReference	Ende-Zu-Ende-Referenz	String	
PaymentType	Zahlungsart	String	
BeneficiaryName	Empfängername	String	
BeneficiaryIban	Empfänger IBAN	String	
BeneficiaryBic	Empfänger BIC	String	
Amount	Betrag	Number	
AmountInEur	Betrag in EUR	Number	
CurrencyIsoCode	Währung	String	Bei SEPA-Zahlungen immer „EUR“
RemittanceInformation	Verwendungszweck	String	
DueTime	Fälligkeitstermin	DateTime	
IntervalType	Art des Intervalls	String	
Interval	Intervall	Integer	
Status	Status	String	Enthält den aktuellen Status der Zahlung: • Open (Offen) • Executed (Bereitgestellt) • Template (Vorlage)
Description	Beschreibung	String	
OrderingPartyAccount	Auftraggeberkonto	AccountDto	Enthält Details des Auftraggeberkontos
PaymentProperties	Eigenschaften	Array vom Typ String	Enthält alle der Zahlung zugeordnete Zahlungseigenschaften
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) der erstellten SEPA-Überweisung, welche für weitere Auswertungen und Methodenaufrufe (z.B. das Bereitstellen der Zahlungen - siehe nachfolgendes Kapitel) verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall werden im `Objects`-Array die erzeugten SEPA-Überweisungen mit deren Details (z.B. Betrag, Verwendungszweck, Fälligkeitstermin, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

5.2.1.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `Create2` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var sctPaymentsClient = new SepaCreditTransferPaymentsClient(httpClient);
sctPaymentsClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var sctPaymentDto = new SepaCreditTransferPaymentCreateRequestDto() { Amount = 12.34,
OrderingPartyAccountOidOrIban = "DE95500800000000001234", BeneficiaryName = "Max Mustermann",
BeneficiaryIban = "DE45300500000000333333", RemittanceInformation = "RG-Nr. 12345" };

var task = sctPaymentsClient.Create2Async(sctPaymentDto);
var result = task.Result;
if (result.Result == SepaCreditTransferPaymentListResultDtoResult.Success)
{
   log.Info(result.Message);
   foreach (var createdPayment in result.Objects)
   {
       var oid = createdPayment.Oid;
       //...
   }
}
else
{
   throw new Exception(result.Message);
}
```

5.2.2 Bereitstellen von SEPA- Überweisungen

Der API-Controller `SepaCreditTransferPayments` stellt für die Bereitstellung von SEPA-Überweisungen als Kommunikationsauftrag in ennoxx.banking die Methode `Provide` zur Verfügung. Der oder die in diesem Zuge erzeugten Kommunikationsaufträge werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/SepaCreditTransferPayments/SepaCreditTransferPayments_Provide

5.2.2.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/SepaCreditTransferPayments/Provide
Method	GET
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
URL-Parameter „criteria“	Enthält das Filterkriterium für die auszuführenden SEPA-Überweisungen. Beispielsweise können an dieser Stelle gezielt eine oder mehrere Oid’s mitgegeben werden, alternativ könnten auch alle fälligen Zahlungen mit dem Status „Open“ bereitgestellt werden.

Das Filterkriterium für die auszuführenden SEPA-Überweisungen wird als Parameter in der URL mitgegeben, siehe nachfolgende Beispiele.

Beispiel zur Bereitstellung einer SEPA-Überweisung über dessen Oid:

```
https://xycompany.ennoxx.cloud/api/SepaCreditTransferPayments/Provide?criteria=Oid%20%3D%20'14b3f837
-806b-4cb4-83d2-057ac54fc131''
```

Beispiel zur Bereitstellung aller fälligen SEPA-Überweisungen anhand Status und Fälligkeitstermin:

```
https://xycompany.ennoxx.cloud/api/SepaCreditTransferPayments/Provide
?criteria=Status%20%3D%20'Open'%20and%20DueTime%20%3C%3D%20'2021-09-21'
```

5.2.2.2 Aufbau der Http-Response

Die Http-Response liefert ein `BasicJobListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:


```
{
  "Objects": [
    {
      "Channel": {
        "Caption": "string",
        "ChannelType": "Communication",
        "Name": "string",
        "Description": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "SpecificDetail": "string",
      "State": "None",
      "Caption": "string",
      "CommunicationDirection": "Send",
      "DueTime": "2021-09-21T08:24:22.297Z",
      "IntervalType": "Single",
      "Interval": 0,
      "LastExecution": "2021-09-21T08:24:22.297Z",
      "PermissionClass": "string",
      "WebUrl": "string",
      "LastLog": {
        "State": "Success",
        "DateTime": "2021-09-21T08:24:22.297Z",
        "Description": "string",
        "Protocol": "string"
      },
      "Files": [
        {
          "OriginalFile": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ContentType": "Undefined",
          "ReportHtml": "string",
          "ReportSummaryHtml": "string",
          "ReportPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ReportSummaryPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "FileHash": {
            "Method": "None",
            "Value": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        }
      ],
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ BasicJobDto	Enthält die angelegten Aufträge
Channel	Kanal	BasicChannelDto	Enthält Basisinformationen des Kanals
SpecificDetail	Spezifische Details	String	Enthält eine zusammenfassende Kurzinformation des Auftrags
State	Status	String	Aktueller Status des Auftrags
Caption	Bezeichnung	String	
CommunicationDirection	Richtung	String	Enthält die Richtung des Auftrags (in diesem Fall: Senden)
DueTime	Ausführungszeitpunkt	DateTime	
IntervalType	Art des Intervalls	String	Einmaliger oder wiederkehrender Auftrag mit dem angegebenen Intervall
Interval	Intervall	Integer	
LastExecution	Letzte Ausführung am	DateTime	Nur belegt bei wiederkehrenden Aufträgen, für die ein letztes Ausführungsdatum festgelegt wurde
PermissionClass	Zugriffsklasse	String	
WebUrl	Web URL	String	Enthält den Weblink mit dem der Auftrag im Web UI angezeigt werden kann
LastLog	Letztes Log	BasicLogDto	Enthält Informationen des letzten Logs
Files	Dateien	Array vom Typ ContextFileDto	Enthält bei Sendeaufträgen die zu übertragenden Nutzdaten (in diesem Fall die erzeugte Zahlungsdatei)
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des erstellten Auftrags, welche für weitere Auswertungen und Methodenaufrufe (z.B. für die Autorisierung) verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall werden im `Objects`-Array die erzeugten Aufträge mit deren Details (z.B. Kanal, Status, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

5.2.2.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `Provide` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var sctPaymentsClient = new SepaCreditTransferPaymentsClient(httpClient);
sctPaymentsClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

// paymentOid sollte bei der Anlage der SEPA-Überweisung gemerkt worden sein
var paymentOid = "14b3f837-806b-4cb4-83d2-057ac54fc131";
var task = sctPaymentsClient.ProvideAsync(string.Format("Oid = '{0}'", paymentOid));
var result = task.Result;
if (result.Result == BasicJobListResultDtoResult.Success)
{
   log.Info(result.Message);
   foreach (var createdJob in result.Objects)
   {
       var oid = createdJob.Oid;
       //...
   }
}
else
{
   throw new Exception(result.Message);
}
```

5.3 Aktivieren des Sendens nach Erstunterschrift (VEU)

Der API-Controller `EbicsJobs` stellt in ennoxx.banking die Methode `ActivateSendAfterFirstSignature` zur Verfügung, mit der man einen EBICS Sendeauftrag ohne (vollständige) Autorisierung in die verteilte elektronische Unterschrift (VEU) der Bank leiten kann. Der Aufruf dieser Methode kann nur für EBICS Sendeaufträge erfolgen, bei denen noch keine Unterschrift geleistet wurde, und aktiviert dadurch die Option „Nach erster Unterschrift senden“. Der Auftrag wird dann in der Folge bereits nach der ersten geleisteten Unterschrift an die Bank gesendet.
Der aktualisierte EBICS-Auftrag wird innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/EbicsJobs/EbicsJobs_ActivateSendAfterFirstSignature

5.3.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/EbicsJobs/ActivateSendAfterFirstSignature
Method	GET
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
URL-Parameter „oid“	Objekt-ID (Oid) des EBICS Sendeauftrags

Die Objekt-ID (Oid) des EBICS Sendeauftrags, der nach der ersten Unterschrift gesendet werden soll, wird als Parameter in der URL mitgegeben, siehe nachfolgendes Beispiel:

```
https://xycompany.ennoxx.cloud/api/EbicsJobs/ActivateSendAfterFirstSignature?oid=c6d927ad-b11d-4c51-80d9-2bc046170b6c'
```

5.3.2 Aufbau der Http-Response

Die Http-Response liefert ein `EbicsJobListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:


```
{
  "Objects": [
    {
      "OrderNumber": "string",
      "OrderType": {
        "Code": "string",
        "Caption": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "EbicsChannel": {
        "EbicsVersion": "EBICS_2_5",
        "SignatureVersion": "A006",
        "HostName": "string",
        "Url": "string",
        "CustomerId": "string",
        "Users": [
          {
            "UserId": "string",
            "InternalUser": {
              "UserName": "string",
              "FirstName": "string",
              "LastName": "string",
              "Oid": "00000000-0000-0000-0000-000000000000",
              "ClassName": "string"
            },
            "AuthorisationType": "None",
            "State": "New",
            "IniLetter": {
              "FileName": "string",
              "FileSize": 0,
              "Data": "string"
            },
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          }
        ],
        "DefaultUser": {
          "UserId": "string",
          "InternalUser": {
            "UserName": "string",
            "FirstName": "string",
            "LastName": "string",
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          },
          "AuthorisationType": "None",
          "State": "New",
          "IniLetter": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "EncryptionMethod": "string",
        "Caption": "string",
        "ChannelType": "Communication",
        "Name": "string",
        "Description": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "EbicsUser": {
        "UserId": "string",
        "InternalUser": {
          "UserName": "string",
          "FirstName": "string",
```

```
        "LastName": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "AuthorisationType": "None",
      "State": "New",
      "IniLetter": {
        "FileName": "string",
        "FileSize": 0,
        "Data": "string"
      },
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    },
    "Signatures": [
      {
        "AuthorisationType": "None",
        "KeyType": "A006",
        "DateTimeSigned": "2021-09-20T12:36:47.820Z",
        "User": {
          "UserId": "string",
          "InternalUser": {
            "UserName": "string",
            "FirstName": "string",
            "LastName": "string",
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          },
          "AuthorisationType": "None",
          "State": "New",
          "IniLetter": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      }
    ],
    "Protocols": [
      {
        "TimeStamp": "2021-09-20T12:36:47.820Z",
        "ActionCode": 0,
        "ActionText": "string",
        "DisplayFile": "string",
        "OrderID": "string",
        "OrderText": "string",
        "OrderType": "string",
        "ReferencedOrderType": "string",
        "ReferencedOrderId": "string",
        "ResultCode": 0,
        "ResultText": "string",
        "Text": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      }
    ],
    "FetchFrom": "2021-09-20T12:36:47.820Z",
    "FetchTo": "2021-09-20T12:36:47.820Z",
    "Channel": {
      "Caption": "string",
      "ChannelType": "Communication",
      "Name": "string",
      "Description": "string",
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    },
    "SpecificDetail": "string",
    "State": "None",
    "Caption": "string",
    "CommunicationDirection": "Send",
    "DueTime": "2021-09-20T12:36:47.820Z",
```

```
      "IntervalType": "Single",
      "Interval": 0,
      "LastExecution": "2021-09-20T12:36:47.820Z",
      "PermissionClass": "string",
      "WebUrl": "string",
      "LastLog": {
        "State": "Success",
        "DateTime": "2021-09-20T12:36:47.820Z",
        "Description": "string",
        "Protocol": "string"
      },
      "Files": [
        {
          "OriginalFile": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ContentType": "Undefined",
          "ReportHtml": "string",
          "ReportSummaryHtml": "string",
          "ReportPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ReportSummaryPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "FileHash": {
            "Method": "None",
            "Value": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        }
      ],
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ EbicsJobDto	Enthält den aktualisierten EBICS Auftrag
OrderNumber	Auftragsnummer	String	Die Auftragsnummer wird bei einer erfolgreichen Ausführung von der Bank vergeben
OrderType	Auftragsart	EbicsOrderTypeDto	Enthält Details zur Auftragsart (z.B. CCT)
EbicsChannel	EBICS Kanal	EbicsChannelDto	Enthalt Details zum EBICS Kanal
EbicsUser	EBICS Benutzer	EbicsUserDto	Enthält Details des EBICS Transportbenutzers
Signatures	Unterschriften	Array vom Typ EbicsSignatureDto	Enthält die geleisteten Unterschriften des Auftrags
Protocols	EBICS Protokolle	Array vom Typ EbicsProtocolDto	Enthält die EBICS Protokolle des Auftrags, nachdem dieser zur Bank gesendet und ein PTK-Auftrag durchgeführt wurde
FetchFrom	Historischer Abruf von	DateTime	Nur belegt bei historischen Datenabrufen
FetchTo	Historischer Abruf bis	DateTime	Nur belegt bei historischen Datenabrufen
Channel	Kanal	BasicChannelDto	Enthält Basisinformationen des Kanals
SpecificDetail	Spezifische Details	String	Enthält eine zusammenfassende Kurzinformation des Auftrags
State	Status	String	Aktueller Status des Auftrags
Caption	Bezeichnung	String	
CommunicationDirection	Richtung	String	Enthält die Richtung des Auftrags (Senden oder Empfangen)
DueTime	Ausführungszeitpunkt	DateTime	
IntervalType	Art des Intervalls	String	Einmaliger oder wiederkehrender Autrag mit dem angegebenen Intervall
Interval	Intervall	Integer	
LastExecution	Letzte Ausführung am	DateTime	Nur belegt bei wiederkehrenden Aufträgen, für die eine letztes Ausführungsdatum festgelegt wurde
PermissionClass	Zugriffsklasse	String	
WebUrl	Web URL	String	Enthält den Weblink mit dem der Auftrag im Web UI angezeigt werden kann
LastLog	Letztes Log	BasicLogDto	Enthält Informationen des letzten Logs
Files	Dateien	Array vom Typ ContextFileDto	Enthält bei Sendeaufträgen die zu übertragenden Nutzdaten
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des EBICS-Auftrags, welche für weitere Auswertungen und Methodenaufrufe verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall wird im `Objects`-Array der aktualisierte EBICS-Auftrag mit dessen Details (z.B. Auftragsart, EBICS-Kanal, Signaturen, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

5.3.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `ActivateSendAfterFirstSignature` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var ebicsJobsClient = new EbicsJobsClient(httpClient);
ebicsJobsClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

// jobOid sollte bei der Anlage des Auftrags gemerkt worden sein
var jobOid = "c6d927ad-b11d-4c51-80d9-2bc046170b6c";
var task = ebicsJobsClient.ActivateSendAfterFirstSignatureAsync(jobOid);
var result = task.Result;
if (result.Result == EbicsJobListResultDtoResult.Success)
{
   log.Info(result.Message);
   var updatedEbicsJob = result.Objects.First();
   //...
}
else
{
   throw new Exception(result.Message);
}
```

5.4 Autorisierung von SEPA-Überweisungen

Der API-Controller `EbicsJobs` stellt für die Autorisierung bzw. Signierung eines EBICS-Auftrags in ennoxx.banking die Methode `Sign` zur Verfügung. Der aktualisierte EBICS-Auftrag wird anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/EbicsJobs/EbicsJobs_Sign

5.4.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/EbicsJobs/Sign
Method	GET
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
URL-Parameter „oid“	Objekt-ID (Oid) des zu unterschreibenden EBICS Auftrags
URL-Parameter „password“	Schlüsselpasswort für die elektronische Unterschrift des angemeldeten Benutzers

Die Objekt-ID (Oid) des zu unterschreibenden EBICS Auftrags und das Schlüsselpasswort für die elektronische Unterschrift (mit dem A006-Unterschriftsschlüssel) des angemeldeten Benutzers werden als Parameter in der URL mitgegeben, siehe nachfolgendes Beispiel:

```
https://xycompany.ennoxx.cloud/api/EbicsJobs/Sign?oid=c6d927ad-b11d-4c51-80d9-
2bc046170b6c&password=test1234
```

5.4.2 Aufbau der Http-Response

Die Http-Response liefert ein `EbicsJobListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:


```
{
  "Objects": [
    {
      "OrderNumber": "string",
      "OrderType": {
        "Code": "string",
        "Caption": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "EbicsChannel": {
        "EbicsVersion": "EBICS_2_5",
        "SignatureVersion": "A006",
        "HostName": "string",
        "Url": "string",
        "CustomerId": "string",
        "Users": [
          {
            "UserId": "string",
            "InternalUser": {
              "UserName": "string",
              "FirstName": "string",
              "LastName": "string",
              "Oid": "00000000-0000-0000-0000-000000000000",
              "ClassName": "string"
            },
            "AuthorisationType": "None",
            "State": "New",
            "IniLetter": {
              "FileName": "string",
              "FileSize": 0,
              "Data": "string"
            },
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          }
        ],
        "DefaultUser": {
          "UserId": "string",
          "InternalUser": {
            "UserName": "string",
            "FirstName": "string",
            "LastName": "string",
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          },
          "AuthorisationType": "None",
          "State": "New",
          "IniLetter": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "EncryptionMethod": "string",
        "Caption": "string",
        "ChannelType": "Communication",
        "Name": "string",
        "Description": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "EbicsUser": {
        "UserId": "string",
        "InternalUser": {
          "UserName": "string",
          "FirstName": "string",
```

```
        "LastName": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "AuthorisationType": "None",
      "State": "New",
      "IniLetter": {
        "FileName": "string",
        "FileSize": 0,
        "Data": "string"
      },
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    },
    "Signatures": [
      {
        "AuthorisationType": "None",
        "KeyType": "A006",
        "DateTimeSigned": "2021-09-20T12:36:47.820Z",
        "User": {
          "UserId": "string",
          "InternalUser": {
            "UserName": "string",
            "FirstName": "string",
            "LastName": "string",
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          },
          "AuthorisationType": "None",
          "State": "New",
          "IniLetter": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      }
    ],
    "Protocols": [
      {
        "TimeStamp": "2021-09-20T12:36:47.820Z",
        "ActionCode": 0,
        "ActionText": "string",
        "DisplayFile": "string",
        "OrderID": "string",
        "OrderText": "string",
        "OrderType": "string",
        "ReferencedOrderType": "string",
        "ReferencedOrderId": "string",
        "ResultCode": 0,
        "ResultText": "string",
        "Text": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      }
    ],
    "FetchFrom": "2021-09-20T12:36:47.820Z",
    "FetchTo": "2021-09-20T12:36:47.820Z",
    "Channel": {
      "Caption": "string",
      "ChannelType": "Communication",
      "Name": "string",
      "Description": "string",
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    },
    "SpecificDetail": "string",
    "State": "None",
    "Caption": "string",
    "CommunicationDirection": "Send",
    "DueTime": "2021-09-20T12:36:47.820Z",
```

```
      "IntervalType": "Single",
      "Interval": 0,
      "LastExecution": "2021-09-20T12:36:47.820Z",
      "PermissionClass": "string",
      "WebUrl": "string",
      "LastLog": {
        "State": "Success",
        "DateTime": "2021-09-20T12:36:47.820Z",
        "Description": "string",
        "Protocol": "string"
      },
      "Files": [
        {
          "OriginalFile": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ContentType": "Undefined",
          "ReportHtml": "string",
          "ReportSummaryHtml": "string",
          "ReportPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ReportSummaryPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "FileHash": {
            "Method": "None",
            "Value": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        }
      ],
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ EbicsJobDto	Enthält den aktualisierten EBICS Auftrag inkl. der oder den Signaturen
OrderNumber	Auftragsnummer	String	Die Auftragsnummer wird bei einer erfolgreichen Ausführung von der Bank vergeben
OrderType	Auftragsart	EbicsOrderTypeDto	Enthält Details zur Auftragsart (z.B. CCT)
EbicsChannel	EBICS Kanal	EbicsChannelDto	Enthalt Details zum EBICS Kanal
EbicsUser	EBICS Benutzer	EbicsUserDto	Enthält Details des EBICS Transportbenutzers
Signatures	Unterschriften	Array vom Typ EbicsSignatureDto	Enthält die geleisteten Unterschriften des Auftrags
Protocols	EBICS Protokolle	Array vom Typ EbicsProtocolDto	Enthält die EBICS Protokolle des Auftrags, nachdem dieser zur Bank gesendet und ein PTK-Auftrag durchgeführt wurde
FetchFrom	Historischer Abruf von	DateTime	Nur belegt bei historischen Datenabrufen
FetchTo	Historischer Abruf bis	DateTime	Nur belegt bei historischen Datenabrufen
Channel	Kanal	BasicChannelDto	Enthält Basisinformationen des Kanals
SpecificDetail	Spezifische Details	String	Enthält eine zusammenfassende Kurzinformation des Auftrags
State	Status	String	Aktueller Status des Auftrags
Caption	Bezeichnung	String	
CommunicationDirection	Richtung	String	Enthält die Richtung des Auftrags (Senden oder Empfangen)
DueTime	Ausführungszeitpunkt	DateTime	
IntervalType	Art des Intervalls	String	Einmaliger oder wiederkehrender Autrag mit dem angegebenen Intervall
Interval	Intervall	Integer	
LastExecution	Letzte Ausführung am	DateTime	Nur belegt bei wiederkehrenden Aufträgen, für die eine letztes Ausführungsdatum festgelegt wurde
PermissionClass	Zugriffsklasse	String	
WebUrl	Web URL	String	Enthält den Weblink mit dem der Auftrag im Web UI angezeigt werden kann
LastLog	Letztes Log	BasicLogDto	Enthält Informationen des letzten Logs
Files	Dateien	Array vom Typ ContextFileDto	Enthält bei Sendeaufträgen die zu übertragenden Nutzdaten
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des EBICS-Auftrags, welche für weitere Auswertungen und Methodenaufrufe verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall wird im `Objects`-Array der aktualisierte EBICS-Auftrag mit dessen Details (z.B. Auftragsart, EBICS-Kanal, Signaturen, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

