---DRAFT Version 0.6---

SIX eBill SWP API: Empfehlung für die Schnittstelle zwischen Softwareherstellern und Netzwerkpartnern der SIX eBill

Einleitung

Bisher hatten viele Softwarehersteller Schnittstellen zu PostFinance (für E-Finance-Kunden) wie auch zu Paynet (für alle anderen Banken) implementiert. Mit der Neukonzeption der eBill durch SIX wird ein Netzwerkpartnermodell eingeführt, das weiteren Netzwerkpartnern (NWPs) erlaubt, E-Rechnungen als eBills direkt an die SIX Infrastruktur zu senden, ohne auf die bisherigen Einlieferkanäle eingeschränkt zu sein. Einzelne (auch grosse) Rechnungssteller können eBills nicht selbst direkt an SIX einliefern.

Aktuell gibt es rund ein Duzend NWPs, weitere werden wohl dazu kommen. Um zu verhindern, dass Softwarehersteller, welche ihren Kunden die Wahl des NWP freistellen möchten, ebenso viele unterschiedliche Schnittstellen implementieren müssen, gibt es die vorliegende Schnittstellenempfehlung. Sie basiert auf REST mit OAuth2 und vor allem auf dem NWP-API der SIX.

Sie ist für NWPs wie auch für Softwarehersteller freiwillig. Wer sie vollständig implementiert, darf seine Produktinfos ergänzen mit dem Hinweis: «Unterstützt die SIX eBill SWP API-Empfehlung».

Die aktuellen Mitglieder der Arbeitsgruppe, welche diese Empfehlung erarbeitet haben, streben an, die Verantwortung über diese Empfehlung einer sinnvollen Governance zu unterstellen, und sind auf der Suche nach einem geeigneten Rahmen dafür. Es könnte also sein, dass es zukünftig eine Lizenz, eine Erklärung oder eine Vereinbarung benötigt, um diese Empfehlung nutzen zu können. Sie soll aber auf jeden Fall kostenlos bleiben, für NWPs wie auch für Softwarehersteller.

Die aktuelle Version dieser Schnittstellenempfehlung befindet sich jeweils auf GitHub[1].

Dokumentumfang

Das vorliegende Dokument setzt grundlegende Kenntnisse über die Funktionsweise der eBill voraus. Grundsätzlich werden hier deshalb nur Abweichungen zur öffentlich zugänglichen Spezifikation des NWP-API der SIX[2] erwähnt. Es lohnt sich also, diese Spezifikation vorab ausreichend zu kennen.

Das SWP-API

Beim API gibt es naturgemäss einige Abweichungen zur Basis von SIX. Ein Beispiel gibt es beim Einliefern von Rechnungen. Vermutlich werden diese von allen NWPs nicht synchron an SIX eBill weitergeleitet, weshalb die zugehörige Response anders aussieht. Details zu den einzelnen Endpoints, ihren Parametern, Rückgabewerten und Vorbedingungen sind im yaml-File ersichtlich, das zur Schnittstellenempfehlung gehört.

BillerIDs

Die Verwendung der 17-stelligen, rein nummerischen PID (Party-ID) hat sich in der Schweiz im Umfeld der E-Rechnung seit über 20 Jahren sehr bewährt. Deshalb wird sie im Rahmen dieser Schnittstellenempfehlung für alle vorgeschrieben. NWPs müssen also für jeden seiner Biller so eine genormte Nummer erzeugen.

Die alphanummerischen BillerIDs, welche von der SIX eBill vergeben werden, sollen also nur zwischen NWP und SIX verwendet werden, und extern nicht offengelegt werden.