5.4.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `Sign` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var ebicsJobsClient = new EbicsJobsClient(httpClient);
ebicsJobsClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

// jobOid sollte bei der Anlage des Auftrags gemerkt worden sein
var jobOid = "c6d927ad-b11d-4c51-80d9-2bc046170b6c";
var task = ebicsJobsClient.SignAsync(jobOid, "test1234");
var result = task.Result;
if (result.Result == EbicsJobListResultDtoResult.Success)
{
   log.Info(result.Message);
   foreach (var signature in result.Objects.First().Signatures)
   {
       //...
   }
}
else
{
   throw new Exception(result.Message);
}
```

5.5 Ermittlung des Status eines EBICS-Auftrages mit SEPA-Überweisungen

5.5.1 Variante 1 – Abfrage mittels REST-API

Der API-Controller `EbicsJobs` stellt für die Statusabfrage von EBICS-Aufträgen in ennoxx.banking die Methode `Read` zur Verfügung. Der oder die abgefragten EBICS-Aufträge werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/EbicsJobs/EbicsJobs_Read

5.5.1.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/EbicsJobs
Method	GET
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
Optionaler URL-Parameter „oid“	Optionale Angabe der Objekt-ID (Oid) für die gezielte Abfrage eines EBICS-Auftrags
Optionaler URL-Parameter „criteria“	Optionale Angabe eines benutzerdefinierten Filterkriteriums für die Abfrage mehrerer EBICS-Aufträge in einem Rutsch (z.B. Anhand mehrerer Oids oder Anhand Auftragsart und/oder Fälligkeitstermin)

Die Objekt-ID (Oid) eines bestimmten EBICS Auftrags oder alternativ ein Filterkriterium für die Abfrage mehrerer EBICS-Aufträge in einem Rutsch werden als Parameter in der URL mitgegeben, siehe nachfolgende Beispiele.

Beispiel der gezielten Abfrage mittels Oid:

```
https://xycompany.ennoxx.cloud/api/EbicsJobs?oid=c6d927ad-b11d-4c51-80d9-2bc046170b6c
```

Beispiel für die Abfrage anhand eines Filterkriteriums (Auftragsart und Fälligkeitstermin):

```
https://xycompany.ennoxx.cloud/api/EbicsJobs?criteria=OrderType.Code%20%3D%20'CCT'%20and%20DueTime%20%3E%3D%20'2021-09-21'
```

5.5.1.2 Aufbau der Http-Response

Die Http-Response liefert ein `EbicsJobListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:


```
{
  "Objects": [
    {
      "OrderNumber": "string",
      "OrderType": {
        "Code": "string",
        "Caption": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "EbicsChannel": {
        "EbicsVersion": "EBICS_2_5",
        "SignatureVersion": "A006",
        "HostName": "string",
        "Url": "string",
        "CustomerId": "string",
        "Users": [
          {
            "UserId": "string",
            "InternalUser": {
              "UserName": "string",
              "FirstName": "string",
              "LastName": "string",
              "Oid": "00000000-0000-0000-0000-000000000000",
              "ClassName": "string"
            },
            "AuthorisationType": "None",
            "State": "New",
            "IniLetter": {
              "FileName": "string",
              "FileSize": 0,
              "Data": "string"
            },
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          }
        ],
        "DefaultUser": {
          "UserId": "string",
          "InternalUser": {
            "UserName": "string",
            "FirstName": "string",
            "LastName": "string",
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          },
          "AuthorisationType": "None",
          "State": "New",
          "IniLetter": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "EncryptionMethod": "string",
        "Caption": "string",
        "ChannelType": "Communication",
        "Name": "string",
        "Description": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "EbicsUser": {
        "UserId": "string",
        "InternalUser": {
          "UserName": "string",
          "FirstName": "string",
          "LastName": "string",
```

```
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "AuthorisationType": "None",
      "State": "New",
      "IniLetter": {
        "FileName": "string",
        "FileSize": 0,
        "Data": "string"
      },
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    },
    "Signatures": [
      {
        "AuthorisationType": "None",
        "KeyType": "A006",
        "DateTimeSigned": "2021-09-20T12:36:47.820Z",
        "User": {
          "UserId": "string",
          "InternalUser": {
            "UserName": "string",
            "FirstName": "string",
            "LastName": "string",
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          },
          "AuthorisationType": "None",
          "State": "New",
          "IniLetter": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      }
    ],
    "Protocols": [
      {
        "TimeStamp": "2021-09-20T12:36:47.820Z",
        "ActionCode": 0,
        "ActionText": "string",
        "DisplayFile": "string",
        "OrderID": "string",
        "OrderText": "string",
        "OrderType": "string",
        "ReferencedOrderType": "string",
        "ReferencedOrderId": "string",
        "ResultCode": 0,
        "ResultText": "string",
        "Text": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      }
    ],
    "FetchFrom": "2021-09-20T12:36:47.820Z",
    "FetchTo": "2021-09-20T12:36:47.820Z",
    "Channel": {
      "Caption": "string",
      "ChannelType": "Communication",
      "Name": "string",
      "Description": "string",
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    },
    "SpecificDetail": "string",
    "State": "None",
    "Caption": "string",
    "CommunicationDirection": "Send",
    "DueTime": "2021-09-20T12:36:47.820Z",
    "IntervalType": "Single",
```

```
      "Interval": 0,
      "LastExecution": "2021-09-20T12:36:47.820Z",
      "PermissionClass": "string",
      "WebUrl": "string",
      "LastLog": {
        "State": "Success",
        "DateTime": "2021-09-20T12:36:47.820Z",
        "Description": "string",
        "Protocol": "string"
      },
      "Files": [
        {
          "OriginalFile": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ContentType": "Undefined",
          "ReportHtml": "string",
          "ReportSummaryHtml": "string",
          "ReportPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ReportSummaryPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "FileHash": {
            "Method": "None",
            "Value": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        }
      ],
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ EbicsJobDto	Enthält die angefragten EBICS-Aufträge mit deren aktuellem Status
OrderNumber	Auftragsnummer	String	Die Auftragsnummer wird bei einer erfolgreichen Ausführung von der Bank vergeben
OrderType	Auftragsart	EbicsOrderTypeDto	Enthält Details zur Auftragsart (z.B. CCT)
EbicsChannel	EBICS Kanal	EbicsChannelDto	Enthalt Details zum EBICS Kanal
EbicsUser	EBICS Benutzer	EbicsUserDto	Enthält Details des EBICS Transportbenutzers
Signatures	Unterschriften	Array vom Typ EbicsSignatureDto	Enthält die geleisteten Unterschriften des Auftrags
Protocols	EBICS Protokolle	Array vom Typ EbicsProtocolDto	Enthält die EBICS Protokolle des Auftrags, nachdem dieser zur Bank gesendet und ein PTK-Auftrag durchgeführt wurde
FetchFrom	Historischer Abruf von	DateTime	Nur belegt bei historischen Datenabrufen
FetchTo	Historischer Abruf bis	DateTime	Nur belegt bei historischen Datenabrufen
Channel	Kanal	BasicChannelDto	Enthält Basisinformationen des Kanals
SpecificDetail	Spezifische Details	String	Enthält eine zusammenfassende Kurzinformation des Auftrags
State	Status	String	Aktueller Status des Auftrags
Caption	Bezeichnung	String	
CommunicationDirection	Richtung	String	Enthält die Richtung des Auftrags (Senden oder Empfangen)
DueTime	Ausführungszeitpunkt	DateTime	
IntervalType	Art des Intervalls	String	Einmaliger oder wiederkehrender Autrag mit dem angegebenen Intervall
Interval	Intervall	Integer	
LastExecution	Letzte Ausführung am	DateTime	Nur belegt bei wiederkehrenden Aufträgen, für die ein letztes Ausführungsdatum festgelegt wurde
PermissionClass	Zugriffsklasse	String	
WebUrl	Web URL	String	Enthält den Weblink mit dem der Auftrag im Web UI angezeigt werden kann
LastLog	Letztes Log	BasicLogDto	Enthält Informationen des letzten Logs
Files	Dateien	Array vom Typ ContextFileDto	Enthält bei Sendeaufträgen die zu übertragenden Nutzdaten
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des EBICS-Auftrags, welche für weitere Auswertungen und Methodenaufrufe verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall werden im `Objects`-Array die angefragten EBICS-Aufträge mit deren Details (z.B. Auftragsart, EBICS-Kanal, Status, Signaturen, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

5.5.1.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `Read` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var ebicsJobsClient = new EbicsJobsClient(httpClient);
ebicsJobsClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

// jobOid sollte bei der Anlage des Auftrags gemerkt worden sein
var jobOid = "c6d927ad-b11d-4c51-80d9-2bc046170b6c";
var task = ebicsJobsClient.ReadAsync(jobOid, null);
var result = task.Result;
if (result.Result == EbicsJobListResultDtoResult.Success)
{
   log.Info(result.Message);
   foreach (var ebicsJob in result.Objects)
   {
       var jobState = ebicsJob.State;
       //...
   }
}
else
{
   throw new Exception(result.Message);
}
```

6 Initiierung von SEPA-Lastschriften (SDD)

6.1 Variante 1 – Fertige XML-Datei

Der API-Controller `BasicCommunicationJobs` stellt für die Übermittlung einer XML-Zahlungsdatei in ennoxx.banking die Methode `CreateFromFile` zur Verfügung. Da der Prozess mit dem für SEPA-Überweisungen übereinstimmt, können Details hierzu dem Kapitel 5.1 entnommen werden.

6.1.1 Übermittlung der XML-Zahlungsdatei

Da der Prozess mit dem für SEPA-Überweisungen übereinstimmt, können Details hierzu dem Kapitel 5.1.1 entnommen werden.

6.2 Variante 2 – Einzelzahlungen über ennoxx.banking

6.2.1 Anlage von SEPA-Lastschriften

Der API-Controller `SepaDirectDebitPayments` stellt für die Anlage von neuen SEPA-Lastschriften in ennoxx.banking die Methoden `Create2` (für eine Einzelanlage) sowie `CreateBulk2` (für die Anlage mehrerer SEPA-Lastschriften in einem Rutsch) zur Verfügung. Die in diesem Zuge erzeugten Zahlungen werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/SepaDirectDebitPayments/SepaDirectDebitPayments_Create2

6.2.1.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/SepaDirectDebitPayments/Create2 oder https://xycompany.ennoxx.cloud/api/SepaDirectDebitPayments/CreateBulk2
Method	POST
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json

Die Daten der zu erstellenden SEPA-Lastschrift werden im Http-Request als Datenstrom mitgegeben. Das hierbei zu nutzende `SepaDirectDebitPaymentCreateRequestDto`-Objekt ist eine JSON-Struktur mit dem nachfolgenden Aufbau. Bei Nutzung der Methode `CreateBulk2` wird die gleiche Dto-Struktur in einem Array verwendet.

```
{
  "Amount": 0,
  "RemittanceInformation": "string",
  "DueTime": "2021-09-21T12:19:13.492Z",
  "IntervalType": "Single",
  "Interval": 0,
  "OrderingPartyAccountOidOrIban": "string",
  "MandateOidOrReference": "string",
  "SignatureDate": "2021-09-21T12:19:13.492Z",
  "DirectDebitType": "CORE",
  "MandateValidity": "RECURRING",
  "EndToEndReference": "string",
  "DebtorName": "string",
  "DebtorIban": "string",
  "DebtorBic": "string"
}
```

Feldname	Bezeichnung	Datentyp	Pflichtfeld	Beschreibung
Amount	Betrag	Number	Ja	
RemittanceInformation	Verwendungszweck	String	Nein	
DueTime	Fälligkeitstermin	DateTime	Nein	Sofern nicht angegeben, wird der aktuelle Tag verwendet
IntervalType	Art des Intervalls	String	Nein	Einmalige oder wiederkehrende Zahlung
Interval	Intervall	Integer	Nein	
OrderingPartyAccount OidOrIban	Auftraggeberkonto	String	Ja	Wahlweise die Objekt-ID (Oid) oder die IBAN des Auftraggeberkontos
MandateOidOrReference	SEPA-Mandat	String	Ja	Wahlweise die Objekt-ID (Oid) oder die Mandatsreferenz des SEPA-Mandats
SignatureDate	Unterschriftsdatum	DateTime	Nein	Unterschriftdatum des SEPA-Mandats Muss nur belegt werden, wenn das SEPA-Mandat noch nicht in ennoxx.banking existiert
DirectDebitType	Lastschrifttyp	String	Nein	Lastschrifttyp (Basis- oder Firmenlastschrift) • CORE • B2B Muss nur belegt werden, wenn das SEPA-Mandat noch nicht in ennoxx.banking existiert
MandateValidity	Gültigkeit	String	Nein	Gültigkeit des Mandats (einmalig oder wiederkehrend) • RECURRING • SINGLE Muss nur belegt werden, wenn das SEPA-Mandat noch nicht in ennoxx.banking existiert
EndToEndReference	Ende-Zu-Ende-Referenz	String	Nein	Sofern nicht belegt, wird eine Ende-Zu-Ende-Referenz durch ennoxx.banking erzeugt
DebtorName	Zahlungspflichtiger Name	String	Nein	Muss nur belegt werden, wenn das SEPA-Mandat noch nicht in ennoxx.banking existiert
DebtorIban	Zahlungspflichtiger IBAN	String	Nein	Muss nur belegt werden, wenn das SEPA-Mandat noch nicht in ennoxx.banking existiert
DebtorBic	Zahlungspflichtiger BIC	String	Nein	

6.2.1.2 Aufbau der Http-Response

Die Http-Response liefert ein `SepaDirectDebitPaymentListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:


```
{
  "Objects": [
    {
      "MandateReference": "string",
      "SignatureDate": "2021-09-21T12:19:13.489Z",
      "DirectDebitType": "CORE",
      "MandateValidity": "RECURRING",
      "EndToEndReference": "string",
      "BeneficiaryName": "string",
      "BeneficiaryIban": "string",
      "BeneficiaryBic": "string",
      "Amount": 0,
      "AmountInEur": 0,
      "CurrencyIsoCode": "string",
      "RemittanceInformation": "string",
      "DueTime": "2021-09-21T12:19:13.489Z",
      "IntervalType": "Single",
      "Interval": 0,
      "Status": "Open",
      "Description": "string",
      "OrderingPartyAccount": {
        "Description": "string",
        "Iban": "string",
        "Bic": "string",
        "AccountNumber": "string",
        "NationalBankCode": "string",
        "AccountIdentifier": "string",
        "AccountIdentifier2": "string",
        "AccountIdentifier3": "string",
        "AccountType": "OpenAccount",
        "AccountCategory": "string",
        "CurrencyIsoCode": "string",
        "AccountHolder": {
          "CompanyName": "string",
          "CreditorIdentification": "string",
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "PaymentProperties": [
        "string"
      ],
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ SepaDirectDebit PaymentDto	Enthält die angelegten SEPA-Lastschriften
MandateReference	Mandatsreferenz	String	
SignatureDate	Unterschriftsdatum	DateTime	
DirectDebitType	Lastschrifttyp	String	
MandateValidity	Gültigkeit	String	
EndToEndReference	Ende-Zu-Ende-Referenz	String	
BeneficiaryName	Zahlungspflichtiger Name	String	
BeneficiaryIban	Zahlungspflichtiger IBAN	String	
BeneficiaryBic	Zahlungspflichtiger BIC	String	
Amount	Betrag	Number	
AmountInEur	Betrag in EUR	Number	
CurrencyIsoCode	Währung	String	Bei SEPA-Zahlungen immer „EUR“
RemittanceInformation	Verwendungszweck	String	
DueTime	Fälligkeitstermin	DateTime	
IntervalType	Art des Intervalls	String	
Interval	Intervall	Integer	
Status	Status	String	Enthält den aktuellen Status der Zahlung: • Open (Offen) • Executed (Bereitgestellt) • Template (Vorlage)
Description	Beschreibung	String	
OrderingPartyAccount	Auftraggeberkonto	AccountDto	Enthält Details des Auftraggeberkontos
PaymentProperties	Eigenschaften	Array vom Typ String	Enthält alle der Zahlung zugeordnete Zahlungseigenschaften
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) der erstellten SEPA-Überweisung, welche für weitere Auswertungen und Methodenaufrufe (z.B. das Bereitstellen der Zahlungen - siehe nachfolgendes Kapitel) verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall werden im `Objects`-Array die erzeugten SEPA-Lastschriften mit deren Details (z.B. Mandatsinformationen, Betrag, Verwendungszweck, Fälligkeitstermin, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

6.2.1.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `Create2` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var sddPaymentsClient = new SepaDirectDebitPaymentsClient(httpClient);
sddPaymentsClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var sddPaymentDto = new SepaDirectDebitPaymentCreateRequestDto() { Amount = 12.34,
OrderingPartyAccountOidOrIban = "DE95500800000000001234", MandateOidOrReference = "M-12345",
SignatureDate = new DateTimeOffset(DateTime.Today), DirectDebitType =
SepaDirectDebitPaymentCreateRequestDtoDirectDebitType.CORE, MandateValidity =
SepaDirectDebitPaymentCreateRequestDtoMandateValidity.RECURRING, DebtorName = "Max Mustermann",
DebtorIban = "DE45300500000000333333", RemittanceInformation = "RG-Nr. 12345" };

var task = sddPaymentsClient.Create2Async(sddPaymentDto);
var result = task.Result;
if (result.Result == SepaDirectDebitPaymentListResultDtoResult.Success)
{
   log.Info(result.Message);
   foreach (var createdPayment in result.Objects)
   {
       var oid = createdPayment.Oid;
       //...
   }
}
else
{
   throw new Exception(result.Message);
}
```

6.2.2 Bereitstellen von SEPA-Lastschriften

Der API-Controller `SepaDirectDebitPayments` stellt für die Bereitstellung von SEPA-Lastschriften als Kommunikationsauftrag in ennoxx.banking die Methode `Provide` zur Verfügung. Der oder die in diesem Zuge erzeugten Kommunikationsaufträge werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/SepaDirectDebitPayments/SepaDirectDebitPayments_Provide

6.2.2.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/SepaDirectDebitPayments/Provide
Method	GET
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
URL-Parameter „criteria“	Enthält das Filterkriterium für die auszuführenden SEPA-Lastschriften. Beispielsweise können an dieser Stelle gezielt eine oder mehrere Oid’s mitgegeben werden, alternativ könnten auch alle fälligen Zahlungen mit dem Status „Open“ bereitgestellt werden.

Das Filterkriterium für die auszuführenden SEPA-Lastschriften wird als Parameter in der URL mitgegeben, siehe nachfolgende Beispiele.

Beispiel zur Bereitstellung einer SEPA-Lastschrift über dessen Oid:

```
https://xycompany.ennoxx.cloud/api/SepaDirectDebitPayments/Provide?criteria=Oid%20%3D%20'14b3f837-806b-4cb4-83d2-057ac54fc131'
```

Beispiel zur Bereitstellung aller fälligen SEPA-Lastschriften anhand Status und Fälligkeitstermin:

```
https://xycompany.ennoxx.cloud/api/SepaDirectDebitPayments/Provide
?criteria=Status%20%3D%20'Open'%20and%20DueTime%20%3C%3D%20'2021-09-21'
```

6.2.2.2 Aufbau der Http-Response

Die Http-Response liefert ein `BasicJobListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:


```
{
  "Objects": [
    {
      "Channel": {
        "Caption": "string",
        "ChannelType": "Communication",
        "Name": "string",
        "Description": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "SpecificDetail": "string",
      "State": "None",
      "Caption": "string",
      "CommunicationDirection": "Send",
      "DueTime": "2021-09-21T08:24:22.297Z",
      "IntervalType": "Single",
      "Interval": 0,
      "LastExecution": "2021-09-21T08:24:22.297Z",
      "PermissionClass": "string",
      "WebUrl": "string",
      "LastLog": {
        "State": "Success",
        "DateTime": "2021-09-21T08:24:22.297Z",
        "Description": "string",
        "Protocol": "string"
      },
      "Files": [
        {
          "OriginalFile": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ContentType": "Undefined",
          "ReportHtml": "string",
          "ReportSummaryHtml": "string",
          "ReportPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ReportSummaryPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "FileHash": {
            "Method": "None",
            "Value": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        }
      ],
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ BasicJobDto	Enthält die angelegten Aufträge
Channel	Kanal	BasicChannelDto	Enthält Basisinformationen des Kanals
SpecificDetail	Spezifische Details	String	Enthält eine zusammenfassende Kurzinformation des Auftrags
State	Status	String	Aktueller Status des Auftrags
Caption	Bezeichnung	String	
CommunicationDirection	Richtung	String	Enthält die Richtung des Auftrags (in diesem Fall: Senden)
DueTime	Ausführungszeitpunkt	DateTime	
IntervalType	Art des Intervalls	String	Einmaliger oder wiederkehrender Auftrag mit dem angegebenen Intervall
Interval	Intervall	Integer	
LastExecution	Letzte Ausführung am	DateTime	Nur belegt bei wiederkehrenden Aufträgen, für die eine letztes Ausführungsdatum festgelegt wurde
PermissionClass	Zugriffsklasse	String	
WebUrl	Web URL	String	Enthält den Weblink mit dem der Auftrag im Web UI angezeigt werden kann
LastLog	Letztes Log	BasicLogDto	Enthält Informationen des letzten Logs
Files	Dateien	Array vom Typ ContextFileDto	Enthält bei Sendeaufträgen die zu übertragenden Nutzdaten (in diesem Fall die erzeugte Zahlungsdatei)
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des erstellten Auftrags, welche für weitere Auswertungen und Methodenaufrufe (z.B. für die Autorisierung) verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall werden im `Objects`-Array die erzeugten Aufträge mit deren Details (z.B. Kanal, Status, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

6.2.2.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `Provide` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var sddPaymentsClient = new SepaDirectDebitPaymentsClient(httpClient);
sddPaymentsClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

// paymentOid sollte bei der Anlage der SEPA-Lastschrift gemerkt worden sein
var paymentOid = "14b3f837-806b-4cb4-83d2-057ac54fc131";
var task = sddPaymentsClient.ProvideAsync(string.Format("Oid = '{0}'", paymentOid));
var result = task.Result;
if (result.Result == BasicJobListResultDtoResult.Success)
{
   log.Info(result.Message);
   foreach (var createdJob in result.Objects)
   {
       var oid = createdJob.Oid;
       //...
   }
}
else
{
   throw new Exception(result.Message);
}
```

6.3 Autorisierung von SEPA-Lastschriften

Da der Prozess mit dem für SEPA-Überweisungen übereinstimmt, können Details hierzu dem Kapitel 5.4 entnommen werden.

6.4 Abfrage des Status eines EBICS-Auftrages mit SEPA-Lastschriften

Da der Prozess mit dem für SEPA-Überweisungen übereinstimmt, können Details hierzu dem Kapitel 5.5 entnommen werden.

6.4.1 Variante 1 – Abfrage mittels REST-API

Da der Prozess mit dem für SEPA-Überweisungen übereinstimmt, können Details hierzu dem Kapitel 5.5.1 entnommen werden.

7 Initiierung von Auslandszahlungen (DTAZV)

7.1 Variante 1 – Fertige DTAZV-Datei

Der API-Controller `BasicCommunicationJobs` stellt für die Übermittlung einer DTAZV-Zahlungsdatei in ennoxx.banking die Methode `CreateFromFile` zur Verfügung. Da der Prozess mit dem für SEPA-Überweisungen übereinstimmt, können Details hierzu dem Kapitel 5.1 entnommen werden.

7.1.1 Übermittlung der DTAZV-Zahlungsdatei

Da der Prozess mit dem für SEPA-Überweisungen übereinstimmt, können Details hierzu dem Kapitel 5.1.1 entnommen werden.

7.2 Variante 2 – Einzelzahlungen über ennoxx.banking

7.2.1 Anlage von Auslandszahlungen (DTAZV)

Der API-Controller `ForeignDePayments` stellt für die Anlage von neuen DTAZV Auslandszahlungen in ennoxx.banking die Methoden `Create2` (für eine Einzelanlage) sowie `CreateBulk2` (für die Anlage mehrerer Auslandszahlungen in einem Rutsch) zur Verfügung. Die in diesem Zuge erzeugten Zahlungen werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/ForeignDePayments/ForeignDePayments_Create2

7.2.1.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/ForeignDePayments/Create2 oder https://xycompany.ennoxx.cloud/api/ForeignDePayments/CreateBulk2
Method	POST
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json

Die Daten der zu erstellenden DTAZV Auslandszahlung werden im Http-Request als Datenstrom mitgegeben. Das hierbei zu nutzende `ForeignDePaymentCreateRequestDto`-Objekt ist eine JSON-Struktur mit dem nachfolgenden Aufbau. Bei Nutzung der Methode `CreateBulk2` wird die gleiche Dto-Struktur in einem Array verwendet.

```
{
  "PaymentType": "Standard",
  "Amount": 0,
  "CurrencyIsoCode": "string",
  "RemittanceInformation": "string",
  "DueTime": "2022-03-18T14:38:06.804Z",
  "IntervalType": "Single",
  "Interval": 0,
  "OrderingPartyAccountOidOrIban": "string",
  "BeneficiaryName": "string",
  "BeneficiaryIban": "string",
  "BeneficiaryAccountNumber": "string",
  "BeneficiaryBicOrNationalBankCode": "string",
  "BeneficiaryAddress": "string",
  "BeneficiaryAddressExtension": "string",
  "BeneficiaryCountryIsoCode": "string",
  "BankName1": "string",
  "BankName2": "string",
  "BankAddress": "string",
  "BankAddressExtension": "string",
  "BankCountryIsoCode": "string",
  "ChargesArrangement": "BothParties",
  "ChargesAccountOidOrIban": "string",
  "OrderInformation1": "string",
  "OrderInformation2": "string",
  "NameAndPhoneForInquiries": "string",
  "OwnReference": "string",
  "InstructionKey1": "None",
  "InstructionKey2": "None",
```

```
  "InstructionKey3": "None",
  "InstructionKey4": "None",
  "AdditionalInstructionInformation": "string"
}
```

Feldname	Bezeichnung	Datentyp	Pflichtfeld	Beschreibung
PaymentType	Zahlungsart	String	Nein	Enthält die Zahlungsart: • Standard • SwiftUrgent • CrossBorder • ChequeFreeShipping • ChequeRegisteredLetter • ChequeCourier • ChequeRegisteredLetterCourier • ChequeOrderingPartyFreeShipping • ChequeOrderingPartyRegisteredLetter • ChequeOrderingPartyCourier • ChequeOrderingPartyRegisteredLetterCourier Sofern nicht angegeben, wird Standard verwendet.
Amount	Betrag	Number	Ja	
CurrencyIsoCode	Währung ISO Code	String	Ja	
RemittanceInformation	Verwendungszweck	String	Nein	
DueTime	Fälligkeitstermin	DateTime	Nein	Sofern nicht angegeben, wird der aktuelle Tag verwendet
IntervalType	Art des Intervalls	String	Nein	Einmalige oder wiederkehrende Zahlung
Interval	Intervall	Integer	Nein	
OrderingPartyAccount OidOrIban	Auftraggeberkonto	String	Ja	Wahlweise die Objekt-ID (Oid) oder die IBAN des Auftraggeberkontos. Sofern ein Währungskonto verwendet werden soll (bei Commerzbank), muss dieses mittels Oid angegeben werden.
BeneficiaryName	Empfänger Name	String	Ja	
BeneficiaryIban	Empfänger IBAN	String	Nein	Wird nicht bei Scheckziehungen belegt. In allen anderen Fällen gilt, entweder Empfänger IBAN oder Kontonummer muss belegt sein.
BeneficiaryAccountNumber	Empfänger Kontonummer	String	Nein	Wird nicht bei Scheckziehungen belegt. In allen anderen Fällen gilt, entweder Empfänger IBAN oder Kontonummer muss belegt sein.
BeneficiaryBicOrNational BankCode	Empfänger BIC bzw. Bankleitzahl	String	Nein	Wird nicht bei Scheckziehungen belegt. In allen anderen Fällen gilt, entweder Empfänger BIC oder Bankleitzahl muss angegeben werden.
BeneficiaryAddress	Empfänger Adresse	String	Nein	
BeneficiaryAddress Extension	Empfänger Adresszusatz	String	Nein	
BeneficiaryCountryIsoCode	Empfänger Ländercode	String	Nein	
BankName1	Bank Name 1	String	Nein	
BankName2	Bank Name 2	String	Nein	
BankAddress	Bank Adresse	String	Nein	
BankAddressExtension	Bank Adresszusatz	String	Nein	
BankCountryIsoCode	Bank Ländercode	String	Nein	

ChargesArrangement	Gebührenregelung	String	Nein	Enthält die Gebührenregelung: • BothParties • OrderingParty • CounterParty Sofern nicht angegeben, wird BothParties verwendet.
ChargesAccountOidOrIban	Gebührenkonto	String	Nein	Wahlweise die Objekt-ID (Oid) oder die IBAN des Gebührenkontos. Sofern ein Währungskonto verwendet werden soll (bei Commerzbank), muss dieses mittels Oid angegeben werden. Ein Gebührenkonto muss nur angegeben werden, sofern sich dieses vom Auftraggeberkonto unterscheidet.
OrderInformation1	Ordervermerk	String	Nein	Darf nur bei Scheckziehungen belegt werden.
OrderInformation2	Ordervermerk 2	String	Nein	Darf nur bei Scheckziehungen belegt werden.
NameAndPhoneFor Inquiries	Ansprechpartner für Rückfragen	String	Nein	
OwnReference	Auftraggeberreferenz (T23)	String	Nein	
InstructionKey1	Weisungsschlüssel 1	String	Nein	Enthält einen optionalen Weisungsschlüssel: • None • OnlyByCheque • OnlyUponIdentification • AdviceInstitutionByPhone • AdviceInstitutionByTelecommunication • AdviceBeneficiaryByPhone • AdviceBeneficiaryByTelecommunication • SettlementOfTrade • IntraCompanyPayment • EuePayment Sofern nicht angegeben wird None (=keine Belegung) verwendet.
InstructionKey2	Weisungsschlüssel 2	String	Nein	Siehe Weisungsschlüssel 1
InstructionKey3	Weisungsschlüssel 3	String	Nein	Siehe Weisungsschlüssel 1
InstructionKey4	Weisungsschlüssel 4	String	Nein	Siehe Weisungsschlüssel 1
AdditionalInstuction Information	Zusatzinformationen zum Weisungsschlüssel	String	Nein	

7.2.1.2 Aufbau der Http-Response

Die Http-Response liefert ein `ForeignDePaymentListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:


```
{
  "Objects": [
    {
      "PaymentType": "Standard",
      "BeneficiaryName": "string",
      "BeneficiaryIban": "string",
      "BeneficiaryAccountNumber": "string",
      "BeneficiaryBicOrNationalBankCode": "string",
      "BeneficiaryAddress": "string",
      "BeneficiaryAddressExtension": "string",
      "BeneficiaryCountry": {
        "IsoCode": "string",
        "Caption": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "BankName1": "string",
      "BankName2": "string",
      "BankAddress": "string",
      "BankAddressExtension": "string",
      "BankCountry": {
        "IsoCode": "string",
        "Caption": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "ChargesArrangement": "BothParties",
      "ChargesAccount": {
        "Description": "string",
        "Iban": "string",
        "Bic": "string",
        "AccountNumber": "string",
        "NationalBankCode": "string",
        "AccountIdentifier": "string",
        "AccountIdentifier2": "string",
        "AccountIdentifier3": "string",
        "AccountType": "OpenAccount",
        "AccountCategory": "string",
        "CurrencyIsoCode": "string",
        "AccountHolder": {
          "CompanyName": "string",
          "CreditorIdentification": "string",
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "OrderInformation1": "string",
      "OrderInformation2": "string",
      "NameAndPhoneForInquiries": "string",
      "OwnReference": "string",
      "InstructionKey1": "None",
      "InstructionKey2": "None",
      "InstructionKey3": "None",
      "InstructionKey4": "None",
      "AdditionalInstructionInformation": "string",
      "Amount": 0,
      "AmountInEur": 0,
      "CurrencyIsoCode": "string",
      "RemittanceInformation": "string",
      "DueTime": "2022-03-18T14:38:06.801Z",
      "IntervalType": "Single",
      "Interval": 0,
      "Status": "Open",
      "Description": "string",
      "OrderingPartyAccount": {
        "Description": "string",
        "Iban": "string",
        "Bic": "string",
```

```
        "AccountNumber": "string",
        "NationalBankCode": "string",
        "AccountIdentifier": "string",
        "AccountIdentifier2": "string",
        "AccountIdentifier3": "string",
        "AccountType": "OpenAccount",
        "AccountCategory": "string",
        "CurrencyIsoCode": "string",
        "AccountHolder": {
          "CompanyName": "string",
          "CreditorIdentification": "string",
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "PaymentProperties": [
        "string"
      ],
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```


Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ ForeignDePaymentDto	Enthält die angelegten DTAZV Auslandszahlungen
PaymentType	Zahlungsart	String	
BeneficiaryName	Empfänger Name	String	
BeneficiaryIban	Empfänger IBAN	String	
BeneficiaryAccountNumber	Empfänger Kontonummer	String	
BeneficiaryBicOrNationalBankCode	Empfänger BIC/BLZ	String	
BeneficiaryAddress	Empfänger Adresse	String	
BeneficiaryAddressExtension	Empfänger Adresszusatz	String	
BeneficiaryCountry	Empfänger Land	CountryDto	
BankName1	Bank Name 1	String	
BankName2	Bank Name 2	String	
BankAddress	Bank Adresse	String	
BankAddressExtension	Bank Adresszusatz	String	
BankCountry	Bank Land	CountryDto	
ChargesArrangement	Gebührenregelung	String	
ChargesAccount	Gebührenkonto	AccountDto	Enthält Details des Gebührenkontos
OrderInformation1	Ordervermerk	String	
OrderInformation2	Ordervermerk 2	String	
NameAndPhoneForInquiries	Ansprechpartner für Rückfragen	String	
OwnReference	Auftraggeberreferenz (T23)	String	

InstructionKey1	Weisungsschlüssel 1	String	
InstructionKey1	Weisungsschlüssel 2	String	
InstructionKey1	Weisungsschlüssel 3	String	
InstructionKey1	Weisungsschlüssel 4	String	
AdditionalInstruction Information	Zusatzinformationen zum Weisungsschlüssel	String	
Amount	Betrag	Number	
AmountInEur	Betrag in EUR	Number	
CurrencyIsoCode	Währung	String	
RemittanceInformation	Verwendungszweck	String	
DueTime	Fälligkeitstermin	DateTime	
IntervalType	Art des Intervalls	String	
Interval	Intervall	Integer	
Status	Status	String	Enthält den aktuellen Status der Zahlung: • Open (Offen) • Executed (Bereitgestellt) • Template (Vorlage)
Description	Beschreibung	String	
OrderingPartyAccount	Auftraggeberkonto	AccountDto	Enthält Details des Auftraggeberkontos
PaymentProperties	Eigenschaften	Array vom Typ String	Enthält alle der Zahlung zugeordnete Zahlungseigenschaften
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) der erstellten SEPA-Überweisung, welche für weitere Auswertungen und Methodenaufrufe (z.B. das Bereitstellen der Zahlungen - siehe nachfolgendes Kapitel) verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall werden im `Objects`-Array die erzeugten DTAZV Auslandszahlungen mit deren Details (z.B. Zahlungsart, Gebührenregelung, Weisungsschlüssel, Betrag, Verwendungszweck, Fälligkeitstermin, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

7.2.1.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `Create2` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var dtazvPaymentsClient = new ForeignDePaymentsClient(httpClient);
dtazvPaymentsClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var dtazvPaymentDto = new ForeignDePaymentCreateRequestDto() { Amount = 12.34, CurrencyIsoCode =
"USD", OrderingPartyAccountOidOrIban = "DE95500800000000001234", SignatureDate = new
DateTimeOffset(DateTime.Today), BeneficiaryName = "Max Mustermann", BeneficiaryIban =
"DE45300500000000333333", BeneficiaryBicOrNationalBankCode = "WELADEDDXXX", RemittanceInformation =
"RG-Nr. 12345" };

var task = dtazvPaymentsClient.Create2Async(dtazvPaymentDto);
var result = task.Result;
if (result.Result == ForeignDePaymentListResultDtoResult.Success)
{
   log.Info(result.Message);
   foreach (var createdPayment in result.Objects)
   {
       var oid = createdPayment.Oid;
       //...
   }
}
else
{
   throw new Exception(result.Message);
}
```

7.2.2 Bereitstellen von Auslandszahlungen (DTAZV)

Der API-Controller `ForeignDePayments` stellt für die Bereitstellung von DTAZV Auslandszahlungen als Kommunikationsauftrag in ennoxx.banking die Methode `Provide` zur Verfügung. Der oder die in diesem Zuge erzeugten Kommunikationsaufträge werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/ForeignDePayments/ForeignDePayments_Provide

7.2.2.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/ForeignDePayments/Provide
Method	GET
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
URL-Parameter „criteria“	Enthält das Filterkriterium für die auszuführenden DTAZV Auslandszahlungen. Beispielsweise können an dieser Stelle gezielt eine oder mehrere Oid’s mitgegeben werden, alternativ könnten auch alle fälligen Zahlungen mit dem Status „Open“ bereitgestellt werden.

Das Filterkriterium für die auszuführenden Auslandszahlungen wird als Parameter in der URL mitgegeben, siehe nachfolgende Beispiele.

Beispiel zur Bereitstellung einer Auslandszahlung über dessen Oid:

```
https://xycompany.ennoxx.cloud/api/ForeignDePayments/Provide?criteria=Oid%20%3D%20'ae112dba-461e-4ea2-94cb-a8371a958136'
```

Beispiel zur Bereitstellung aller fälligen Auslandszahlungen anhand Status und Fälligkeitstermin:

```
https://xycompany.ennoxx.cloud/api/ForeignDePayments/Provide
?criteria=Status%20%3D%20'Open'%20and%20DueTime%20%3C%3D%20'2022-03-18'
```

7.2.2.2 Aufbau der Http-Response

Die Http-Response liefert ein `BasicJobListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:


```
{
  "Objects": [
    {
      "Channel": {
        "Caption": "string",
        "ChannelType": "Communication",
        "Name": "string",
        "Description": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "SpecificDetail": "string",
      "State": "None",
      "Caption": "string",
      "CommunicationDirection": "Send",
      "DueTime": "2021-09-21T08:24:22.297Z",
      "IntervalType": "Single",
      "Interval": 0,
      "LastExecution": "2021-09-21T08:24:22.297Z",
      "PermissionClass": "string",
      "WebUrl": "string",
      "LastLog": {
        "State": "Success",
        "DateTime": "2021-09-21T08:24:22.297Z",
        "Description": "string",
        "Protocol": "string"
      },
      "Files": [
        {
          "OriginalFile": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ContentType": "Undefined",
          "ReportHtml": "string",
          "ReportSummaryHtml": "string",
          "ReportPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ReportSummaryPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "FileHash": {
            "Method": "None",
            "Value": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        }
      ],
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ BasicJobDto	Enthält die angelegten Aufträge
Channel	Kanal	BasicChannelDto	Enthält Basisinformationen des Kanals
SpecificDetail	Spezifische Details	String	Enthält eine zusammenfassende Kurzinformation des Auftrags
State	Status	String	Aktueller Status des Auftrags
Caption	Bezeichnung	String	
CommunicationDirection	Richtung	String	Enthält die Richtung des Auftrags (in diesem Fall: Senden)
DueTime	Ausführungszeitpunkt	DateTime	
IntervalType	Art des Intervalls	String	Einmaliger oder wiederkehrender Auftrag mit dem angegebenen Intervall
Interval	Intervall	Integer	
LastExecution	Letzte Ausführung am	DateTime	Nur belegt bei wiederkehrenden Aufträgen, für die eine letztes Ausführungsdatum festgelegt wurde
PermissionClass	Zugriffsklasse	String	
WebUrl	Web URL	String	Enthält den Weblink mit dem der Auftrag im Web UI angezeigt werden kann
LastLog	Letztes Log	BasicLogDto	Enthält Informationen des letzten Logs
Files	Dateien	Array vom Typ ContextFileDto	Enthält bei Sendeaufträgen die zu übertragenden Nutzdaten (in diesem Fall die erzeugte Zahlungsdatei)
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des erstellten Auftrags, welche für weitere Auswertungen und Methodenaufrufe (z.B. für die Autorisierung) verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall werden im `Objects`-Array die erzeugten Aufträge mit deren Details (z.B. Kanal, Status, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

7.2.2.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `Provide` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var dtazvPaymentsClient = new ForeignDePaymentsClient(httpClient);
dtazvPaymentsClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

// paymentOid sollte bei der Anlage der DTAZV Auslandszahlung gemerkt worden sein
var paymentOid = "ae112dba-461e-4ea2-94cb-a8371a958136";
var task = dtazvPaymentsClient.ProvideAsync(string.Format("Oid = '{0}'", paymentOid));
var result = task.Result;
if (result.Result == BasicJobListResultDtoResult.Success)
{
   log.Info(result.Message);
   foreach (var createdJob in result.Objects)
   {
       var oid = createdJob.Oid;
       //...
   }
}
else
{
   throw new Exception(result.Message);
}
```

7.3 Autorisierung von Auslandszahlungen (DTAZV)

Da der Prozess mit dem für SEPA-Überweisungen übereinstimmt, können Details hierzu dem Kapitel 5.4 entnommen werden.

7.4 Abfrage des Status eines EBICS-Auftrages mit Auslandszahlungen

Da der Prozess mit dem für SEPA-Überweisungen übereinstimmt, können Details hierzu dem Kapitel 5.5 entnommen werden.

7.4.1 Variante 1 – Abfrage mittels REST-API

Da der Prozess mit dem für SEPA-Überweisungen übereinstimmt, können Details hierzu dem Kapitel 5.5.1 entnommen werden.

8 Abholung von Kontoinformationen

8.1 Regelmäßige Abholung von Kontoinformationen von der Bank

Für eine regelmäßige, automatisierte Abholung von Kontoinformationen von der Bank muss für jeden EBICS Zugang einmalig ein wiederkehrender EBICS Abholauftrag (mit der Auftragsart STA für MT940 oder C53 für camt.053) eingestellt werden.
Der API-Controller `EbicsJobs` stellt für die Erzeugung eines EBICS-Abholauftrags in ennoxx.banking die Methode `CreateFetchJob` zur Verfügung. Der in diesem Zuge erzeugte EBICS-Auftrag wird anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/EbicsJobs/EbicsJobs_CreateFetchJob

8.1.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/EbicsJobs/CreateFetchJob
Method	GET
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
URL-Parameter „channelOidOrCaption“	Wahlweise die Objekt-ID (Oid) oder die Bezeichnung des EBICS-Kanals
URL-Parameter „orderTypeOidOrCode“	Wahlweise die Objekt-ID (Oid) oder der Code der zu nutzenden Auftragsart (z.B. „STA“ für MT940)
Optionaler URL-Parameter „permissionClass“	Enthält den Namen der Zugriffsklasse, sofern diese dem Auftrag für einen eingeschränkten Zugriff zugeordnet werden soll
Optionaler URL-Parameter „dueTime“	Enthält Datum und Uhrzeit der Ausführung
Optionaler URL-Parameter „intervalType“	Enthält die Art des Ausführungsintervalls. Zu empfehlen ist eine tägliche Ausführung mit dem Wert „Dayly“
Optionaler URL-Parameter „interval“	Enthält das Intervall für die Ausführung. Zu empfehlen ist der Wert 1 für eine tägliche Ausführung
Optionaler URL-Parameter „repeatOnError“	Gibt an, ob bei Auftreten eines Fehlers der Auftrag auf das nächste Intervall gesetzt werden soll. Empfohlen: True

Die Objekt-ID (Oid) oder die Bezeichnung des EBICS Kanals, die Objekt-ID (Oid) oder der Code der zu nutzenden Auftragsart sowie zusätzliche Informationen bzgl. des Ausführungstermins werden als Parameter in der URL mitgegeben, siehe nachfolgendes Beispiel.