Das Format ist 17-stellig, rein nummerisch, und sieht wie folgt aus:

41NN00BBBBBBBBBPP

NN=NWP-ID, siehe unten
00=Diese beiden 0 sind empfohlen für zukünftige Erweiterungen, könnten wie B verwendet werden
BB=Generierte (zufällige oder fortlaufende) Nummer, mit welcher der NWP den Biller identifiziert
PP=Zwei Prüfziffern

Die beiden Prüfziffern können wie folgt berechnet werden (Java-Beispiel):

```
private static long calcBillerId(long billerIdWithoutChecksum) {
  billerIdWithoutChecksum = billerIdWithoutChecksum * 100;
  long checksum = 98 - (billerIdWithoutChecksum % 97); // modulo 97
  return billerIdWithoutChecksum + checksum;
}
```

Liste der NWP-IDs

4100: Reserviert
4101: SIX eBill Infrastruktur und SIX Paynet
4102: Reserviert (wurde aus Versehen ca. Mitte 2019 von SIX eBill Infrastruktur verwendet)
4109: AbaNet von Abacus
4110: PostFinance B2B / SIX eBill für migrierte B2C PostFinance-Kunden
4111: InvoCLOUD
4112: Epsitec
4130: Conextrade von Swisscom
4140: Pentag
4142: gate2b von io-market
4150: Pentag (doppelter Eintrag aus historischen Gründen)
4177: Swiss Post Solutions
4180-4199: Reserviert, dieser Bereich könnte später für 6-stellige NWP-IDs verwendet werden.
Stand: Mai 2020. NWP-IDs können hier[3] kostenlos bezogen werden.

Einlieferformate

Die Schnittstellenempfehlung definiert folgende mögliche Einlieferformate, welche alle teilnehmenden NWPs entgegennehmen müssen:

- ZUGFeRD[4], mit dem Profil EN16931 (entspricht dem EU E-Invoicing Standard[5] und ist identisch mit factur-x[6]), sowie mit den Profilen EXTENDED und BASIC-WL. Für Schweizer Spezialfälle wie QR- und ESR-Referenzen sollen die Empfehlungen der entsprechenden Schweizer GS1-Arbeitsgruppe[7] angewendet werden.
- FSCM-XML (bekannt als «Paynet XML»)[8]
- yellowbill Invoice[9]
- QR-Rechnung[10], zwingend mit dem «Alternativen Verfahren eBill[11]» sowie mit der Syntaxdefinition Swico[12]. Mit diesem «Swico String» können das Fälligkeitsdatum, die MWST-Sätze und -Beträge, Skonti sowie weitere Referenzen übermittelt werden. Diese Informationen sollen, wenn immer möglich, mit übergeben werden.

Bitte beachten Sie, dass es bei diesen Formaten individuelle Gebrauchs- bzw. Lizenzbestimmungen geben kann. Die Anwendung der Einlieferformate sollte aber im Zusammenhang mit dieser Schnittstellenempfehlung in jedem Fall kostenlos sein.

Allen Softwareherstellern, welche noch keines der anderen Formate implementiert haben, wird ZUGFeRD mit dem Profil EN16931 empfohlen.

Es gilt zu beachten, dass die QR-Rechnung und ZUGFeRD mit dem Profil BASIC-WL nicht für «echte» B2B-Rechnungen verwenden werden können, weil sie keine Positionsdaten enthalten können. Deshalb sollen diese nur dann eingesetzt werden, wenn es gute Gründe dafür gibt. Sonst könnte es sein, dass sich für B2B-Anwendungen grössere Nachteile ergeben.

Für alle ZUGFeRD-Profile, das FSCM-XML wie auch für yellowbill Invoice ist aktuell noch nicht definiert, welches Element für QR- und SCOR-Referenz, die E-Mail-Adresse oder die UID des Empfängers der eBill verwendet werden soll. Sobald dies bekannt ist, wird dies in einer kommenden Version des vorliegenden Dokumentes an dieser Stelle ergänzt.

Der Softwarehersteller sollte beim Einliefern der Rechnung das Header-Feld «bcFormat» mitgeben, um dem NWP mitzuteilen, in welchem Format die Rechnung vorliegt.

Ein PDF/A-3[13] ist in allen Fällen erforderlich. Es wird ohne Signatur an den NWP übermittelt. Es darf beliebige Fileattachments enthalten. Nach der Signatur durch den NWP darf die Grösse maximal 10 MB betragen. Allerdings wird dringend empfohlen, eine Obergrenze von etwa 3 oder 4 MB einzuhalten, da sonst die User Experience nicht ideal ist.