```
https://xycompany.ennoxx.cloud/api/EbicsJobs/CreateFetchJob?channelOidOrCaption=EBICS%20Testbank%20XY%20Company%20GmbH&orderTypeOidOrCode=STA&dueTime=2021-09-23%2006%3A00%3A00&intervalType=Dayly&interval=1&repeatOnError=true
```

8.1.2 Aufbau der Http-Response

Die Http-Response liefert ein `EbicsJobListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:


```
{
  "Objects": [
    {
      "OrderNumber": "string",
      "OrderType": {
        "Code": "string",
        "Caption": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "EbicsChannel": {
        "EbicsVersion": "EBICS_2_5",
        "SignatureVersion": "A006",
        "HostName": "string",
        "Url": "string",
        "CustomerId": "string",
        "Users": [
          {
            "UserId": "string",
            "InternalUser": {
              "UserName": "string",
              "FirstName": "string",
              "LastName": "string",
              "Oid": "00000000-0000-0000-0000-000000000000",
              "ClassName": "string"
            },
            "AuthorisationType": "None",
            "State": "New",
            "IniLetter": {
              "FileName": "string",
              "FileSize": 0,
              "Data": "string"
            },
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          }
        ],
        "DefaultUser": {
          "UserId": "string",
          "InternalUser": {
            "UserName": "string",
            "FirstName": "string",
            "LastName": "string",
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          },
          "AuthorisationType": "None",
          "State": "New",
          "IniLetter": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "EncryptionMethod": "string",
        "Caption": "string",
        "ChannelType": "Communication",
        "Name": "string",
        "Description": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "EbicsUser": {
        "UserId": "string",
        "InternalUser": {
          "UserName": "string",
          "FirstName": "string",
```

```
        "LastName": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "AuthorisationType": "None",
      "State": "New",
      "IniLetter": {
        "FileName": "string",
        "FileSize": 0,
        "Data": "string"
      },
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    },
    "Signatures": [
      {
        "AuthorisationType": "None",
        "KeyType": "A006",
        "DateTimeSigned": "2021-09-20T12:36:47.820Z",
        "User": {
          "UserId": "string",
          "InternalUser": {
            "UserName": "string",
            "FirstName": "string",
            "LastName": "string",
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          },
          "AuthorisationType": "None",
          "State": "New",
          "IniLetter": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      }
    ],
    "Protocols": [
      {
        "TimeStamp": "2021-09-20T12:36:47.820Z",
        "ActionCode": 0,
        "ActionText": "string",
        "DisplayFile": "string",
        "OrderID": "string",
        "OrderText": "string",
        "OrderType": "string",
        "ReferencedOrderType": "string",
        "ReferencedOrderId": "string",
        "ResultCode": 0,
        "ResultText": "string",
        "Text": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      }
    ],
    "FetchFrom": "2021-09-20T12:36:47.820Z",
    "FetchTo": "2021-09-20T12:36:47.820Z",
    "Channel": {
      "Caption": "string",
      "ChannelType": "Communication",
      "Name": "string",
      "Description": "string",
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    },
    "SpecificDetail": "string",
    "State": "None",
    "Caption": "string",
    "CommunicationDirection": "Send",
    "DueTime": "2021-09-20T12:36:47.820Z",
```

```
      "IntervalType": "Single",
      "Interval": 0,
      "LastExecution": "2021-09-20T12:36:47.820Z",
      "PermissionClass": "string",
      "WebUrl": "string",
      "LastLog": {
        "State": "Success",
        "DateTime": "2021-09-20T12:36:47.820Z",
        "Description": "string",
        "Protocol": "string"
      },
      "Files": [
        {
          "OriginalFile": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ContentType": "Undefined",
          "ReportHtml": "string",
          "ReportSummaryHtml": "string",
          "ReportPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ReportSummaryPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "FileHash": {
            "Method": "None",
            "Value": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        }
      ],
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ EbicsJobDto	Enthält den angelegten EBICS-Auftrag
OrderNumber	Auftragsnummer	String	Die Auftragsnummer wird bei einer erfolgreichen Ausführung von der Bank vergeben
OrderType	Auftragsart	EbicsOrderTypeDto	Enthält Details zur Auftragsart
EbicsChannel	EBICS Kanal	EbicsChannelDto	Enthalt Details zum EBICS Kanal
EbicsUser	EBICS Benutzer	EbicsUserDto	Enthält Details des EBICS Transportbenutzers
Signatures	Unterschriften	Array vom Typ EbicsSignatureDto	Enthält die geleisteten Unterschriften des Auftrags (bei Abholaufträgen ist diese Liste leer)
Protocols	EBICS Protokolle	Array vom Typ EbicsProtocolDto	Enthält die EBICS Protokolle des Auftrags, nachdem dieser zur Bank gesendet und ein PTK-Auftrag durchgeführt wurde
FetchFrom	Historischer Abruf von	DateTime	Nur belegt bei historischen Datenabrufen
FetchTo	Historischer Abruf bis	DateTime	Nur belegt bei historischen Datenabrufen
Channel	Kanal	BasicChannelDto	Enthält Basisinformationen des Kanals
SpecificDetail	Spezifische Details	String	Enthält eine zusammenfassende Kurzinformation des Auftrags
State	Status	String	Aktueller Status des Auftrags
Caption	Bezeichnung	String	
CommunicationDirection	Richtung	String	Enthält die Richtung des Auftrags (Empfangen)
DueTime	Ausführungszeitpunkt	DateTime	
IntervalType	Art des Intervalls	String	Einmaliger oder wiederkehrender Auftrag mit dem angegebenen Intervall
Interval	Intervall	Integer	
LastExecution	Letzte Ausführung am	DateTime	Nur belegt bei wiederkehrenden Aufträgen, für die ein letztes Ausführungsdatum festgelegt wurde
PermissionClass	Zugriffsklasse	String	
WebUrl	Web URL	String	Enthält den Weblink mit dem der Auftrag im Web UI angezeigt werden kann
LastLog	Letztes Log	BasicLogDto	Enthält Informationen des letzten Logs
Files	Dateien	Array vom Typ ContextFileDto	Enthält bei Sendeaufträgen die zu übertragenden Nutzdaten
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des erstellten EBICS-Auftrags, welche für weitere Auswertungen und Methodenaufrufe verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall wird im `Objects`-Array der erzeugte EBICS-Auftrag mit dessen Details (z.B. Auftragsart, EBICS-Kanal, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

8.1.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `CreateFetchJob` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var ebicsJobsClient = new EbicsJobsClient(httpClient);
ebicsChannelsClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var task = ebicsJobsClient.CreateFetchJobAsync("EBICS Testbank XY Company GmbH", "STA", null, new
DateTimeOffset(new DateTime(2021, 9, 23, 6, 0, 0)), IntervalType.Dayly, 1, true);
var result = task.Result;
if (result.Result == EbicsJobListResultDtoResult.Success)
{
   log.Info(result.Message);
   var oid = result.Objects.First().Oid;
   //...
}
else
{
   throw new Exception(result.Message);
}
```

8.2 Variante 1 – Originaldatei der Bank

Für den Abruf der täglich von der Bank übermittelten Originaldatei (im MT940- bzw. camt.053-Format) mittels der REST-API kann die Nutzung von sogenannten FileExchange Aufträgen zum Einsatz kommen. Über eine zentrale Verarbeitungsregel in ennoxx.banking kann konfiguriert werden, dass die Dateien von allen erfolgreichen Kontoauszugs-Abrufen über einen FileExchange-Kanal mit einer fest hinterlegten System-ID (z.B. „MT940“) für einen passiven Abruf mittels der REST-API zur Verfügung gestellt werden. Die Dateien können bzw. müssen dann solange nacheinander mittels der nachfolgend beschriebenen Methoden abgerufen werden, bis keine Datei mehr zur Abholung zur Verfügung steht.

8.2.1 Per FileExchange bereitgestellte Dateien abrufen

Der API-Controller `FileExchangeJobs` stellt für den Abruf bereitgestellter Dateien in ennoxx.banking die Methode `GetNextContextFiles` zur Verfügung. Die in diesem Zuge abgerufenen Dateien werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/FileExchangeJobs/FileExchangeJobs_GetNextContextFiles

8.2.1.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/FileExchangeJobs/GetNextContextFiles
Method	GET
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
URL-Parameter „systemId“	Enthält die System-ID des FileExchange-Kanals, für den die Dateien abgerufen werden sollen
Optionaler URL-Parameter „reportHtml“	Sofern mit True belegt, wird zusätzlich zu der Originaldatei ein von ennoxx.banking aufbereiteter detaillierter Dateiinhalt im HTML-Format in der Html-Response ausgegeben
Optionaler URL-Parameter „reportSummaryHtml“	Sofern mit True belegt, wird zusätzlich zu der Originaldatei ein von ennoxx.banking aufbereiteter Dateiinhalt im HTML-Format in der Html-Response ausgegeben
Optionaler URL-Parameter „reportPdf“	Sofern mit True belegt, wird zusätzlich zu der Originaldatei ein von ennoxx.banking aufbereiteter detaillierter Dateiinhalt im PDF-Format in der Html-Response ausgegeben
Optionaler URL-Parameter „reportSummaryPdf“	Sofern mit True belegt, wird zusätzlich zu der Originaldatei ein von ennoxx.banking aufbereiteter Dateiinhalt im PDF-Format in der Html-Response ausgegeben
Optionaler URL-Parameter „confirmReceipt“	Sofern mit True belegt, wird der Empfang der Daten direkt bestätigt. Ein erneuter Abruf ist dann nicht mehr möglich. Empfohlen wird, den Empfang besser in einem separaten Schritt zu bestätigen – siehe Kapitel 8.2.2

Die System-ID des FileExchange-Kanals sowie zusätzliche Parameter für den Abruf von zusätzlichen Dateiinhalts-Aufbereitungen aus ennoxx.banking werden als Parameter in der URL mitgegeben, siehe nachfolgendes Beispiel.

```
https://xycompany.ennoxx.cloud/api/FileExchangeJobs/GetNextContextFiles?systemId=MT940&reportHtml=true
```

8.2.1.2 Aufbau der Http-Response

Die Http-Response liefert ein `FileExchangeContextFileListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:


```
{
  "FileExchangeJob": {
    "FileExchangeChannel": {
      "SystemId": "string",
      "EncryptionMethod": "string",
      "Caption": "string",
      "ChannelType": "Communication",
      "Name": "string",
      "Description": "string",
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    },
    "Channel": {
      "Caption": "string",
      "ChannelType": "Communication",
      "Name": "string",
      "Description": "string",
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    },
    "SpecificDetail": "string",
    "State": "None",
    "Caption": "string",
    "CommunicationDirection": "Send",
    "DueTime": "2021-09-21T12:19:13.399Z",
    "IntervalType": "Single",
    "Interval": 0,
    "LastExecution": "2021-09-21T12:19:13.399Z",
    "PermissionClass": "string",
    "WebUrl": "string",
    "LastLog": {
      "State": "Success",
      "DateTime": "2021-09-21T12:19:13.399Z",
      "Description": "string",
      "Protocol": "string"
    },
    "Files": [
      {
        "OriginalFile": {
          "FileName": "string",
          "FileSize": 0,
          "Data": "string"
        },
        "ContentType": "Undefined",
        "ReportHtml": "string",
        "ReportSummaryHtml": "string",
        "ReportPdf": {
          "FileName": "string",
          "FileSize": 0,
          "Data": "string"
        },
        "ReportSummaryPdf": {
          "FileName": "string",
          "FileSize": 0,
          "Data": "string"
        },
        "FileHash": {
          "Method": "None",
          "Value": "string"
        },
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      }
    ],
    "Oid": "00000000-0000-0000-0000-000000000000",
    "ClassName": "string"
  },
  "Objects": [
    {
      "OriginalFile": {
```

```
        "FileName": "string",
        "FileSize": 0,
        "Data": "string"
      },
      "ContentType": "Undefined",
      "ReportHtml": "string",
      "ReportSummaryHtml": "string",
      "ReportPdf": {
        "FileName": "string",
        "FileSize": 0,
        "Data": "string"
      },
      "ReportSummaryPdf": {
        "FileName": "string",
        "FileSize": 0,
        "Data": "string"
      },
      "FileHash": {
        "Method": "None",
        "Value": "string"
      },
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
FileExchangeJob	FileExchange Auftrag	FileExchangeJobDto	Enthält Informationen des verknüpften FileExchange Auftrags (z.B. Kanal, Status, Oid, etc.) Die hier enthaltene Oid wird für die nachfolgende Bestätigung des Datenabrufs benötigt.
Objects	Objekte	Array vom Typ ContextFileDto	Enthält die angeforderten Dateiinformationen
OriginalFile	Originaldatei	FileDto	Enthält den Dateinamen, die Größe in Bytes und den Dateiinhalt als Base64-kodiertes Byte-Array
ContentType	Dateityp	String	Enthält den analysierten Dateityp (z.B. MT940)
ReportHtml	Detaillierter Dateiinhalt	String	Enthält den aufbereiteten detaillierten Dateiinhalt im HTML-Format, sofern im Request angefragt
ReportSummaryHtml	Dateiinhalt	String	Enthält den aufbereiteten Dateiinhalt im HTML-Format, sofern im Request angefragt
ReportPdf	PDF Dateiinhalt	FileDto	Enthält den Dateinamen, die Größe in Bytes und den PDF Dateiinhalt als Base64-kodiertes Byte-Array
ReportSummaryPdf	PDF Detaillierter Dateiinhalt	FileDto	Enthält den Dateinamen, die Größe in Bytes und den detaillierten PDF Dateiinhalt als Base64-kodiertes Byte-Array
FileHash	Hashwert	FileHashDto	Enthält den in ennoxx.banking berechneten Hashwert über die Originaldatei und die verwendete Hashmethode, sofern konfiguriert
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des ContextFiles. Kann ignoriert werden, da dieser nicht benötigt wird.
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall werden im `Objects`-Array die bereitgestellte Orignaldatei sowie die zusätzlich angeforderten Daten zurückgegeben. Im Feld `FileExchangeJob` befindet sich zudem der verknüpfte FileExchange Auftrag, dessen Objekt-ID (Oid) für eine nachfolgende Bestätigung des Datenabrufs benötigt wird. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleiben das `Objects`-Array und auch das Feld `FileExchangeJob` hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen. Sofern keine weiteren Daten zur Abholung vorhanden sind, wird im `Result` der Wert „Warning“ und im `DetailResult` der Wert „NoDataAvailable“ ausgegeben.

8.2.1.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `GetNextContextFiles` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var fileExchangeJobsClient = new FileExchangeJobsClient(httpClient);
fileExchangeJobsClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var task = fileExchangeJobsClient.GetNextContextFilesAsync("MT940", true, false, false, false,
false);
var result = task.Result;
if (result.Result == FileExchangeContextFileListResultDtoResult.Success)
{
   log.Info(result.Message);
   var fileExchangeJobOid = result.FileExchangeJob.Oid;
   foreach (var contextFile in result.Objects)
   {
       var fileName = contextFile.OriginalFile.FileName;
       var fileContent = contextFile.OriginalFile.Data;
       var reportHtml = contextFile.ReportHtml;
       //...
   }
}
else
{
   throw new Exception(result.Message);
}
```

8.2.2 Erfolgreichen Abruf der Dateien bestätigen

Der API-Controller `FileExchangeJobs` stellt für die Bestätigung eines erfolgreichen Datenabrufs in ennoxx.banking die Methode `ConfirmReceipt` zur Verfügung. Der aktualisierte FileExchange Auftrag wird anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/FileExchangeJobs/FileExchangeJobs_ConfirmReceipt

8.2.2.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/FileExchangeJobs/ConfirmReceipt
Method	GET
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
URL-Parameter „oid“	Enthält die Objekt-ID (Oid) des FileExchange Auftrags, für den die Dateien erfolgreich abgerufen wurden

Die Objekt-ID des FileExchange Auftrags für den die Datenabholung bestätigt werden soll wird als Parameter in der URL mitgegeben, siehe nachfolgendes Beispiel.

```
https://xycompany.ennoxx.cloud/api/FileExchangeJobs/ConfirmReceipt?oid=c6d927ad-b11d-4c51-80d9-2bc046170b6c
```

8.2.2.2 Aufbau der Http-Response

Die Http-Response liefert ein `FileExchangeJobListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:


```
{
  "Objects": [
    {
      "FileExchangeChannel": {
        "SystemId": "string",
        "EncryptionMethod": "string",
        "Caption": "string",
        "ChannelType": "Communication",
        "Name": "string",
        "Description": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "Channel": {
        "Caption": "string",
        "ChannelType": "Communication",
        "Name": "string",
        "Description": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "SpecificDetail": "string",
      "State": "None",
      "Caption": "string",
      "CommunicationDirection": "Send",
      "DueTime": "2021-09-21T12:19:13.406Z",
      "IntervalType": "Single",
      "Interval": 0,
      "LastExecution": "2021-09-21T12:19:13.406Z",
      "PermissionClass": "string",
      "WebUrl": "string",
      "LastLog": {
        "State": "Success",
        "DateTime": "2021-09-21T12:19:13.406Z",
        "Description": "string",
        "Protocol": "string"
      },
      "Files": [
        {
          "OriginalFile": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ContentType": "Undefined",
          "ReportHtml": "string",
          "ReportSummaryHtml": "string",
          "ReportPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ReportSummaryPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "FileHash": {
            "Method": "None",
            "Value": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        }
      ],
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
```

```
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ FileExchangeJobDto	Enthält Informationen zum aktualisierten FileExchange Auftrag (Kanal, Status, etc.). Der Status des Auftrags sollte von „WaitForFetchExecution“ auf „Successful“ gewechselt worden sein.
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall wird im `Objects`-Array der aktualisierte FileExchange Auftrag zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

8.2.2.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `ConfirmReceipt` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var fileExchangeJobsClient = new FileExchangeJobsClient(httpClient);
fileExchangeJobsClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

// fileExchangeJobOid sollte beim Datenabruf gemerkt worden sein
var fileExchangeJobOid = "c6d927ad-b11d-4c51-80d9-2bc046170b6c";
var task = fileExchangeJobsClient.ConfirmReceiptAsync(fileExchangeJobOid);
var result = task.Result;
if (result.Result == FileExchangeJobListResultDtoResult.Success)
{
   log.Info(result.Message);
   //...
}
else
{
   throw new Exception(result.Message);
}
```

8.3 Variante 2 – Einzeltransaktionen abrufen

Der API-Controller `AccountingTransactions` stellt für den Abruf von in ennoxx.banking importierten Kontoumsätzen die Methode `Read` zur Verfügung. Die angeforderten Kontoumsätze werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/AccountingTransactions/AccountingTransactions_Read

8.3.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/AccountingTransactions
Method	GET
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
Optionaler URL-Parameter „oid“	Enthält die Objekt-ID (Oid) eines bestimmten Kontoumsatzes
Optionaler URL-Parameter „iban“	Enthält die IBAN des verknüpften Kontos der abzurufenden Kontumsätze
Optionaler URL-Parameter „transactionDate“	Enthält das Buchungsdatum der abzurufenden Kontoumsätze
Optionaler URL-Parameter „valueDate“	Enthält das Valutadatum der abzurufenden Kontoumsätze
Optionaler URL-Parameter „criteria“	Enthält ein benutzerdefiniertes Filterkriterium für die abzurufenden Kontoumsätze (z.B. Filterung anhand des Betrags oder anhand des Verwendungszwecks)

Die Objekt-ID eines gezielten Kontoumsatzes oder die IBAN, das Buchungsdatum, das Valutadatum bzw. ein benutzerdefiniertes Filterkriterium können als Parameter in der URL mitgegeben werden, siehe nachfolgende Beispiele. Die letzten vier Parameter können dabei auch gleichzeitig verwendet werden und werden in diesem Fall mittels eines UND-Operators verknüpft.

Beispiel eines Kontoumsatz-Abrufs über dessen Oid:

```
https://xycompany.ennoxx.cloud/api/AccountingTransactions?oid=c6d927ad-b11d-4c51-80d9-2bc046170b6c
```

Beispiel eines Kontoumsatz-Abrufs über verschiedene Parameter:

```
https://xycompany.ennoxx.cloud/api/AccountingTransactions?iban=DE95500800000000001234&transactionDate=2021-09-06
```

8.3.2 Aufbau der Http-Response

Die Http-Response liefert ein `AccountingTransactionListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:


```
{
  "Objects": [
    {
      "AccountStatement": {
        "OpeningBalance": 0,
        "OpeningBalanceInEur": 0,
        "OpeningBalanceDate": "2021-09-21T12:19:13.005Z",
        "ClosingBalance": 0,
        "ClosingBalanceInEur": 0,
        "ClosingBalanceDate": "2021-09-21T12:19:13.005Z",
        "StatementNumber": 0,
        "MessageId": "string",
        "Transactions": [
          {}
        ],
        "Account": {
          "Description": "string",
          "Iban": "string",
          "Bic": "string",
          "AccountNumber": "string",
          "NationalBankCode": "string",
          "AccountIdentifier": "string",
          "AccountIdentifier2": "string",
          "AccountIdentifier3": "string",
          "AccountType": "OpenAccount",
          "AccountCategory": "string",
          "CurrencyIsoCode": "string",
          "AccountHolder": {
            "CompanyName": "string",
            "CreditorIdentification": "string",
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "Amount": 0,
      "AmountInEur": 0,
      "Currency": "string",
      "Gvc": "string",
      "TransactionCode": "string",
      "TransactionText": "string",
      "PrimaNota": "string",
      "RemittanceInformation": "string",
      "TransactionDate": "2021-09-21T12:19:13.005Z",
      "ValueDate": "2021-09-21T12:19:13.005Z",
      "TextKeyExtension": "string",
      "CustomerReference": "string",
      "BankReference": "string",
      "EndToEndReference": "string",
      "MandateReference": "string",
      "CreditorId": "string",
      "PaymentPartnerName": "string",
      "PaymentPartnerIban": "string",
      "PaymentPartnerBic": "string",
      "Note": "string",
      "TransactionProperties": [
        "string"
      ],
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
```

```
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ Accounting-TransactionDto	Enthält alle angeforderten Kontoumsätze
AccountStatement	Kontoauszug	AccountStatementDto	Enthält Informationen zum verknüpften Kontoauszug (z.B. Auszugsnummer, Startsaldo, Endsaldo, etc.) sowie zum verknüpften Konto
Amount	Betrag	Number	
AmountInEur	Betrag (EUR)	Number	Enthält den in EUR umgerechneten Betrag, sofern Währungskurse abgerufen wurden
Currency	Währung	String	
Gvc	GVC	String	Enthält den Geschäftsvorfallcode
TransactionCode	Textschlüssel	String	
TransactionText	Buchungstext	String	
PrimaNota	Primanota	String	
RemittanceInformation	Verwendungszweck	String	
TransactionDate	Buchungsdatum	DateTime	
ValueDate	Valutadatum	DateTime	
TextKeyExtension	Textschlüssel-Ergänzung	String	
CustomerReference	Kundenreferenz	String	
BankReference	Bankreferenz	String	
EndToEndReference	Ende-Zu-Ende-Referenz	String	
MandateReference	Mandatsreferenz	String	
CreditorId	Gläubigeridentifikation	String	
PaymentPartnerName	Zahlungspartner Name	String	
PaymentPartnerIban	Zahlungspartner IBAN	String	Enthält ggf. die Kontonummer
PaymentPartnerBic	Zahlungspartner BIC	String	Enthält ggf. die Bankleitzahl
Note	Bemerkung	String	
TransactionProperties	Eigenschaften	Array vom Typ String	Enthält alle dem Kontoumsatz zugeordneten Umsatzeigenschaften
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des Kontoumsatzes
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall werden im `Objects`-Array alle angeforderten Kontoumsätze zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

8.3.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `Read` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var accountingTransactionsClient = new AccountingTransactionsClient(httpClient);
accountingTransactionsClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var task = accountingTransactionsClient. ReadAsync(null, "DE95500800000000001234", new
DateTimeOffset(new DateTime(2021, 9, 6)), null, null);
var result = task.Result;
if (result.Result == AccountingTransactionListResultDtoResult.Success)
{
   log.Info(result.Message);
   foreach (var transaction in result.Objects)
   {
       var amount = transaction.Amount;
       var remittanceInformation = transaction.RemittanceInformation;
       //...
   }
}
else
{
   throw new Exception(result.Message);
}
```

9 Cash Forecasts (Finanzplandaten)

9.1 Anlage von Cash Forecasts

Der API-Controller `CashForecasts` stellt für die Anlage von manuellen Cash Forecasts in ennoxx.banking die Methoden `Create2` (für eine Einzelanlage) sowie `CreateBulk2` (für die Anlage mehrerer Cash Forecasts in einem Rutsch) zur Verfügung. Die in diesem Zuge erzeugten Cash Forecasts werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/CashForecasts/CashForecasts_Create2

9.1.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/CashForecasts/Create2 oder https://xycompany.ennoxx.cloud/api/CashForecasts/CreateBulk2
Method	POST
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json

Die Daten des Cash Forecasts werden im Http-Request als Datenstrom mitgegeben. Das hierbei zu nutzende `CashForecastCreateRequestDto`-Objekt ist eine JSON-Struktur mit dem nachfolgenden Aufbau. Bei Nutzung der Methode `CreateBulk2` wird die gleiche Dto-Struktur in einem Array verwendet.

```
{
  "Caption": "string",
  "AccountOidOrIban": "string",
  "AccountCurrency": "string",
  "CompanyOidOrName": "string",
  "DueTime": "2022-04-07T00:18:17.592Z",
  "Amount": 0,
  "IntervalType": "Single",
  "Interval": 0,
  "LastExecution": "2022-04-07T00:18:17.592Z",
  "TransactionPropertyOidOrCaption": "string",
  "ScenarioOidOrCaption": "string",
  "PermissionClassOidOrCaption": "string"
}
```

Feldname	Bezeichnung	Datentyp	Pflichtfeld	Beschreibung
Caption	Bezeichnung	String	Nein	
AccountOidOrIban	Konto	String	Nein	Wahlweise die Objekt-ID (Oid) oder die IBAN des zu nutzenden Kontos
AccountCurrency	Kontowährung	String	Nein	Währung des zu nutzenden Kontos Wird bei der Kontoermittlung verwendet, sofern das Konto mittels IBAN angegeben wurde. Sofern nicht angegeben, wird automatisch „EUR“ verwendet.
CompanyOidOrName	Geschäftspartner	String	Nein	Wahlweise die Objekt-ID (Oid) oder der Name des Geschäftspartners
DueTime	Fälligkeitstermin	DateTime	Nein	Bei Angabe eines Fälligkeitstermins in der Vergangenheit oder bei keiner Angabe wird automatisch das aktuelle Datum verwendet
Amount	Betrag	Number	Nein	
IntervalType	Art des Intervalls	String	Nein	Die Art des Intervalls: • Single • Hourly • Dayly • Workdayly • Weekly • Monthly • Quarterly • SemiAnually • Anually
Interval	Interval	Integer	Nein	
LastExecution	Nicht ausführen nach	DateTime	Nein	Letzte Ausführung des Cash Forecast (Stop des Intervalls)
TransactionPropertyOidOrCaption	Umsatz Eigenschaft	String	Nein	Wahlweise die Objekt-ID (Oid) oder die Bezeichnung der zu verknüpfenden Umsatzeigenschaft
ScenarioOidOrCaption	Szenario	String	Nein	Wahlweise die Objekt-ID (Oid) oder die Bezeichnung des zu verknüpfenden Szenarios
PermissionClassOidOrCaption	Zugriffsklasse	String	Nein	Wahlweise die Objekt-ID (Oid) oder die Bezeichnung des zu verknüpfenden Zugriffsklasse

9.1.2 Aufbau der Http-Response

Die Http-Response liefert ein `CashForecastListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:


```
{
  "Objects": [
    {
      "Account": {
        "Description": "string",
        "Iban": "string",
        "Bic": "string",
        "AccountNumber": "string",
        "NationalBankCode": "string",
        "AccountIdentifier": "string",
        "AccountIdentifier2": "string",
        "AccountIdentifier3": "string",
        "AccountType": "OpenAccount",
        "AccountCategory": "string",
        "CurrencyIsoCode": "string",
        "AccountHolder": {
          "CompanyName": "string",
          "CreditorIdentification": "string",
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "Amount": 0,
      "AmountInEur": 0,
      "Caption": "string",
      "DueTime": "2022-04-07T00:18:17.580Z",
      "IntervalType": "Single",
      "Interval": 0,
      "LastExecution": "2022-04-07T00:18:17.580Z",
      "Company": "string",
      "Origin": "Manual",
      "Scenario": "string",
      "ForecastTransactionSet": {
        "TransactionProperty": "string",
        "CreateForecastTransactions": true,
        "Type": "string",
        "OriginalOid": "00000000-0000-0000-0000-000000000000",
        "ForecastTransactions": [
          {
            "Account": {
              "Description": "string",
              "Iban": "string",
              "Bic": "string",
              "AccountNumber": "string",
              "NationalBankCode": "string",
              "AccountIdentifier": "string",
              "AccountIdentifier2": "string",
              "AccountIdentifier3": "string",
              "AccountType": "OpenAccount",
              "AccountCategory": "string",
              "CurrencyIsoCode": "string",
              "AccountHolder": {
                "CompanyName": "string",
                "CreditorIdentification": "string",
                "Oid": "00000000-0000-0000-0000-000000000000",
                "ClassName": "string"
              },
              "Oid": "00000000-0000-0000-0000-000000000000",
              "ClassName": "string"
            },
            "Date": "2022-04-07T00:18:17.580Z",
            "Amount": 0,
            "AmountInEur": 0,
            "Company": "string",
            "TransactionProperty": "string",
            "Type": "string",
            "OriginalOid": "00000000-0000-0000-0000-000000000000",
```

```
            "Scenario": "string",
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          }
        ],
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```


Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ CashForecastDto	Enthält die erzeugten Cash Forecasts
Account	Konto	AccountDto	Enthält die Daten des verknüpften Kontos
Amount	Betrag	Number	
AmountInEur	Betrag (EUR)	Number	Enthält den in EUR umgerechneten Betrag, sofern Währungskurse abgerufen wurden
Caption	Bezeichnung	String	
DueTime	Fälligkeitstermin	DateTime	
IntervalType	Art des Intervalls	String	Die Art des Intervalls: • Single • Hourly • Dayly • Workdayly • Weekly • Monthly • Quarterly • SemiAnually • Anually
Interval	Interval	String	
LastExecution	Nicht ausführen nach	DateTime	
Company	Geschäftspartner	String	
Origin	Herkunft	String	Herkunft des Cash Forecasts: • Manual • Automatic
Scenario	Szenario	String	
ForecastTransactionSet	Forecast Transaktionsset	ForecastTransactionSetDto	Enthält die zum Cash Forecast gehörigen Forecast Transaktionen
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des erstellten Cash Forecasts, welche für weitere Auswertungen und Methodenaufrufe verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis

DetailResult	Detailliertes Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall werden im `Objects`-Array die erzeugten Cash Forecasts mit deren Details (z.B. Konto, Betrag, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

9.1.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `Create2` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var cashForecastClient = new CashForecastsClient(httpClient);
cashForecastClient.BaseUrl = httpClientCustomer.BaseAddress.AbsoluteUri;

var cashForecastDto = new CashForecastCreateRequestDto()
{
   Caption = "CashForecast",
   AccountOidOrIban = "DE95500800000000001234",
   AccountCurrency = "EUR",
   Amount = 156.65,
   CompanyOidOrName = "Test Company GmbH",
   IntervalType = CashForecastCreateRequestDtoIntervalType.Single,
   Interval = 0,
   DueTime = new DateTimeOffset(DateTime.Now),
   ScenarioOidOrCaption = "Test Scenario",
   TransactionPropertyOidOrCaption = "Test Transaction Property",
   PermissionClassOidOrCaption = "PermissionClass1"
};
var task = cashForecastClient.Create2Async(cashForecastDto);
var result = task.Result;
if (result.Result == CashForecastListResultDtoResult.Success)
{
   log.Info(result.Message);
   foreach (var createCashForecast in result.Objects)
   {
      var oid = createCashForecast.Oid;
      //...
   }
}
else
{
   throw new Exception(result.Message);
}
```

9.2 Cash Forecasts abrufen

Der API-Controller `CashForecasts` stellt für den Abruf von in ennoxx.banking vorhandenen Cash Forecasts die Methode `Read` zur Verfügung. Die abgefragten Cash Forecasts werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/CashForecasts/CashForecasts_Read

9.2.1.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/CashForecasts
Method	GET
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
Optionaler URL-Parameter „oid“	Optionale Angabe der Objekt-ID (Oid) für die gezielte Abfrage eines Cash Forecasts
Optionaler URL-Parameter „iban“	Optionale Angabe der Konto-IBAN für die Abfrage mehrerer Cash Forecasts in einem Rutsch
Optionaler URL-Parameter „date“	Optionale Angabe des Fälligkeitstermins für die Abfrage mehrerer Cash Forecasts in einem Rutsch
Optionaler URL-Parameter „criteria“	Optionale Angabe eines benutzerdefinierten Filterkriteriums für die Abfrage mehrerer Cash Forecasts in einem Rutsch (z.B. Anhand mehrerer Oids oder Anhand Betrag und/oder Fälligkeitstermin)

Die Objekt-ID (Oid) eines bestimmten Cash Forecasts oder alternativ Konto-IBAN, Fälligkeitstermin, ein Filterkriterium für die Abfrage mehrerer Cash Forecasts in einem Rutsch werden als Parameter in der URL mitgegeben, siehe nachfolgende Beispiele.

Beispiel der gezielten Abfrage mittels Oid:

```
https://xycompany.ennoxx.cloud/api/CashForecasts?oid=c6d927ad-b11d-4c51-80d9-2bc046170b6c
```

Beispiel für die Abfrage anhand der Konto-IBAN:

```
https://xycompany.ennoxx.cloud/api/CashForecasts?iban=DE23%208466%206215%200080%202154%2023'
```

Beispiel für die Abfrage anhand des Fälligkeitstermins:

```
https://xycompany.ennoxx.cloud/api/CashForecasts?date=2022-04-07'
```

Beispiel für die Abfrage anhand eines Filterkriteriums (Betrag und Interval):

```
https://xycompany.ennoxx.cloud/CashForecasts?criteria=Amount%20%3D%3D%20'152'%20%26%26%20IntervalType%20%3D%3D%20'Single''
```

9.2.1.2 Aufbau der Http-Response

Die Http-Response liefert ein `CashForecastListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:


```
{
  "Objects": [
    {
      "Account": {
        "Description": "string",
        "Iban": "string",
        "Bic": "string",
        "AccountNumber": "string",
        "NationalBankCode": "string",
        "AccountIdentifier": "string",
        "AccountIdentifier2": "string",
        "AccountIdentifier3": "string",
        "AccountType": "OpenAccount",
        "AccountCategory": "string",
        "CurrencyIsoCode": "string",
        "AccountHolder": {
          "CompanyName": "string",
          "CreditorIdentification": "string",
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "Amount": 0,
      "AmountInEur": 0,
      "Caption": "string",
      "DueTime": "2022-04-07T13:52:49.715Z",
      "IntervalType": "Single",
      "Interval": 0,
      "LastExecution": "2022-04-07T13:52:49.715Z",
      "Company": "string",
      "Origin": "Manual",
      "Scenario": "string",
      "ForecastTransactionSet": {
        "TransactionProperty": "string",
        "CreateForecastTransactions": true,
        "Type": "string",
        "OriginalOid": "00000000-0000-0000-0000-000000000000",
        "ForecastTransactions": [
          {
            "Account": {
              "Description": "string",
              "Iban": "string",
              "Bic": "string",
              "AccountNumber": "string",
              "NationalBankCode": "string",
              "AccountIdentifier": "string",
              "AccountIdentifier2": "string",
              "AccountIdentifier3": "string",
              "AccountType": "OpenAccount",
              "AccountCategory": "string",
              "CurrencyIsoCode": "string",
              "AccountHolder": {
                "CompanyName": "string",
                "CreditorIdentification": "string",
                "Oid": "00000000-0000-0000-0000-000000000000",
                "ClassName": "string"
              },
              "Oid": "00000000-0000-0000-0000-000000000000",
              "ClassName": "string"
            },
            "Date": "2022-04-07T13:52:49.715Z",
            "Amount": 0,
            "AmountInEur": 0,
            "Company": "string",
            "TransactionProperty": "string",
            "Type": "string",
            "OriginalOid": "00000000-0000-0000-0000-000000000000",
            "Scenario": "string",
```

```
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          }
        ],
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```


Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ CashForecastDto	Enthält die angefragten Cash Forecasts
Account	Konto	AccountDto	
Amount	Betrag	Number	
AmountInEur	Betrag (EUR)	Number	Enthält den in EUR umgerechneten Betrag, sofern Währungskurse abgerufen wurden
Caption	Bezeichnung	String	
DueTime	Fälligkeitstermin	DateTime	
IntervalType	Art des Intervalls	String	Die Art des Intervalls: • Single • Hourly • Dayly • Workdayly • Weekly • Monthly • Quarterly • SemiAnually • Anually
Interval	Interval	String	
LastExecution	Fälligkeitstermin	DateTime	Letzte Ausführung des Cash Forecast (Stop des Intervalls)
Company	Geschäftspartner	String	
Origin	Herkunft	String	Herkunft des Cash Forecasts: • Manual • Automatic
Scenario	Szenario	String	
ForecastTransactionSet	Forecast Transaktionsset	ForecastTransactionSetDto	Enthält alle zum Cash Forecast gehörigen Cash Forecast Transaktionen
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des angefragten Cash Forecasts, welche für weitere Auswertungen und Methodenaufrufe verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis

DetailResult	Detailliertes Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall werden im `Objects`-Array die angefragten Cash Forecasts mit deren Details (z.B. Konto, Betrag, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

9.2.1.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `Read` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var cashForecastClient = new CashForecastsClient(httpClient);
cashForecastClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

// forecastOid sollte bei der Anlage des Cash Forecasts gemerkt worden sein
var forecastOid = "c6d927ad-b11d-4c51-80d9-2bc046170b6c";
var task = cashForecastClient.ReadAsync(forecastOid, null, null, null);
var result = task.Result;
if (result.Result == CashForecastListResultDtoResult.Success)
{
   log.Info(result.Message);
   foreach (var cashForecast in result.Objects)
   {
       var amount = cashForecast.Amount;
       //...
   }
}
else
{
   throw new Exception(result.Message);
}
```

9.3 Forecast Transaktionen abrufen

Der API-Controller `CashForecasts` stellt für den Abruf von in ennoxx.banking vorhandenen Forecast Transaktionen die Methode `Read` zur Verfügung. Der oder die abgefragten Forecast Transaktionen werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/CashForecasts/CashForecasts_ReadTransactions

9.3.1.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/CashForecasts/ForecastTransactions
Method	GET
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
Optionaler URL-Parameter „oid“	Optionale Angabe der Objekt-ID (Oid) für die gezielte Abfrage einer Forecast Transaktion
Optionaler URL-Parameter „iban“	Optionale Angabe der Konto-IBAN für die Abfrage mehrerer Forecast Transaktionen in einem Rutsch
Optionaler URL-Parameter „date“	Optionale Angabe des Datum für die Abfrage mehrerer Forecast Transaktionen in einem Rutsch
Optionaler URL-Parameter „criteria“	Optionale Angabe eines benutzerdefinierten Filterkriteriums für die Abfrage mehrerer Forecast Transaktionen in einem Rutsch (z.B. Anhand mehrerer Oids oder Anhand Betrag und/oder Fälligkeitstermin)

Die Objekt-ID (Oid) einer bestimmten Forecast Transaktion oder alternativ Konto-IBAN, Fälligkeitstermin, ein Filterkriterium für die Abfrage mehrerer Forecast Transaktionen in einem Rutsch werden als Parameter in der URL mitgegeben, siehe nachfolgende Beispiele.

Beispiel der gezielten Abfrage mittels Oid:

```
https://xycompany.ennoxx.cloud/api/CashForecasts/ForecastTransactions?oid=c6d927ad-b11d-4c51-80d9-2bc046170b6c
```

Beispiel für die Abfrage anhand der Konto-IBAN:

```
https://xycompany.ennoxx.cloud/api/CashForecasts/ForecastTransactions?iban=DE23%208466%206215%200080%202154%2023'
```

Beispiel für die Abfrage anhand des Fälligkeitstermins:

```
https://xycompany.ennoxx.cloud/api/CashForecasts/ForecastTransactions?date=2022-04-07'
```

Beispiel für die Abfrage anhand eines Filterkriteriums (Betrag und Datum):

```
https://xycompany.ennoxx.cloud/CashForecasts/ForecastTransactions?
criteria=Amount%20%3D%3D%20'152'%20%26%26%20Date%20%3D%3D%20'2022-04-07'
```

9.3.1.2 Aufbau der Http-Response

Die Http-Response liefert ein `ForecastTransactionListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:


```
{
  "Objects": [
    {
      "Account": {
        "Description": "string",
        "Iban": "string",
        "Bic": "string",
        "AccountNumber": "string",
        "NationalBankCode": "string",
        "AccountIdentifier": "string",
        "AccountIdentifier2": "string",
        "AccountIdentifier3": "string",
        "AccountType": "OpenAccount",
        "AccountCategory": "string",
        "CurrencyIsoCode": "string",
        "AccountHolder": {
          "CompanyName": "string",
          "CreditorIdentification": "string",
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "Date": "2022-04-07T13:52:49.721Z",
      "Amount": 0,
      "AmountInEur": 0,
      "Company": "string",
      "TransactionProperty": "string",
      "Type": "string",
      "OriginalOid": "00000000-0000-0000-0000-000000000000",
      "Scenario": "string",
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```


Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ ForecastTransactionDto	Enthält die angefragten Forecast Transaktionen
Account	Konto	String	
Date	Datum	DateTime	
Amount	Betrag	String	
AmountInEur	Betrag (EUR)	String	Enthält den in EUR umgerechneten Betrag, sofern Währungskurse abgerufen wurden
Company	Geschäftspartner	String	
TransactionProperty	Umsatz Eigenschaft	String	
Type	Typ	String	Vollqualifizierter Klassenname des zugrundeliegenden Datentyps der Forecast Transaktion
OriginalOid	Originale OID	String	Enthält die originale eindeutige Objekt-ID (Guid) des zugrundeliegenden Datentyps der Forecast Transaktion, welche für weitere Auswertungen und Methodenaufrufe verwendet werden kann

Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) der angefragten Forecast Transaktion, welche für weitere Auswertungen und Methodenaufrufe verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Scenario	Szenario	String	Die Bezeichnung des hinterlegten Szenarios
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error Warning
Message	Nachricht	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Detailliertes Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall werden im `Objects`-Array angefragten Forecast Transaktion mit deren Details (z.B. Konto, Betrag, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

9.3.1.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `Read` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var cashForecastClient = new CashForecastsClient(httpClient);
cashForecastClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var task = cashForecastClient.ReadTransactionsAsync(null, null, null, null);
var result = task.Result;
if (result.Result == ForecastTransactionListResultDtoResult.Success)
{
   log.Info(result.Message);
   foreach (var forecastTransaction in result.Objects)
   {
       var amount = forecastTransaction.Amount;
       //...
   }
}
else
{
   throw new Exception(result.Message);
}
```

10 On-Boarding

10.1 Anlage Unternehmen

Der API-Controller `Identities` stellt für die Anlage von neuen Unternehmen in ennoxx.banking die Methoden `Create2` (für eine Einzelanlage) sowie `CreateBulk2` (für die Anlage mehrerer Unternehmen in einem Rutsch) zur Verfügung. Die in diesem Zuge erzeugten Unternehmen werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/Identities/Identities_Create2

10.1.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/Identities/Create2 oder https://xycompany.ennoxx.cloud/api/Identities/CreateBulk2
Method	POST
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json

Die Daten des Unternehmens werden im Http-Request als Datenstrom mitgegeben. Das hierbei zu nutzende `IdentityCreateRequestDto`-Objekt ist eine JSON-Struktur mit dem nachfolgenden Aufbau. Bei Nutzung der Methode `CreateBulk2` wird die gleiche Dto-Struktur in einem Array verwendet.

```
{
  "CompanyName": "string",
  "CreditorIdentification": "string"
}
```

Feldname	Bezeichnung	Datentyp	Pflichtfeld	Beschreibung
CompanyName	Firmenname	String	Ja	
CreditorIdentification	Gläubigeridentifikation	String	Nein	Für SEPA Lastschriften muss die Gläubigeridentifikation belegt sein

10.1.2 Aufbau der Http-Response

Die Http-Response liefert ein `IdentityListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:

```
{
  "Objects": [
    {
      "CompanyName": "string",
      "CreditorIdentification": "string",
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ IdentityDto	Enthält die erzeugten Unternehmen
CompanyName	Firmenname	String	
CreditorIdentification	Gläubigeridentifikation	String	Für SEPA Lastschriften muss die Gläubigeridentifikation belegt sein
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des erstellten Unternehmens, welche für weitere Auswertungen und Methodenaufrufe verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall werden im `Objects`-Array die erzeugten Unternehmen mit deren Details (z.B. Unternehmensnamen, Gläubigeridentifikation, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

10.1.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `Create2` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var identitiesClient = new IdentitiesClient(httpClient);
identitiesClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var identityDto = new IdentityCreateRequestDto() { CompanyName = "XY Company GmbH",
CreditorIdentification = "DE98ZZZ09999999999" };
var task = identitiesClient.Create2Async(identityDto);
var result = task.Result;
if (result.Result == IdentityListResultDtoResult.Success)
{
   log.Info(result.Message);
   foreach (var createdIdentity in result.Objects)
   {
       var oid = createdIdentity.Oid;
       //...
   }
}
else
{
   throw new Exception(result.Message);
}
```

10.2 Anlage Benutzer

Der API-Controller `Users` stellt für die Anlage von neuen Benutzern in ennoxx.banking die Methoden `Create2` (für eine Einzelanlage) sowie `CreateBulk2` (für die Anlage mehrerer Benutzer in einem Rutsch) zur Verfügung. Die in diesem Zuge erzeugten Benutzer werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/Users/Users_Create2

10.2.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/Users/Create2 oder https://xycompany.ennoxx.cloud/api/Users/CreateBulk2
Method	POST
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json

Die Daten des Benutzers werden im Http-Request als Datenstrom mitgegeben. Das hierbei zu nutzende `UserCreateRequestDto`-Objekt ist eine JSON-Struktur mit dem nachfolgenden Aufbau. Bei Nutzung der Methode `CreateBulk2` wird die gleiche Dto-Struktur in einem Array verwendet.

```
{
  "UserName": "string",
  "FirstName": "string",
  "LastName": "string",
  "InitialPassword": "string",
  "WebUi": true,
  "Api": true,
  "ApiLoginCredentialType": "UsernameAndPasswordOrAuthToken",
  "OData": true,
  "ODataLoginCredentialType": "UsernameAndPasswordOrAuthToken"
}
```

Feldname	Bezeichnung	Datentyp	Pflichtfeld	Beschreibung
UserName	Benutzername	String	Ja	Enthält das Logon, welches für die Anmeldung mit Benutzernamen und Passwort benötigt wird
FirstName	Vorname	String	Nein	
LastName	Nachname	String	Nein	
InitialPassword	Initialpasswort	String	Nein	Sofern nicht angegeben, wird durch ennoxx.banking automatisch ein Initialpasswort für die Erstanmeldung erzeugt
WebUi	WebUI	Boolean	Nein	Setzt den Anwendungszugriff des Benutzers für das WebUI Standardbelegung: True
Api	API	Boolean	Nein	Setzt den Anwendungszugriff des Benutzers für die API Standardbelegung: False
ApiLoginCredentialType	Anmeldetyp API	String	Nein	Legt den Anmeldetyp des Benutzers für die API fest Standardbelegung: „UserNameAndPasswordOr AuthToken“
OData	OData	Boolean	Nein	Setzt den Anwendungszugriff des Benutzers für den OData Service Standardbelegung: False
ODataLoginCredential Type	Anmeldetyp OData	String	Nein	Legt den Anmeldetyp des Benutzers für den OData Service fest Standardbelegung: „UserNameAndPasswordOr AuthToken“

10.2.2 Aufbau der Http-Response

Die Http-Response liefert ein `ExtendedUserListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:

```
{
  "Objects": [
    {
      "InitialPassword": "string",
      "Roles": [
        {
          "Name": "string",
          "IsAdministrative": true,
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        }
      ],
      "UserName": "string",
      "FirstName": "string",
      "LastName": "string",
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ ExtendedUserDto	Enthält die erzeugten Benutzer
InitialPassword	Initialpasswort	String	Initialpasswort für die Erstanmeldung des Benutzers. Das Passwort muss anschließend einmalig durch den Benutzer geändert werden.
Roles	Rollen	Array vom Typ RolesDto	Auflistung aller dem Benutzer zugeordneten Benutzerrollen
UserName	Benutzername	String	Enthält das Logon, welches für die Anmeldung mit Benutzernamen und Passwort benötigt wird
FirstName	Vorname	String	
LastName	Nachname	String	
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des erstellten Benutzers, welche für weitere Auswertungen und Methodenaufrufe verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall werden im `Objects`-Array die erzeugten Benutzer mit deren Details (z.B. Benutzernamen, Rollen, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

10.2.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `Create2` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var usersClient = new UsersClient(httpClient);
usersClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var userDto = new UserCreateRequestDto() { UserName = "testuser1", FirstName = "Test", LastName =
"User1", WebUi = true, Api = true };
var task = usersClient.Create2Async(userDto);
var result = task.Result;
if (result.Result == ExtendedUserListResultDtoResult.Success)
{
   log.Info(result.Message);
   foreach (var createdUser in result.Objects)
   {
       var oid = createdUser.Oid;
       //...
   }
}
else
{
   throw new Exception(result.Message);
}
```

10.3 Passwortänderung eines Benutzers

Der API-Controller `Users` stellt für die eigene Passwortänderung eines Benutzers in ennoxx.banking die Methode `ChangePassword` zur Verfügung. Das Ergebnis sowie die aktualisierten Benutzerdaten werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/Users/Users_ChangePassword

10.3.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/Users/ChangePassword
Method	GET
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
URL-Parameter „newPassword“	Angabe des neuen Passworts

Das neue Passwort des Benutzers wird als Parameter in der URL mitgegeben, siehe nachfolgendes Beispiel. Zu beachten ist, dass das neue Passwort den in ennoxx.banking hinterlegten Passwortrichtlinien entsprechen muss, da es sonst zu einem Fehler führt.

```
https://xycompany.ennoxx.cloud/api/Users/ChangePassword?newPassword=test1234
```

10.3.2 Aufbau der Http-Response

Die Http-Response liefert ein `ExtendedUserListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:

```
{
  "Objects": [
    {
      "InitialPassword": "string",
      "Roles": [
        {
          "Name": "string",
          "IsAdministrative": true,
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        }
      ],
      "UserName": "string",
      "FirstName": "string",
      "LastName": "string",
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ ExtendedUserDto	Enthält den aktualisierten Benutzer
InitialPassword	Initialpasswort	String	Ein vorhandenens Initialpasswort wird nach der erfolgreichen Passwortänderung gelöscht
Roles	Rollen	Array vom Typ RolesDto	Auflistung aller dem Benutzer zugeordneten Benutzerrollen
UserName	Benutzername	String	Enthält das Logon, welches für die Anmeldung mit Benutzernamen und Passwort benötigt wird
FirstName	Vorname	String	
LastName	Nachname	String	
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des erstellten Benutzers, welche für weitere Auswertungen und Methodenaufrufe verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall wird im `Objects`-Array der aktualisierte Benutzer mit dessen Details (z.B. Benutzernamen, Rollen, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

10.3.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `ChangePassword` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var userName = "testuser1";
var password = "password1";
var credentials = new NetworkCredential(userName, password);
var handler = new HttpClientHandler { Credentials = credentials };
var httpClient = new HttpClient(handler);
httpClient.BaseAddress = new Uri(url);

var usersClient = new UsersClient(httpClient);
usersClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var task = usersClient.ChangePasswordAsync("test1234");
var result = task.Result;
if (result.Result == ExtendedUserListResultDtoResult.Success)
{
   log.Info(result.Message);
   var oid = result.Objects.First().Oid;
   //...
}
else
{
   throw new Exception(result.Message);
}
```

10.4 Zuweisung von Benutzerrollen zu einem Benutzer

Der API-Controller `Users` stellt für die Zuweisung von Benutzerrollen zu einem Benutzer in ennoxx.banking die Methode `AssignRole` zur Verfügung. Das Ergebnis sowie die aktualisierten Benutzerdaten werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/Users/Users_AssignRole

10.4.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/Users/AssignRole
Method	GET
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
URL-Parameter „oidOrUserName“	Wahlweise die Objekt-ID (Oid) oder der Benutzername des Benutzers, dem die Benutzerrolle hinzugefügt werden soll
URL-Parameter „roleOidOrName“	Wahlweise die Objekt-ID (Oid) oder der Name der hinzuzufügenden Benutzerrolle

Die Oid oder der Benutzername des Benutzers sowie die Oid bzw. der Name der hinzuzufügenden Benutzerrolle werden als Parameter in der URL mitgegeben, siehe nachfolgendes Beispiel.

```
https://xycompany.ennoxx.cloud/api/Users/AssignRole?oidOrUserName=testuser1&roleOidOrName=Administrator
```

10.4.2 Aufbau der Http-Response

Die Http-Response liefert ein `ExtendedUserListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:

```
{
  "Objects": [
    {
      "InitialPassword": "string",
      "Roles": [
        {
          "Name": "string",
          "IsAdministrative": true,
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        }
      ],
      "UserName": "string",
      "FirstName": "string",
      "LastName": "string",
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ ExtendedUserDto	Enthält den aktualisierten Benutzer
InitialPassword	Initialpasswort	String	
Roles	Rollen	Array vom Typ RolesDto	Auflistung aller dem Benutzer zugeordneten Benutzerrollen
UserName	Benutzername	String	Enthält das Logon, welches für die Anmeldung mit Benutzernamen und Passwort benötigt wird
FirstName	Vorname	String	
LastName	Nachname	String	
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des erstellten Benutzers, welche für weitere Auswertungen und Methodenaufrufe verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall wird im `Objects`-Array der aktualisierte Benutzer mit dessen Details (z.B. Benutzernamen, Rollen, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

10.4.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `AssignRole` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var usersClient = new UsersClient(httpClient);
usersClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var task = usersClient.AssignRoleAsync("testuser1", "Administrator");
var result = task.Result;
if (result.Result == ExtendedUserListResultDtoResult.Success)
{
   log.Info(result.Message);
   var oid = result.Objects.First().Oid;
   //...
}
else
{
   throw new Exception(result.Message);
}
```

10.5 Zuweisung von Unternehmen zu einem Benutzer

Der API-Controller `Users` stellt für die Zuweisung von Unternehmen zu einem Benutzer in ennoxx.banking die Methode `AssignIdentity` zur Verfügung. Das Ergebnis sowie die aktualisierten Benutzerdaten werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/Users/Users_AssignIdentity

10.5.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/Users/AssignIdentity
Method	GET
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
URL-Parameter „oidOrUserName“	Wahlweise die Objekt-ID (Oid) oder der Benutzername des Benutzers, dem das Unternehmen hinzugefügt werden soll
URL-Parameter „identityOidOrCompanyName“	Wahlweise die Objekt-ID (Oid) oder der Firmenname des hinzuzufügenden Unternehmens

Die Oid oder der Benutzername des Benutzers sowie die Oid bzw. der Firmenname des hinzuzufügenden Unternehmens werden als Parameter in der URL mitgegeben, siehe nachfolgendes Beispiel.

```
https://xycompany.ennoxx.cloud/api/Users/AssignIdentity?oidOrUserName=testuser1&identityOidOrCompany
Name= XY%20Company%20GmbH
```

10.5.2 Aufbau der Http-Response

Die Http-Response liefert ein `ExtendedUserListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:

```
{
  "Objects": [
    {
      "InitialPassword": "string",
      "Roles": [
        {
          "Name": "string",
          "IsAdministrative": true,
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        }
      ],
      "UserName": "string",
      "FirstName": "string",
      "LastName": "string",
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ ExtendedUserDto	Enthält den aktulaisierten Benutzer
InitialPassword	Initialpasswort	String	
Roles	Rollen	Array vom Typ RolesDto	Auflistung aller dem Benutzer zugeordneten Benutzerrollen
UserName	Benutzername	String	Enthält das Logon, welches für die Anmeldung mit Benutzernamen und Passwort benötigt wird
FirstName	Vorname	String	
LastName	Nachname	String	
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des erstellten Benutzers, welche für weitere Auswertungen und Methodenaufrufe verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall wird im `Objects`-Array der aktualisierte Benutzer mit dessen Details (z.B. Benutzernamen, Rollen, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

10.5.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `AssignIdentity` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var usersClient = new UsersClient(httpClient);
usersClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var task = usersClient.AssignIdentityAsync("testuser1", "XY Company GmbH");
var result = task.Result;
if (result.Result == ExtendedUserListResultDtoResult.Success)
{
   log.Info(result.Message);
   var oid = result.Objects.First().Oid;
   //...
}
else
{
   throw new Exception(result.Message);
}
```

10.6 Erstellung Auth Token für einen Benutzer

Der API-Controller `Users` stellt für die Erzeugung eines Auth Tokens für einen Benutzer in ennoxx.banking die Methode `CreateAuthToken` zur Verfügung. Der erzeugte Auth Token sowie die Benutzerdaten werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/Users/Users_CreateAuthToken

10.6.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/Users/CreateAuthToken
Method	GET
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
Optionaler URL-Parameter „expirationDate“	Angabe eines Ablaufdatums für den Auth Token. Sofern nicht belegt, ist dieser automatisch ein Jahr lang gültig.
Optionaler URL-Parameter „description“	Angabe einer Beschreibung für den Auth Token.

Ein Ablaufdatum und/oder eine Beschreibung für den Auth Token können optional als Parameter in der URL mitgegeben werden, siehe nachfolgendes Beispiel.

```
https://xycompany.ennoxx.cloud/api/Users/CreateAuthToken?expirationDate=2025-12-31&description=Token%20f%C3%BCr%20die%20Nutzung%20der%20API
```

10.6.2 Aufbau der Http-Response

Die Http-Response liefert ein `AuthTokenListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:

```
{
  "Objects": [
    {
      "User": {
        "InitialPassword": "string",
        "Roles": [
          {
            "Name": "string",
            "IsAdministrative": true,
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          }
        ],
        "UserName": "string",
        "FirstName": "string",
        "LastName": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "Token": "string",
      "ExpirationDate": "2025-12-31T00:00:00.000Z",
      "Description": "string",
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ AuthTokenDto	Enthält den erzeugten Auth Token
User	Benutzer	Extended UserDto	Enthält die Benutzerdaten
Token	Token	String	Der durch ennoxx.banking erstellte Auth Token. Dieser muss gemerkt werden, da er aus Sicherheitsgründen zu einem späteren Zeitpunkt nicht mehr durch ennoxx.banking angezeigt oder abgerufen werden kann.
ExpirationDate	Ablaufdatum	DateTime	Ablaufdatum des Auth Tokens. Standardmäßige Gütigkeit: 1 Jahr
Description	Beschreibung	String	
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des erstellten Auth Tokens. Kann ignoriert werden da dieser für keinen späteren Prozess benötigt wird.
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall wird im `Objects`-Array der erstellte Auth Token inkl. der zugehörigen Benutzerdaten zurückgegeben. Da der Auth Token aus Sicherheitsgründen zu keinem späteren Zeitpunkt mehr angezeigt werden kann, muss dieser an dieser Stelle dringend gemerkt werden. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

10.6.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `CreateAuthToken` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var usersClient = new UsersClient(httpClient);
usersClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var task = usersClient.CreateAuthTokenAsync(new System.DateTimeOffset(new System.DateTime(2025, 12,
31)), "Token für die Nutzung der API");
var result = task.Result;
if (result.Result == AuthTokenListResultDtoResult.Success)
{
   log.Info(result.Message);
   var token = result.Objects.First().Token;
   //...
}
else
{
   throw new Exception(result.Message);
}
```

10.7 Anlage Konten

Der API-Controller `Accounts` stellt für die Anlage von neuen Konten in ennoxx.banking die Methoden `Create2` (für eine Einzelanlage) sowie `CreateBulk2` (für die Anlage mehrerer Konten in einem Rutsch) zur Verfügung. Die in diesem Zuge erzeugten Konten werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/Accounts/Accounts_Create2

10.7.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/Accounts/Create2 oder https://xycompany.ennoxx.cloud/api/Accounts/CreateBulk2
Method	POST
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json

Die Daten des zu erstellenden Kontos werden im Http-Request als Datenstrom mitgegeben. Das hierbei zu nutzende `AccountCreateRequestDto`-Objekt ist eine JSON-Struktur mit dem nachfolgenden Aufbau. Bei Nutzung der Methode `CreateBulk2` wird die gleiche Dto-Struktur in einem Array verwendet.