Onboarding JSON: Konfiguration der Software beim Biller

Die Aufschaltung des Billers erfolgt durch den NWP, nachdem dieser die Identität des Billers geprüft und entsprechender Vertrag unterzeichnet wurde.

Die Software beim Biller kann On-Premise installiert sein, in einer Cloud gehostet sein, oder er kann eine SaaS-Lösung («Online Fakturierung») seiner Wahl verwenden. In allen Fällen benötigt er diverse Konfigurationseinstellungen, wie z.B. seine 17-stellige BillerID, die URL des API des NWP, seinen initialen Autorisierungscode etc. Um auch hier ein standardisiertes Aufschaltprozedere anbieten zu können, wurde ein Onboarding JSON-File definiert, das alle relevanten Informationen für die Aufschaltung enthält. Der Softwarehersteller bzw. SaaS-Anbieter kann dieses als File speichern und mehrfach darauf zugreifen, oder er liest die Elemente einmalig in seine Stammdaten ein. Es sieht wie folgt aus:

{	
"version": "1.0",	Dateiversion, alle 1.* sind rückwärtskompatibel
"is_test": false,	Testsystem?
"expiration_date": "2020-02-20T23:59:59+01:00",	Ablaufzeitpunkt dieser Datei
"audience": "swp",	Für Biller und Empfänger
"party": {	
" id": "41097799999999999",	Name des Billers bzw. Empfängers
"name": "Muster AG",	Biller-ID bzw. Party-ID, wird vom NWP vergeben
"is_sender": true,	Sendet eBills?
"is_receiver": false,	Empfängt eBills? (Zukünftige Erweiterung)
"is_b2b_sender": false,	Darf B2B-Rechnungen senden?
"is_b2b_receiver": false	Kann B2B-Rechnungen empfangen?
},	
"nwp": {	
" id": "4109",	Name des NWP
" name": "Mein NWP",	ID des NWP, 4- bis 6-stellig, rein nummerisch
"logo_url": "www.mynwp.ch/logo.png",	URL auf Logo
"info_url": "www.mynwp.ch/about",	URL für Browser bei Klick auf Logo
"api_endpoint": {	
"url": "https://api.mynwp.ch/biller/v1",	Endpoint des SWP-API, z.B. für Rechnungsupload
"headers": ["Authorization: Bearer czZCaGmF0M2JW"]	Header-Felder, sollten so mitgegeben werden
}	
},	
"auth": {	
"issuer": "https://auth.mynwp.ch",	Basis-URL des IdP
"authorization_endpoint": {	
"params": {	
"code": "oaerhgergha0ghaergj",	Einmalig verwendbarer Autorisierungs-Code
"grant_type": "authorization_code",	Der verwendete OAuth-Grant Typ
"client_id": "https://ebill-swp.org",	Client-ID für das SWP-API
"redirect_uri": "tag:ebill-swp.org,2020:biller-onboarding-json"	URI für das Biller Onboarding
}	
},	
"token_endpoint": {	
"url": "https://auth.mynwp.ch/oauth/v1/token",	Basis-URL des IdP
"headers": ["Authorization: Bearer czZCaGmF0M2JW"]	Header-Felder, sollten so mitgegeben werden
}	
}	
}	

Nach erfolgter Authentisierung (siehe nächstes Kapitel) kann die Software des Billers den SWP-API Endpoint aufrufen, um z.B. Rechnungen hochzuladen oder Anmeldungen abzuholen. Die verfügbaren API Endpoints sind im yaml-File definiert und dokumentiert.

Für normale Aufschaltungen mit vorbereiteter Software ist kein Testbetrieb notwendig. Für Fälle, wo mit dem Testsystem (oftmals Integrationssystem genannt) des NWP kommuniziert werden soll, muss beim NWP ein Test-Onboarding-JSON angefordert werden. In diesem ist das Element «is_test» auf true gesetzt, und die Endpoints darin verweisen auf das Testsystem.

Authorization

Die Authorization funktioniert mit dem Authorization Code Grant von OAuth2. Dessen Grundidee ist, dass der Client sich mit einem sogenannten Access-Token beim API-Server autorisiert. Access-Tokens sollen dabei eine relativ kurze Gültigkeitsdauer haben (kleineres Window of Opportunity für Angreifer).