```
{
  "Description": "string",
  "Iban": "string",
  "Bic": "string",
  "AccountNumber": "string",
  "NationalBankCode": "string",
  "AccountIdentifier": "string",
  "AccountIdentifier2": "string",
  "AccountIdentifier3": "string",
  "AccountType": "OpenAccount",
  "AccountCategory": "string",
  "CurrencyIsoCode": "string",
  "AccountHolderOidOrName": "string"
}
```

Feldname	Bezeichnung	Datentyp	Pflichtfeld	Beschreibung
Description	Bezeichnung	String	Nein	
Iban	IBAN	String	Nein	IBAN oder Kontonummer muss belegt werden. Für die Durchführung von SEPA Zahlungen wird eine IBAN benötigt.
Bic	BIC	String	Nein	
AccountNumber	Kontonummer	String	Nein	IBAN oder Kontonummer muss belegt werden. Für die Durchführung von Auslandszahlungen (DTAZV) wird eine Kontonummer benötigt.
NationalBankCode	Bankleitzahl	String	Nein	Für die Durchführung von Auslandszahlungen (DTAZV) wird eine Bankleitzahl benötigt.
AccountIdentifier	Konto ID	String	Nein	Wird für die Zuordnung von Kontoauszügen im MT940 Format benötigt. Hierzu muss der Wert aus Feld :25: des MT940 hinterlegt werden. Bei camt-Formaten erfolgt die Zuordnung über die IBAN, in diesem Fall kann das Feld leer gelassen werden.
AccountIdentifier2	Konto ID 2	String	Nein	Alternative Konto ID
AccountIdentifier3	Konto ID 3	String	Nein	Alternative Konto ID
AccountType	Kontoart	String	Nein	Kontoart: • OpenAccount • AssetsAccount • LoanAccount • Guarantee • Other Sofern nicht angegeben, wird OpenAccount verwendet.
AccountCategory	Kontokategorie	String	Nein	
CurrencyIsoCode	Währung	String	Ja	ISO Code der Kontowährung (z.B. „EUR“)
AccountHolderOidOrName	Kontoinhaber	String	Ja	Wahlweise Angabe der Objekt-ID (Oid) oder Name eines bestehenden Kontoinhabers (Unternehmen)

10.7.2 Aufbau der Http-Response

Die Http-Response liefert ein `AccountListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:

```
{
  "Objects": [
    {
      "Description": "string",
      "Iban": "string",
      "Bic": "string",
      "AccountNumber": "string",
      "NationalBankCode": "string",
      "AccountIdentifier": "string",
      "AccountIdentifier2": "string",
      "AccountIdentifier3": "string",
      "AccountType": "OpenAccount",
      "AccountCategory": "string",
      "CurrencyIsoCode": "string",
      "AccountHolder": {
        "CompanyName": "string",
        "CreditorIdentification": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ AccountDto	Enthält die angelegten Konten
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall werden im `Objects`-Array die erzeugten Konten mit deren Details (z.B. IBAN, BIC, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

10.7.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `Create2` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var accountsClient = new AccountsClient(httpClient);
accountsClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var accountDto = new AccountCreateRequestDto() { Iban = "DE95500800000000001234", CurrencyIsoCode =
"EUR", AccountHolderOidOrName = "XY Company GmbH" };
var task = accountsClient.Create2Async(accountDto);
var result = task.Result;
if (result.Result == AccountListResultDtoResult.Success)
{
   log.Info(result.Message);
   foreach (var createdAccount in result.Objects)
   {
       var oid = createdAccount.Oid;
       //...
   }
}
else
{
   throw new Exception(result.Message);
}
```

10.8 Anlage EBICS-Zugang

Der API-Controller `EbicsChannels` stellt für die Anlage von neuen EBICS-Zugängen in ennoxx.banking die Methoden `Create2` (für eine Einzelanlage) sowie `CreateBulk2` (für die Anlage mehrerer Zugänge in einem Rutsch) zur Verfügung. Der oder die in diesem Zuge erzeugten Zugänge werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/EbicsChannels/EbicsChannels_Create2

10.8.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/EbicsChannels/Create2 oder https://xycompany.ennoxx.cloud/api/EbicsChannels/CreateBulk2
Method	POST
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json

Die Daten des zu erstellenden EBICS-Zugangs werden im Http-Request als Datenstrom mitgegeben. Das hierbei zu nutzende `EbicsChannelCreateRequestDto`-Objekt ist eine JSON-Struktur mit dem nachfolgenden Aufbau. Bei Nutzung der Methode `CreateBulk2` wird die gleiche Dto-Struktur in einem Array verwendet.

```
{
  "Caption": "string",
  "EbicsVersion": "EBICS_2_5",
  "SignatureVersion": "A006",
  "HostName": "string",
  "Url": "string",
  "CustomerId": "string"
}
```

Feldname	Bezeichnung	Datentyp	Pflichtfeld	Beschreibung
Caption	Bezeichnung	String	Nein	Hier sollte idealerweise eine aussagekräftige Bezeichnung für den EBICS Zugang mitgegeben werden. Z.B. mit dem Kundennamen und dem Banknamen.
EbicsVersion	EBICS Version	String	Nein	Beinhaltet die zu verwendende EBICS Version. Sofern nicht angegeben, wird die aktuellste Version (EBICS 2.5) verwendet (wird empfohlen).
SignatureVersion	Unterschriftsversion	String	Nein	Beinhaltet die zu verwendende Version für den Unterschriftsschlüssel. Sofern nicht angegeben, wird die aktuellste Version (A006) verwendet (wird empfohlen).
Hostname	Hostname	String	Ja	Beinhaltet den Hostnamen der Bank. Dieser wird von der Bank mit den EBICS-Zugangsdaten zur Verfügung gestellt.
Url	URL	String	Ja	Beinhaltet die EBICS-URL der Bank. Diese wird von der Bank mit den EBICS-Zugangsdaten zur Verfügung gestellt.
CustomerId	Kunden-ID	String	Ja	Beinhaltet die Kunden-ID des Kunden. Diese wird von der Bank mit den EBICS-Zugangsdaten zur Verfügung gestellt.

10.8.2 Aufbau der Http-Response

Die Http-Response liefert ein `EbicsChannelListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:


```
{
  "Objects": [
    {
      "EbicsVersion": "EBICS_2_5",
      "SignatureVersion": "A006",
      "HostName": "string",
      "Url": "string",
      "CustomerId": "string",
      "Users": [
        {
          "UserId": "string",
          "InternalUser": {
            "UserName": "string",
            "FirstName": "string",
            "LastName": "string",
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          },
          "AuthorisationType": "A",
          "State": "New",
          "IniLetter": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        }
      ],
      "DefaultUser": {
        "UserId": "string",
        "InternalUser": {
          "UserName": "string",
          "FirstName": "string",
          "LastName": "string",
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "AuthorisationType": "None",
        "State": "New",
        "IniLetter": {
          "FileName": "string",
          "FileSize": 0,
          "Data": "string"
        },
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "EncryptionMethod": "string",
      "Caption": "string",
      "ChannelType": "Communication",
      "Name": "string",
      "Description": "string",
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ EbicsChannelDto	Enthält die angelegten EBICS-Kanäle
EbicsVersion	EBICS Version	String	
SignatureVersion	Unterschriftsversion	String	
Hostname	Hostname	String	
Url	URL	String	
CustomerId	Kunden-ID	String	
Users	EBICS Teilnehmer	Array vom Typ EbicsUserDto	Auflistung aller EBICS Teilnehmer des Kanals
DefaultUser	Standardbenutzer	EbicsUserDto	Enthält den Standardbenutzer des Kanals, der als Transportbenutzer für einen EBICS Auftrag verwendet wird. Der erste zugeordnete Benutzer wird automatisch als Standardbenutzer verwendet. Eine nachträgliche Änderung ist über die Methode `SetDefaultUser` oder über das WebUI möglich.
EncryptionMethod	Verschlüsselungsmethode	String	Angabe einer optionalen Verschlüsselungsmethode. Kann ignoriert werden, da diese bei EBICS nicht zum Einsatz kommt.
Caption	Bezeichnung	String	
ChannelType	Typ	String	Enthält den Kanaltyp „Communication“.
Name	Kanalname	String	Enthält den internen Kanalnamen „EBICS“.
Description	Kanlbeschreibung	String	Enthält eine interne Kanalbeschreibung mit der Kunden-ID.
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des erstellten EBICS-Kanals, welche für weitere Auswertungen und Methodenaufrufe verwendet werden kann.
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall werden im `Objects`-Array die erzeugten EBICS-Kanäle mit deren Details (z.B. Bezeichung, Hostname, Kunden-ID, Teilnehmer, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

10.8.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `Create2` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var ebicsChannelsClient = new EbicsChannelsClient(httpClient);
ebicsChannelsClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var ebicsChannelDto = new EbicsChannelCreateRequestDto() { Caption = "EBICS Testbank XY Company
GmbH", HostName = "TESTBANK", Url = "https://ebics-testbank.com/ebics", CustomerId = "K1234567", };
var task = ebicsChannelsClient.Create2Async(ebicsChannelDto);
var result = task.Result;
if (result.Result == EbicsChannelListResultDtoResult.Success)
{
   log.Info(result.Message);
   foreach (var createdEbicsChannel in result.Objects)
   {
       var oid = createdEbicsChannel.Oid;
       //...
   }
}
else
{
   throw new Exception(result.Message);
}
```

10.9 EBICS-Teilnehmer zu EBICS-Zugang hinzufügen

Der API-Controller `EbicsChannels` stellt für die Anlage von neuen EBICS-Teilnehmern in ennoxx.banking die Methode `CreateUser2` zur Verfügung. Der in diesem Zuge erzeugte EBICS-Teilnehmer wird anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/EbicsChannels/EbicsChannels_CreateUser2

10.9.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/EbicsChannels/CreateUser2
Method	POST
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json

Die Daten des zu erstellenden EBICS-Teilnehmers werden im Http-Request als Datenstrom mitgegeben. Das hierbei zu nutzende `EbicsUserCreateRequestDto`-Objekt ist eine JSON-Struktur mit dem nachfolgenden Aufbau.

```
{
  "ChannelOidOrCaption": "string",
  "UserId": "string",
  "InternalUserOidOrName": "string",
  "AuthorisationType": "None"
}
```

Feldname	Bezeichnung	Datentyp	Pflichtfeld	Beschreibung
ChannelOidOrCaption	EBICS Kanal	String	Ja	Wahlweise die Objekt-ID (Oid) oder die Bezeichnung des EBICS Kanals, zu dem der EBICS-Teilnehmer hinzugefügt werden soll.
UserId	Teilnehmer-ID	String	Ja	Beinhaltet die Teilnehmer-ID des zu erstellenden EBICS-Teilnehmers. Diese wird von der Bank mit den EBICS-Zugangsdaten zur Verfügung gestellt.
InternalUserOidOrName	Interner Benutzer	String	Ja	Wahlweise die Objekt-ID (Oid) oder der Benutzername des internen ennoxx.banking Benutzers, der dem EBICS-Teilnehmer zugeordnet werden soll.
AuthorisationType	Unterschriftsberechtigung	String	Nein	Beinhaltet die Unterschriftsberechtigung des EBICS-Teilnehmers: • „None“ (Keine Unterschriftsberechtigung) • „E“ (Einzelunterschrift) • „A“ (Erstunterschrift) • „B“ (Zweitunterschrift) • „T“ (Transportunterschrift) Die Unterschriftsberechtigung wird von der Bank mit den EBICS-Zugangsdaten zur Verfügung gestellt. Sofern nicht angegeben, wird „None“ verwendet.

10.9.2 Aufbau der Http-Response

Die Http-Response liefert ein `EbicsChannelListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:


```
{
  "Objects": [
    {
      "EbicsVersion": "EBICS_2_5",
      "SignatureVersion": "A006",
      "HostName": "string",
      "Url": "string",
      "CustomerId": "string",
      "Users": [
        {
          "UserId": "string",
          "InternalUser": {
            "UserName": "string",
            "FirstName": "string",
            "LastName": "string",
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          },
          "AuthorisationType": "A",
          "State": "New",
          "IniLetter": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        }
      ],
      "DefaultUser": {
        "UserId": "string",
        "InternalUser": {
          "UserName": "string",
          "FirstName": "string",
          "LastName": "string",
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "AuthorisationType": "None",
        "State": "New",
        "IniLetter": {
          "FileName": "string",
          "FileSize": 0,
          "Data": "string"
        },
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "EncryptionMethod": "string",
      "Caption": "string",
      "ChannelType": "Communication",
      "Name": "string",
      "Description": "string",
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ EbicsChannelDto	Enthält den aktualisierten EBICS-Kanal
EbicsVersion	EBICS Version	String	
SignatureVersion	Unterschriftsversion	String	
Hostname	Hostname	String	
Url	URL	String	
CustomerId	Kunden-ID	String	
Users	EBICS Teilnehmer	Array vom Typ EbicsUserDto	Auflistung aller EBICS Teilnehmer des Kanals
DefaultUser	Standardbenutzer	EbicsUserDto	Enthält den Standardbenutzer des Kanals, der als Transportbenutzer für einen EBICS Auftrag verwendet wird. Der erste zugeordnete Benutzer wird automatisch als Standardbenutzer verwendet. Eine nachträgliche Änderung ist über die Methode `SetDefaultUser` oder über das WebUI möglich.
EncryptionMethod	Verschlüsselungsmethode	String	Angabe einer optionalen Verschlüsselungsmethode. Kann ignoriert werden, da diese bei EBICS nicht zum Einsatz kommt.
Caption	Bezeichnung	String	
ChannelType	Typ	String	Enthält den Kanaltyp „Communication“.
Name	Kanalname	String	Enthält den internen Kanalnamen „EBICS“.
Description	Kanlbeschreibung	String	Enthält eine interne Kanalbeschreibung mit der Kunden-ID.
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des erstellten EBICS-Kanals, welche für weitere Auswertungen und Methodenaufrufe verwendet werden kann.
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall wird im `Objects`-Array der aktualisierte EBICS-Kanal mit dessen Details (z.B. Bezeichung, Hostname, Kunden-ID, Teilnehmer, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

10.9.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `CreateUser2` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var ebicsChannelsClient = new EbicsChannelsClient(httpClient);
ebicsChannelsClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var ebicsUserDto = new EbicsUserCreateRequestDto() { ChannelOidOrCaption = "EBICS Testbank XY
Company GmbH", UserId = "U1234567", InternalUserOidOrName = "testuser1", AuthorisationType =
EbicsUserCreateRequestDtoAuthorisationType.A };
var task = ebicsChannelsClient.CreateUser2Async(ebicsUserDto);
var result = task.Result;
if (result.Result == EbicsChannelListResultDtoResult.Success)
{
   log.Info(result.Message);
   foreach (var ebicsUser in result.Objects.First().Users)
   {
       var oid = ebicsUser.Oid;
       //...
   }
}
else
{
   throw new Exception(result.Message);
}
```

10.10 Erstellung EBICS-Transportschlüssel

Der API-Controller `Users` stellt für die Erstellung von EBICS-Transportschlüsseln (in der aktuellsten Version X002/E002 und einer Schlüssellänge von je 2.048 Bit) für einen Benutzer in ennoxx.banking die Methode `CreateEbicsTransportKeys` zur Verfügung. Das Ergebnis sowie die aktualisierten Benutzerdaten werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/Users/Users_CreateEbicsTransportKeys

10.10.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/Users/CreateEbicsTransportKeys
Method	GET
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
URL-Parameter „oidOrUserName“	Wahlweise die Objekt-ID (Oid) oder der Benutzername des Benutzers, für den die EBICS Transportschlüssel erstellt werden sollen
URL-Parameter „password“	Schlüsselpasswort

Die Oid oder der Benutzername des Benutzers, für den die Transportschlüssel erzeugt werden sollen sowie das Schlüsselpasswort werden als Parameter in der URL mitgegeben, siehe nachfolgendes Beispiel.

```
https://xycompany.ennoxx.cloud/api/Users/CreateEbicsTransportKeys?oidOrUserName=testuser1&password=test1234
```

10.10.2 Aufbau der Http-Response

Die Http-Response liefert ein `ExtendedUserListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:

```
{
  "Objects": [
    {
      "InitialPassword": "string",
      "Roles": [
        {
          "Name": "string",
          "IsAdministrative": true,
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        }
      ],
      "UserName": "string",
      "FirstName": "string",
      "LastName": "string",
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ ExtendedUserDto	Enthält den aktualisierten Benutzer
InitialPassword	Initialpasswort	String	
Roles	Rollen	Array vom Typ RolesDto	Auflistung aller dem Benutzer zugeordneten Benutzerrollen
UserName	Benutzername	String	Enthält das Logon, welches für die Anmeldung mit Benutzernamen und Passwort benötigt wird
FirstName	Vorname	String	
LastName	Nachname	String	
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des erstellten Benutzers, welche für weitere Auswertungen und Methodenaufrufe verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall wird im `Objects`-Array der aktualisierte Benutzer mit dessen Details (z.B. Benutzernamen, Rollen, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

10.10.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `CreateEbicsTransportKeys` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var usersClient = new UsersClient(httpClient);
usersClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var task = usersClient.CreateEbicsTransportKeysAsync("testuser1", "test1234");
var result = task.Result;
if (result.Result == ExtendedUserListResultDtoResult.Success)
{
   log.Info(result.Message);
   var oid = result.Objects.First().Oid;
   //...
}
else
{
   throw new Exception(result.Message);
}
```

10.11 Erstellung EBICS-Unterschriftsschlüssel

Der API-Controller `Users` stellt für die Erstellung eines EBICS-Unterschriftsschlüssels (in der aktuellsten Version A006 und einer Schlüssellänge von 2.048 Bit) für einen Benutzer in ennoxx.banking die Methode `CreateEbicsSignatureKey` zur Verfügung. Das Ergebnis sowie die aktualisierten Benutzerdaten werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/Users/Users_CreateEbicsSignatureKey

10.11.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/Users/CreateEbicsSignatureKey
Method	GET
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
URL-Parameter „oidOrUserName“	Wahlweise die Objekt-ID (Oid) oder der Benutzername des Benutzers, für den der EBICS Unterschriftsschlüssel erstellt werden soll
URL-Parameter „password“	Schlüsselpasswort

Die Oid oder der Benutzername des Benutzers, für den der Unterschriftsschlüssel erzeugt werden soll sowie das Schlüsselpasswort werden als Parameter in der URL mitgegeben, siehe nachfolgendes Beispiel.

```
https://xycompany.ennoxx.cloud/api/Users/CreateEbicsSignatureKey?oidOrUserName=testuser1&password=test1234
```

10.11.2 Aufbau der Http-Response

Die Http-Response liefert ein `ExtendedUserListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:

```
{
  "Objects": [
    {
      "InitialPassword": "string",
      "Roles": [
        {
          "Name": "string",
          "IsAdministrative": true,
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        }
      ],
      "UserName": "string",
      "FirstName": "string",
      "LastName": "string",
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ ExtendedUserDto	Enthält den aktualisierten Benutzer
InitialPassword	Initialpasswort	String	
Roles	Rollen	Array vom Typ RolesDto	Auflistung aller dem Benutzer zugeordneten Benutzerrollen
UserName	Benutzername	String	Enthält das Logon, welches für die Anmeldung mit Benutzernamen und Passwort benötigt wird
FirstName	Vorname	String	
LastName	Nachname	String	
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des erstellten Benutzers, welche für weitere Auswertungen und Methodenaufrufe verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall wird im `Objects`-Array der aktualisierte Benutzer mit dessen Details (z.B. Benutzernamen, Rollen, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

10.11.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `CreateEbicsSignatureKey` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var usersClient = new UsersClient(httpClient);
usersClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var task = usersClient.CreateEbicsSignatureKeyAsync("testuser1", "test1234");
var result = task.Result;
if (result.Result == ExtendedUserListResultDtoResult.Success)
{
   log.Info(result.Message);
   var oid = result.Objects.First().Oid;
   //...
}
else
{
   throw new Exception(result.Message);
}
```

10.12 Änderung Passwort des Unterschriftsschlüssels

Der API-Controller `Users` stellt für die Änderung eines EBICS-Unterschriftsschlüssel-Passworts in ennoxx.banking die Methode `ChangeEbicsSignatureKeyPassword` zur Verfügung. Die Änderung erfolgt für den aktuellsten Unterschriftsschlüssel des angemeldeten Benutzers. Das Ergebnis sowie die aktualisierten Benutzerdaten werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/Users/Users_ChangeEbicsSignatureKeyPassword

10.12.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/Users/ChangeEbicsSignatureKeyPassword
Method	GET
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
URL-Parameter „oldPassword“	Altes Schlüsselpasswort
URL-Parameter „newPassword“	Neues Schlüsselpasswort

Das alte und das neue Schlüsselpasswort werden als Parameter in der URL mitgegeben, siehe nachfolgendes Beispiel.

```
https://xycompany.ennoxx.cloud/api/Users/ChangeEbicsSignatureKeyPassword?oldPassowrd=test1234&newPassword=test5678
```

10.12.2 Aufbau der Http-Response

Die Http-Response liefert ein `ExtendedUserListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:

```
{
  "Objects": [
    {
      "InitialPassword": "string",
      "Roles": [
        {
          "Name": "string",
          "IsAdministrative": true,
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        }
      ],
      "UserName": "string",
      "FirstName": "string",
      "LastName": "string",
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ ExtendedUserDto	Enthält den aktualisierten Benutzer
InitialPassword	Initialpasswort	String	
Roles	Rollen	Array vom Typ RolesDto	Auflistung aller dem Benutzer zugeordneten Benutzerrollen
UserName	Benutzername	String	Enthält das Logon, welches für die Anmeldung mit Benutzernamen und Passwort benötigt wird
FirstName	Vorname	String	
LastName	Nachname	String	
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des erstellten Benutzers, welche für weitere Auswertungen und Methodenaufrufe verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall wird im `Objects`-Array der aktualisierte Benutzer mit dessen Details (z.B. Benutzernamen, Rollen, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

10.12.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `ChangeEbicsSignatureKeyPassword` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var usersClient = new UsersClient(httpClient);
usersClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var task = usersClient.ChangeEbicsSignatureKeyPasswordAsync("test1234", "test5678");
var result = task.Result;
if (result.Result == ExtendedUserListResultDtoResult.Success)
{
   log.Info(result.Message);
   var oid = result.Objects.First().Oid;
   //...
}
else
{
   throw new Exception(result.Message);
}
```

10.13 Initialisierung der Benutzerschlüssel bei der Bank

Der API-Controller `EbicsChannels` stellt für die Benutzerinitialisierung eines EBICS-Zugangs in ennoxx.banking die Methode `CreateIniHiaJobs` zur Verfügung. Die in diesem Zuge erzeugten EBICS-Aufträge (mit den beiden Auftragsarten INI und HIA) werden anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/EbicsChannels/EbicsChannels_CreateIniHiaJobs

10.13.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/EbicsChannels/CreateIniHiaJobs
Method	GET
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
URL-Parameter „oidOrCaption“	Wahlweise die Objekt-ID (Oid) oder die Bezeichnung des EBICS-Kanals
URL-Parameter „userOidOrUserName“	Wahlweise die Objekt-ID (Oid) oder der Benutzername (Logon) des ennoxx.banking Benutzers für den die Benutzerinitialisierung durchgeführt werden soll
URL-Parameter „password“	Das Schlüsselpasswort für die EBICS Transportschlüssel (X002/E002) des Benutzers

Die Objekt-ID (Oid) oder die Bezeichnung des EBICS Kanals, die Objekt-ID (Oid) oder der Benutzername (Logon) des ennoxx.banking Benutzers und das Schlüsselpasswort der EBICS Transportschlüssel werden als Parameter in der URL mitgegeben, siehe nachfolgendes Beispiel.

```
https://xycompany.ennoxx.cloud/api/EbicsChannels/CreateEbicsIniHiaJobs?oidOrCaption=EBICS%20Testbank%20XY%20Company%20GmbH&oidOrUserName=testuser1&password=test1234
```

10.13.2 Aufbau der Http-Response

Die Http-Response liefert ein `EbicsJobListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:


```
{
  "Objects": [
    {
      "OrderNumber": "string",
      "OrderType": {
        "Code": "string",
        "Caption": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "EbicsChannel": {
        "EbicsVersion": "EBICS_2_5",
        "SignatureVersion": "A006",
        "HostName": "string",
        "Url": "string",
        "CustomerId": "string",
        "Users": [
          {
            "UserId": "string",
            "InternalUser": {
              "UserName": "string",
              "FirstName": "string",
              "LastName": "string",
              "Oid": "00000000-0000-0000-0000-000000000000",
              "ClassName": "string"
            },
            "AuthorisationType": "None",
            "State": "New",
            "IniLetter": {
              "FileName": "string",
              "FileSize": 0,
              "Data": "string"
            },
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          }
        ],
        "DefaultUser": {
          "UserId": "string",
          "InternalUser": {
            "UserName": "string",
            "FirstName": "string",
            "LastName": "string",
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          },
          "AuthorisationType": "None",
          "State": "New",
          "IniLetter": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "EncryptionMethod": "string",
        "Caption": "string",
        "ChannelType": "Communication",
        "Name": "string",
        "Description": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "EbicsUser": {
        "UserId": "string",
        "InternalUser": {
          "UserName": "string",
          "FirstName": "string",
```

```
        "LastName": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "AuthorisationType": "None",
      "State": "New",
      "IniLetter": {
        "FileName": "string",
        "FileSize": 0,
        "Data": "string"
      },
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    },
    "Signatures": [
      {
        "AuthorisationType": "None",
        "KeyType": "A006",
        "DateTimeSigned": "2021-09-20T12:36:47.820Z",
        "User": {
          "UserId": "string",
          "InternalUser": {
            "UserName": "string",
            "FirstName": "string",
            "LastName": "string",
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          },
          "AuthorisationType": "None",
          "State": "New",
          "IniLetter": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      }
    ],
    "Protocols": [
      {
        "TimeStamp": "2021-09-20T12:36:47.820Z",
        "ActionCode": 0,
        "ActionText": "string",
        "DisplayFile": "string",
        "OrderID": "string",
        "OrderText": "string",
        "OrderType": "string",
        "ReferencedOrderType": "string",
        "ReferencedOrderId": "string",
        "ResultCode": 0,
        "ResultText": "string",
        "Text": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      }
    ],
    "FetchFrom": "2021-09-20T12:36:47.820Z",
    "FetchTo": "2021-09-20T12:36:47.820Z",
    "Channel": {
      "Caption": "string",
      "ChannelType": "Communication",
      "Name": "string",
      "Description": "string",
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    },
    "SpecificDetail": "string",
    "State": "None",
    "Caption": "string",
    "CommunicationDirection": "Send",
    "DueTime": "2021-09-20T12:36:47.820Z",
```

```
      "IntervalType": "Single",
      "Interval": 0,
      "LastExecution": "2021-09-20T12:36:47.820Z",
      "PermissionClass": "string",
      "WebUrl": "string",
      "LastLog": {
        "State": "Success",
        "DateTime": "2021-09-20T12:36:47.820Z",
        "Description": "string",
        "Protocol": "string"
      },
      "Files": [
        {
          "OriginalFile": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ContentType": "Undefined",
          "ReportHtml": "string",
          "ReportSummaryHtml": "string",
          "ReportPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ReportSummaryPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "FileHash": {
            "Method": "None",
            "Value": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        }
      ],
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ EbicsJobDto	Enthält die beiden angelegten EBICS-Aufträge (mit den Auftragsarten INI und HIA)
OrderNumber	Auftragsnummer	String	Die Auftragsnummer wird bei einer erfolgreichen Ausführung von der Bank vergeben
OrderType	Auftragsart	EbicsOrderTypeDto	Enthält Details zur Auftragsart (INI oder HIA)
EbicsChannel	EBICS Kanal	EbicsChannelDto	Enthalt Details zum EBICS Kanal
EbicsUser	EBICS Benutzer	EbicsUserDto	Enthält Details des EBICS Transport-benutzers inkl. der vom Benutzer zu unterschreibenden INI-Briefe, die an die Bank gesendet werden müssen
Signatures	Unterschriften	Array vom Typ EbicsSignatureDto	Enthält die geleisteten Unterschriften des Auftrags (bei INI und HIA ist diese Liste leer)
Protocols	EBICS Protokolle	Array vom Typ EbicsProtocolDto	Enthält die EBICS Protokolle des Auftrags, nachdem dieser zur Bank gesendet und ein PTK-Auftrag durchgeführt wurde
FetchFrom	Historischer Abruf von	DateTime	Nur belegt bei historischen Datenabrufen
FetchTo	Historischer Abruf bis	DateTime	Nur belegt bei historischen Datenabrufen
Channel	Kanal	BasicChannelDto	Enthält Basisinformationen des Kanals
SpecificDetail	Spezifische Details	String	Enthält eine zusammenfassende Kurzinformation des Auftrags
State	Status	String	Aktueller Status des Auftrags
Caption	Bezeichnung	String	
CommunicationDirection	Richtung	String	Enthält die Richtung des Auftrags (Senden oder Empfangen)
DueTime	Ausführungszeitpunkt	DateTime	
IntervalType	Art des Intervalls	String	Einmaliger oder wiederkehrender Autrag mit dem angegebenen Intervall (INI und HIA sind einmalige Aufträge)
Interval	Intervall	Integer	
LastExecution	Letzte Ausführung am	DateTime	Nur belegt bei wiederkehrenden Aufträgen, für die eine letztes Ausführungsdatum festgelegt wurde
PermissionClass	Zugriffsklasse	String	
WebUrl	Web URL	String	Enthält den Weblink mit dem der Auftrag im Web UI angezeigt werden kann
LastLog	Letztes Log	BasicLogDto	Enthält Informationen des letzten Logs
Files	Dateien	Array vom Typ ContextFileDto	Enthält bei Sendeaufträgen die zu übertragenden Nutzdaten
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des erstellten EBICS-Auftrags, welche für weitere Auswertungen und Methodenaufrufe verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall werden im `Objects`-Array die beiden erzeugten EBICS-Aufträge (mit den Auftragsarten INI und HIA) mit deren Details (z.B. Auftragsart, EBICS-Kanal, Oid, etc.) zurückgegeben. Der Transportbenutzer im Feld `EbicsUser` enthält die Initialisierungsbriefe, die vom Benutzer manuell unterschrieben und anschließend an die Bank gesendet werden müssen. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

10.13.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `CreateIniHiaJobs` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var ebicsChannelsClient = new EbicsChannelsClient(httpClient);
ebicsChannelsClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var task = ebicsChannelsClient.CreateIniHiaJobsAsync("EBICS Testbank XY Company GmbH", "testuser1",
"test1234");
var result = task.Result;
if (result.Result == EbicsJobListResultDtoResult.Success)
{
   log.Info(result.Message);
   foreach (var createdEbicsJob in result.Objects)
   {
       var oid = createdEbicsJob.Oid;
       var iniLetterContent = createdEbicsJob.EbicsUser.IniLetter.Data;
       //...
   }
}
else
{
   throw new Exception(result.Message);
}
```

10.14 Abholung der Bankschlüssel

Nachdem im ersten Schritt die EBICS Benutzerintialisierung mittels INI und HIA erfolgt ist und die unterschriebenen Initialisierungsbriefe an die Bank gesendet wurden, müssen abschließend einmalig die Bankschlüssel abgerufen werden. Dies kann allerdings erst erfolgen, sofern der Benutzer durch die Bank freigeschaltet wurde.

Der API-Controller `EbicsChannels` stellt für die Abholung von Bankschlüsseln eines EBICS-Zugangs in ennoxx.banking die Methode `CreateHpbJob` zur Verfügung. Der in diesem Zuge erzeugte EBICS-Auftrag (mit der Auftragsart HPB) wird anschließend innerhalb der Http-Response zurückgegeben. Im Falle eines Fehlers innerhalb des Prozesses wird der entsprechende Status sowie die Fehlermeldung zurückgegeben.

Technische Details zu dieser Methode können der Swagger-UI unter folgendem Link entnommen werden:
https://xycompany.ennoxx.cloud/api/swagger/ui/index#!/EbicsChannels/EbicsChannels_CreateHpbJob

10.14.1 Aufbau des Http-Requests

Nachfolgende Parameter müssen im Http-Request belegt werden:

Parameter	Inhalt
URL	https://xycompany.ennoxx.cloud/api/EbicsChannels/CreateHpbJob
Method	GET
Authorization Header	Angabe der Login-Daten, wahlweise: • Basic-Authorization (Benutzername und Passwort) • Bearer-Authorization (Auth Token)
ContentType	application/json
URL-Parameter „oidOrCaption“	Wahlweise die Objekt-ID (Oid) oder die Bezeichnung des EBICS-Kanals
URL-Parameter „userOidOrUserName“	Wahlweise die Objekt-ID (Oid) oder der Benutzername (Logon) des ennoxx.banking Benutzers für dessen EBICS Zugang die Bankschlüssel abgerufen werden sollen

Die Objekt-ID (Oid) oder die Bezeichnung des EBICS Kanals sowie die Objekt-ID (Oid) oder der Benutzername (Logon) des ennoxx.banking Benutzers werden als Parameter in der URL mitgegeben, siehe nachfolgendes Beispiel.

```
https://xycompany.ennoxx.cloud/api/EbicsChannels/CreateEbicsHpbJob?oidOrCaption=EBICS%20Testbank%20XY%20Company%20GmbH&oidOrUserName=testuser1
```

10.14.2 Aufbau der Http-Response

Die Http-Response liefert ein `EbicsJobListResultDto`-Objekt mit der nachfolgenden JSON-Struktur:


```
{
  "Objects": [
    {
      "OrderNumber": "string",
      "OrderType": {
        "Code": "string",
        "Caption": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "EbicsChannel": {
        "EbicsVersion": "EBICS_2_5",
        "SignatureVersion": "A006",
        "HostName": "string",
        "Url": "string",
        "CustomerId": "string",
        "Users": [
          {
            "UserId": "string",
            "InternalUser": {
              "UserName": "string",
              "FirstName": "string",
              "LastName": "string",
              "Oid": "00000000-0000-0000-0000-000000000000",
              "ClassName": "string"
            },
            "AuthorisationType": "None",
            "State": "New",
            "IniLetter": {
              "FileName": "string",
              "FileSize": 0,
              "Data": "string"
            },
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          }
        ],
        "DefaultUser": {
          "UserId": "string",
          "InternalUser": {
            "UserName": "string",
            "FirstName": "string",
            "LastName": "string",
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          },
          "AuthorisationType": "None",
          "State": "New",
          "IniLetter": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "EncryptionMethod": "string",
        "Caption": "string",
        "ChannelType": "Communication",
        "Name": "string",
        "Description": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "EbicsUser": {
        "UserId": "string",
        "InternalUser": {
          "UserName": "string",
          "FirstName": "string",
```

```
        "LastName": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      },
      "AuthorisationType": "None",
      "State": "New",
      "IniLetter": {
        "FileName": "string",
        "FileSize": 0,
        "Data": "string"
      },
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    },
    "Signatures": [
      {
        "AuthorisationType": "None",
        "KeyType": "A006",
        "DateTimeSigned": "2021-09-20T12:36:47.820Z",
        "User": {
          "UserId": "string",
          "InternalUser": {
            "UserName": "string",
            "FirstName": "string",
            "LastName": "string",
            "Oid": "00000000-0000-0000-0000-000000000000",
            "ClassName": "string"
          },
          "AuthorisationType": "None",
          "State": "New",
          "IniLetter": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        },
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      }
    ],
    "Protocols": [
      {
        "TimeStamp": "2021-09-20T12:36:47.820Z",
        "ActionCode": 0,
        "ActionText": "string",
        "DisplayFile": "string",
        "OrderID": "string",
        "OrderText": "string",
        "OrderType": "string",
        "ReferencedOrderType": "string",
        "ReferencedOrderId": "string",
        "ResultCode": 0,
        "ResultText": "string",
        "Text": "string",
        "Oid": "00000000-0000-0000-0000-000000000000",
        "ClassName": "string"
      }
    ],
    "FetchFrom": "2021-09-20T12:36:47.820Z",
    "FetchTo": "2021-09-20T12:36:47.820Z",
    "Channel": {
      "Caption": "string",
      "ChannelType": "Communication",
      "Name": "string",
      "Description": "string",
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    },
    "SpecificDetail": "string",
    "State": "None",
    "Caption": "string",
    "CommunicationDirection": "Send",
    "DueTime": "2021-09-20T12:36:47.820Z",
```

```
      "IntervalType": "Single",
      "Interval": 0,
      "LastExecution": "2021-09-20T12:36:47.820Z",
      "PermissionClass": "string",
      "WebUrl": "string",
      "LastLog": {
        "State": "Success",
        "DateTime": "2021-09-20T12:36:47.820Z",
        "Description": "string",
        "Protocol": "string"
      },
      "Files": [
        {
          "OriginalFile": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ContentType": "Undefined",
          "ReportHtml": "string",
          "ReportSummaryHtml": "string",
          "ReportPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "ReportSummaryPdf": {
            "FileName": "string",
            "FileSize": 0,
            "Data": "string"
          },
          "FileHash": {
            "Method": "None",
            "Value": "string"
          },
          "Oid": "00000000-0000-0000-0000-000000000000",
          "ClassName": "string"
        }
      ],
      "Oid": "00000000-0000-0000-0000-000000000000",
      "ClassName": "string"
    }
  ],
  "Result": "Success",
  "Message": "string",
  "DetailResult": "NoMoreDetails"
}
```

Feldname	Bezeichnung	Datentyp	Beschreibung
Objects	Objekte	Array vom Typ EbicsJobDto	Enthält den angelegten EBICS-Auftrag (mit der Auftragsart HPB)
OrderNumber	Auftragsnummer	String	Die Auftragsnummer wird bei einer erfolgreichen Ausführung von der Bank vergeben
OrderType	Auftragsart	EbicsOrderTypeDto	Enthält Details zur Auftragsart (HPB)
EbicsChannel	EBICS Kanal	EbicsChannelDto	Enthalt Details zum EBICS Kanal
EbicsUser	EBICS Benutzer	EbicsUserDto	Enthält Details des EBICS Transportbenutzers
Signatures	Unterschriften	Array vom Typ EbicsSignatureDto	Enthält die geleisteten Unterschriften des Auftrags (bei HPB ist diese Liste leer)
Protocols	EBICS Protokolle	Array vom Typ EbicsProtocolDto	Enthält die EBICS Protokolle des Auftrags, nachdem dieser zur Bank gesendet und ein PTK-Auftrag durchgeführt wurde
FetchFrom	Historischer Abruf von	DateTime	Nur belegt bei historischen Datenabrufen
FetchTo	Historischer Abruf bis	DateTime	Nur belegt bei historischen Datenabrufen
Channel	Kanal	BasicChannelDto	Enthält Basisinformationen des Kanals
SpecificDetail	Spezifische Details	String	Enthält eine zusammenfassende Kurzinformation des Auftrags
State	Status	String	Aktueller Status des Auftrags
Caption	Bezeichnung	String	
CommunicationDirection	Richtung	String	Enthält die Richtung des Auftrags (Empfangen)
DueTime	Ausführungszeitpunkt	DateTime	
IntervalType	Art des Intervalls	String	Einmaliger oder wiederkehrender Auftrag mit dem angegebenen Intervall (HPB ist ein einmaliger Auftrag)
Interval	Intervall	Integer	
LastExecution	Letzte Ausführung am	DateTime	Nur belegt bei wiederkehrenden Aufträgen, für die eine letztes Ausführungsdatum festgelegt wurde
PermissionClass	Zugriffsklasse	String	
WebUrl	Web URL	String	Enthält den Weblink mit dem der Auftrag im Web UI angezeigt werden kann
LastLog	Letztes Log	BasicLogDto	Enthält Informationen des letzten Logs
Files	Dateien	Array vom Typ ContextFileDto	Enthält bei Sendeaufträgen die zu übertragenden Nutzdaten
Oid	Objekt-ID	String	Enthält die eindeutige Objekt-ID (Guid) des erstellten EBICS-Auftrags, welche für weitere Auswertungen und Methodenaufrufe verwendet werden kann
ClassName	Klassenname	String	Vollqualifizierter Klassenname des Businessobjekts
Result	Ergebnis	String	Ergebniscode des durchgeführten Prozesses: • Success • Error • Warning
Message	Nachricht zum Ergebnis	String	Enthält zusätzliche Informationen zum aufgetretenen Ergebnis
DetailResult	Details zum Ergebnis	String	Zusatzinformation zum aufgetretenen Ergebnis: • NoMoreDetails • NoDataAvailable

Im Erfolgsfall wird im `Objects`-Array der erzeugte EBICS-Auftrag (mit der Auftragsart HPB) mit dessen Details (z.B. Auftragsart, EBICS-Kanal, Oid, etc.) zurückgegeben. Der Status im Feld `Result` wird in diesem Fall als „Success“ angegeben.

Im Fehlerfall bleibt das `Objects`-Array hingegen leer und im Feld `Result` wird „Error“ zurückgegeben. Die zugehörige Fehlermeldung ist in diesem Fall dem Feld `Message` zu entnehmen.

10.14.3 Code-Beispiel (C#)

Nachfolgendes Code-Beispiel zeigt, wie der API-Aufruf der Methode `CreateHpbJob` mittels der Programmiersprache C# erfolgen kann. In dem Beispiel wurde zur vereinfachten Nutzung die Ennoxx.Api.Client Bibliothek verwendet, die automatisch aus der Swagger-Definition der ennoxx.banking API erzeugt wurde. Diese Bibliothek stellt neben den zu verwendenden Dto-Objekten auch für jeden API-Controller einen entsprechenden Client inkl. asynchroner Methoden zur Verfügung.

```
var authToken = "00000000-0000-0000-0000-000000000000";
var httpClient = new HttpClient();
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
httpClient.BaseAddress = new Uri(url);

var ebicsChannelsClient = new EbicsChannelsClient(httpClient);
ebicsChannelsClient.BaseUrl = httpClient.BaseAddress.AbsoluteUri;

var task = ebicsChannelsClient.CreateHpbJobAsync("EBICS Testbank XY Company GmbH", "testuser1");
var result = task.Result;
if (result.Result == EbicsJobListResultDtoResult.Success)
{
   log.Info(result.Message);
   var oid = result.Objects.First().Oid;
   //...
}
else
{
   throw new Exception(result.Message);
}
```