Kurz vor oder auch nach Ablauf der Gültigkeit eines Access-Tokens kann mittels des Refresh-Tokens beim IdP-Server ein neues Access-Token bezogen werden. Refresh-Tokens haben deshalb eine u.U. deutlich längere Lebensdauer als Access-Tokens oder, wenn der NWP das möchte, kein vordefiniertes Ablaufdatum. Ist das Refresh-Token aber abgelaufen, muss der Onboarding-Prozess neu durchgeführt werden.

In aller Regel soll der Client nur die Refresh-Tokens persistieren und bei Bedarf ein neues Access-Token beziehen.

Hinweis für OAuth2-Spezialisten
Bezüglich der Authorization gibt es eine Abweichung zum SIX NWP-API, weil NWPs im Gegensatz zu On-Premise-Installationen ein wirklich schützbares Client-Secret haben können. Ohne dieses ist der Client Credentials Grant nicht wirklich sicher verwendbar. Deshalb wird für das SWP-API der Authorization Code Grant des RFC 6749[14] verwendet. Da der Einsatz eines Browsers nicht in allen Fällen zielführend ist, wird der erste Teil des Authorization Code Grant durch das oben beschriebene Onboarding-JSON ersetzt. Der zweite Teil (ab «D» im RFC) verläuft nach Standard.

Bleibt die Frage, wie der Client zum ersten Refresh-Token kommt. Dies geschieht beim eBill SWP-API, indem der Client vom IdP mittels des initialen Autorisierungscodes ein solches anfordert.

Grundsätzlich gibt es für den Client zwei unterschiedliche Implementationsvarianten, eine ohne und eine mit OAuth2-Client Library.

Variante 1: Ohne OAuth-Client Library

Die Implementation ohne OAuth-Client Library ist sehr einfach, und wird empfohlen, falls keine weiteren Umstände für eine andere Lösung sprechen. Der Ablauf ist wie folgt:

1. Mit dem initialen Authorization Code das erste Refresh- und Access-Token abholen:

```
HTTPS-Aufruf:

URL:
<auth_url>

Methode:
POST

Header:
Content-Type: application/x-www-form-urlencoded
Accept: application/json

Body:
grant_type=authorization_code&client_id=<client_id>&redirect_uri=<redirect_uri>&
code=<authorization_code>
```

Beispiel mit Daten aus dem obigen Onboarding-JSON:

```
POST https://auth.mynwp.ch/oauth/v1/token
Content-Type: application/x-www-form-urlencoded
Accept: application/json

grant_type=authorization_code&client_id=https://ebill-swp.org&redirect_uri=tag:ebill-
swp.org,2020:biller-onboarding&code=oaerhgergha0ghaergj
```

Curl-Beispiel :

```
curl -X POST -H "Content-Type: application/x-www-form-urlencoded" -H "Accept: application/json"
-d "grant_type=
authorization_code&redirect_uri=tag:ebill-swp.org,2020:biller-onboarding-json&client_id=
https://ebill-swp.org&code=eyJhb...T3i0w" https://api.mynwp.ch/test/oauth/v1/token
```

Als Antwort kommt zurück :

```
HTTP/1.1 200
Content-Type: application/json

{
        "access_token": "eyJhbGciO.eyJqdGkiwMn0.qmQjb98g",
        "refresh_token": "eyJhbGciO.eyJqdggSrg53.3rgSJC34ef",
        "token_type": "Bearer",
        "expires_in": 600
}
```

Das zurückgegebene Access-Token ist in diesem Beispiel 600 Sekunden lang gültig. Es ist ratsam, einige Sekunden (z.B. 30) vor Ablauf ein neues Access-Token abzuholen. Das funktioniert wie folgt:

```
HTTPS-Aufruf:

URL:
<auth_url>

Methode:
POST

Header:
Content-Type: application/x-www-form-urlencoded
Accept: application/json

Body:
grant_type=refresh_token&refresh_token=<LetztesRefreshToken>
```

Beispiel:

```
POST https://auth.mynwp.ch/oauth/v1/token
Content-Type: application/x-www-form-urlencoded
Accept: application/json

grant_type=refresh_token&refresh_token= eyJhbGciO.eyJqdggSrg53.3rgSJC34ef
```

Curl-Beispiel :

```
curl -X POST -H "Content-Type: application/x-www-form-urlencoded" -H "Accept: application/json"
-d        "grant_type=refresh_token&client_id=https://ebill-swp.org&refresh_token=eyJhbG...fbIug"
https://api.myswp.ch/test/oauth/v1/token
```

Die Antwort ist die gleiche wie beim initialen Aufruf. Nur ist bei diesem Aufruf das refresh_token in der Antwort optional:

```
HTTP/1.1 200
Content-Type: application/json

{
        "access_token": "eyJhbGciO.eyJqdGkiwMn0.qmQjb98g",
        "refresh_token": "eyJhbGciO.eyJqdggSrg53.3rgSJC34ef",
        "token_type": "Bearer",
        "expires_in": 600
}
```

oder:

```
HTTP/1.1 200
Content-Type: application/json

{
        "access_token": "eyJhbGciO.eyJqdGkiwMn0.qmQjb98g",
        "token_type": "Bearer",
        "expires_in": 600
}
```

Variante 2: Mit OAuth-Client Library

Der SWP kann eine OAuth-Client Library einsetzen und diese ab «D» des erwähnten RFC verwenden. Als Input wird in diesem Fall die «issuer»-URL des Onboarding-JSON verwendet. Die Library wird dann «/.well-known/openid-configuration» aufrufen und sieht im Element «token_endpoint» die gleiche URL wie im «token_endpoint» des Onboarding-JSON-Files.

Weitere Schritte bei beiden Varianten

Zusammenfassend also nochmals: Der Authorization-Code und die Refresh-Tokens dürfen nur an die auth_url geschickt werden, die Access-Tokens nur an den nwp_api_endpoint.

Wird der nwp_api_endpoint mit einem abgelaufenen Access-Token aufgerufen, erhält der Client den HTTP Statuscode 401 (Unauthorized) zurück.

Zu beachten: In den Antworten des IdP ist das Refresh-Token ausser beim ersten Aufruf optional. Die Regel ist, dass die aufrufende Software ein erhaltenes refresh_token in jedem Fall speichert und dieses weiterverwendet. Falls der NWP nicht mit jedem Access-Token auch ein neues Refresh-Token mitsendet, hat er sicherzustellen, dass bei üblichen Usage-Patterns rechtzeitig ein neues Refresh-Token übermittelt wird.

Ebenso kann jeder NWP selbst entscheiden, ob er nur das allerletzte Refresh-Token akzeptiert, oder auch noch die n letzten oder alle, welche innerhalb einer bestimmten Frist ausgestellt wurden. Dies kann z.B. bei On-Premise Installation sinnvoll sein, wenn ein Backup zurückgeladen werden muss. Wie hoch n ist bzw. ob es bei Refresh-Tokens überhaupt eine Gültigkeitsfrist gibt und wie lange diese ist, kann jeder NWP selbst entscheiden.

Die zurückgegebenen Tokens können signierte JWTs sein, müssen aber nicht. Die aufrufende Software soll die Tokens einfach als String behandeln und unverändert bei den entsprechenden Aufrufen weitergeben. Generell wird empfohlen, keine Annahmen über die maximale Länge von Refresh- bzw. Access-Tokens zu machen. Falls dies trotzdem unumgänglich sein sollte, ist man mit einer maximalen Grösse von 16 KiB für die nächsten paar Jahre vermutlich auf der sicheren Seite.

Refresh-Tokens haben beim eBill SWP-API eine garantierte Mindestlebensdauer von 90 Tagen, damit z.B. auch Vereine, welche z.B. nur einmal im Monat neue Anmeldungen herunterladen, und nur einmal pro Jahr Rechnungen versenden, nicht jedes Mal beim NWP ein neues Onboarding-JSON anfordern müssen.

Die Lebensdauer der Access-Tokens ist wesentlich geringer, 10 Minuten sind empfohlen. D.h. der SWP muss damit klarkommen, dass z.B. beim Hochladen von 50'000 Rechnungen zwischenzeitlich ein neues Access-Token angefordert und verwendet werden muss.

B2B-Meldungen zwischen NWPs

Einige NWPs unterstützen auch den B2B-Austausch von Rechnungen und Gutschriften untereinander. Dies ist im Rahmen dieser API-Empfehlung ausdrücklich optional, wobei einzelne NWPs entscheiden können, nur den Versand zuzulassen, andere auch den Empfang.

SWPs, welche B2B-Rechnungen und Gutschriften versenden wollen, müssen nichts Spezielles unternehmen. Die PID des Empfängers bezeichnet statt eines eBill Teilnehmers dann einen B2B-Empfänger. Es ist zu beachten, dass die PIDs, welche in den Stammdaten eingetragen werden, gemäss obiger Definition validiert werden, und nicht nur «4101» und «4110» zugelassen sind. Und es wird stark empfohlen, ein Einlieferformat zu verwenden, das Positionsdaten enthalten kann.

SWPs, welche auch Rechnungen empfangen können, können die entsprechenden Endpoints aufrufen, falls im Onboarding-JSON das is_b2b_receiver-Element auf true gesetzt ist:

"is_b2b_receiver": true,

Die entsprechenden Endpoints sind im yaml-File definiert. Mit dem bcFormat-Headerfeld kann der SWP festlegen, welches Format er empfangen will. Ein PDF wird in allen Fällen mit ausgeliefert.

Nächste Schritte, weitere Versionen

Die Arbeitsgruppe wird zukünftige Erweiterungen des SIX NWP-API für neue Funktionen zeitnah in diese Schnittstellenempfehlung einfliessen lassen. Ausserdem wird überlegt, weitere Funktionen anzubieten, wie z.B. im Bereich separate Attachments, Info über die durch den NWP unterstützten Formate und Biller-Aufschaltung. Letzteres ist insbesondere, aber nicht nur, für Online-Fakturierungslösungen sinnvoll.

Die Arbeitsgruppe bedankt sich herzlich für die unterstützende Begleitung durch:

SIX, Swico und GS1.

Mitglieder der Arbeitsgruppe:

Versionierung dieses Dokumentes

Version	Datum	Beschreibung	Autoren
0.1	30.10.2019	Initialversion	Nicolas Guillet, Abacus Research AG
0.4	31.01.2020	Ergänzungen zum Onboarding	Nicolas Guillet, Abacus Research AG, mit Unterstützung von Pierre Arnaud, Epsitec SA
0.5	18.02.2020	Erweiterung um B2B	Eva Sediki, Nintu Informatik AG
0.6	21.04.2020	Erweiterung um Authorization	Nicolas Guillet, Abacus Research AG

[1] https://github.com/swico/ebill-swp-api
[2] https://www.ebill.ch/de/home/network-partners.html#technical-documentation
[3] Per Mail bei nicolas.guillet@abacus.ch
[4] https://www.ferd-net.de/zugferd
[5] https://invoice.fans/de/en16931/
[6] http://fnfe-mpe.org/factur-x/
[7] https://shop.gs1.ch/de/A~16970/3~511~3/Elektronische-Hybridrechnung-PDF-mit-XML-Basierend-auf-ZUGFeRD-Factur-X
[8] Folgt in Kürze: https://github.com/swico/ebill-swp-api/documents/fscm-xml
[9] https://www.postfinance.ch/de/unternehmen/produkte/debitorenloesungen/e-rechnung-rechnungssteller.html
[10] http://paymentstandards.ch/de/shared/news/2019/ig.html
[11] https://www.ebill.ch/dam/downloads/specifications/d0543-de-01-spez-nutzung-alternatives-verfahren-ebill-swiss-qr-code.pdf
[12] http://paymentstandards.ch/de/shared/news/2019/ig.html
[13] https://de.wikipedia.org/wiki/PDF/A#PDF/A-3
[14] https://www.rfc-editor.org/rfc/pdfrfc/rfc6749.txt.pdf