---DRAFT Version 0.7---

SIX eBill SWP API: Recommendation for the Interface between Software Manufacturers and Network Partners of SIX eBill

Introduction

To date, many software manufacturers have implemented interfaces with PostFinance (for e-finance customers) and Paynet (for all other banks). SIX's eBill redesign is introducing a network partner model that allows additional network partners (NWPs) to send e-invoices directly to the SIX Infrastructure as eBills with no restriction to the previous delivery channels. Individual (even major) invoice issuers cannot deliver eBills directly to SIX themselves.

There are currently around a dozen NWPs, with more likely to join. This interface recommendation is intended to ensure that software manufacturers that wish to give their customers a free choice of NWP do not have to implement so many different interfaces. It is based on REST with OAuth 2.0 and above all on SIX's NWP API.

It is voluntary for NWPs and for software manufacturers. If fully implemented, the following statement may be added to the product information: "Supports the SIX eBill SWP API recommendation".

The current members of the working group that devised this recommendation aim to subject the responsibility for this recommendation to meaningful governance, and are seeking a suitable framework for this. In the future, therefore, a license, declaration or agreement may be necessary in order to use this recommendation. However, the intention is definitely to keep it free of charge for NWPs and for software manufacturers.

The current version of this interface recommendation is available on GitHub[1].

Scope of the document

This document assumes that the reader has basic knowledge about the workings of eBill. As a rule, therefore, only deviations from the publicly available specifications of SIX's NWP API[2] are mentioned here. It would therefore be useful to be sufficiently familiar with these specifications in advance.

The SWP API

In the API, there are naturally a few deviations from the SIX basis. One example relates to the delivery of invoices. These are presumably not forwarded to SIX eBill synchronously by all NWPs, so the associated response looks different. Details on the individual endpoints, their parameters, return values and preconditions can be found in the YAML file that is part of the interface recommendation.

Invoice issuer IDs

The use of the 17-digit, purely numerical PID (Party ID) in the electronic invoice environment has proved very successful in Switzerland for over 20 years. This interface recommendation therefore makes it mandatory for everyone. NWPs must therefore generate such a standardized number for each of their invoice issuers.

The alphanumeric invoice issuer IDs, assigned by SIX eBill, are therefore only to be used between the NWP and SIX, and not disclosed externally.

The format is 17 digits, purely numerical, and looks like this:

41NN00BBBBBBBBBPP

NN=NWP ID, see below
00=These two 0s are recommended for future expansions, could be used like B
BB=Generated (random or consecutive) numbers, with which the NWP identifies the invoice issuer
PP=Two check digits

The two check digits can be calculated as follows (Java example):

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

List of NWP IDs

4100: Reserved
4101: SIX eBill Infrastructure and SIX Paynet
4102: Reserved (was used accidentally by SIX eBill infrastructure around the middle of 2019)
4109: AbaNet from Abacus
4110: PostFinance B2B / SIX eBill for migrated B2C PostFinance customers
4111: InvoCLOUD
4112: Epsitec
4130: Conextrade from Swisscom
4140: Pentag
4142: gate2b from io-market
4150: Pentag (double entry for historical reasons)
4177: Swiss Post Solutions
4180-4199: Reserved, this range could be used later for 6-digit NWP IDs.
As of May 2020. NWP IDs can be obtained free of charge here[3].

Delivery formats

The interface recommendation defines the following possible delivery formats that all participating NWPs must accept:

- ZUGFeRD[4], with the profile EN16931 (complies with the EU e-invoicing standard[5] and is identical to factur-x[6]), and the profiles EXTENDED and BASIC-WL. For Swiss special cases such as QR and payment slip references, the recommendations of the corresponding Swiss GS1 working group[7] are to be applied.
- FSCM-XML (known as "Paynet XML")[8]
- yellowbill Invoice[9]
- QR-bill[10], obligatorily with the "alternative procedure eBill[11]" and with the Swico syntax definition[12]. This "Swico String" allows the due date, VAT rates and amounts, discounts and other references to be transmitted. This information should be included whenever possible.

Please note that these formats may have individual terms of use and license terms. However, the use of the delivery formats in connection with this interface recommendation should always be free of charge.

ZUGFeRD with the profile EN16931 is recommended for any software manufacturers that have not yet implemented one of the other formats.

It should be noted that the QR-bill and ZUGFeRD with the profile BASIC-WL cannot be used for "true" B2B invoices because they cannot include item data. Therefore, they should only be used when there are good reasons for it. Otherwise, significant disadvantages could arise for B2B applications.

For all ZUGFeRD profiles, FSCM-XML and yellowbill Invoice, it has not yet been defined which element is to be used for the QR and SCOR reference, the e-mail address or the corporate identification number of the recipient of the eBill. As soon as this is known, it will be added here in a future version of this document.

On delivery of the invoice, the software manufacturer should provide the header field "bcFormat" in order to notify the NWP of the format of the invoice.

A PDF/A-3[13] is required in all cases. It is transmitted to the NWP without a signature. It may contain any number of file attachments. The size must not exceed 10 MB after being signed by the NWP. However, it is strongly recommended to keep to a limit of around 3 or 4 MB, as otherwise the user experience is not ideal.

Onboarding JSON: configuration of the software at the invoice issuer

The invoice issuer is activated by the NWP, after the latter has verified the identity of the invoice issuer and a corresponding agreement has been signed.

The software at the invoice issuer can be installed on the premises or hosted in a cloud, or the invoice issuer can use a SaaS solution ("online invoicing") of its choice. In all cases, it needs various configuration settings, such as its 17-digit invoice issuer ID, the URL of the NWP's API, its initial authorization code, etc. In order also to offer a standardized activation procedure here, an onboarding JSON file has been defined that contains all relevant information for the activation. The software manufacturer or SaaS provider can save this as a file and access it repeatedly, or feed the elements into its master data once. It looks like this:

{	
"version": "1.0",	File version, all with 1.* are backward compatible
"is_test": false,	Test system?
"expiration_date": "2020-02-20T23:59:59+01:00",	Expiration date of this file
"audience": "biller",	For invoice issuer and recipient
"party": {	
"id": "41097799999999999",	Invoice Issuer ID or party ID, assigned by NWP
"name": "Muster AG",	Name of the invoice issuer or recipient
"is_sender": true,	Sends eBills?
"is_receiver": false,	Receives eBills? (Future expansion)
"is_b2b_sender": false,	Allowed to send B2B invoices?
"is_b2b_receiver": false	Able to receive B2B invoices?
},	
"nwp": {	
"id": "4109",	ID of the NWP, 4 to 6 digits, purely numerical
"name": "Mein NWP",	Name of the NWP
"logo_url": "https://www.mynwp.ch/logo.png",	URL to logo
"info_url": "https://www.mynwp.ch/about",	URL for browser after clicking on logo
"api_endpoint": {	
"url": "https://api.mynwp.ch/biller/v1",	Endpoint of the SWP API, e.g. for invoice upload
"headers": ["X-NWP-Foo: bar"]	Optional header fields
}	
},	
"auth": {	
"issuer": "https://auth.mynwp.ch",	Basis URL of the IdP
"authorization_endpoint": {	
"params": {	
"code": "oaerhgergha0ghaergj",	One-time authorization code
"grant_type": "authorization_code",	The OAuth grant type used
"client_id": "https://ebill-swp.org",	Client ID for the SWP API
"redirect_uri": "tag:ebill-swp.org,2020:biller-onboarding-json"	URI for invoice issuer onboarding
}	
},	
"token_endpoint": {	
"url": "https://auth.mynwp.ch/oauth/v1/token",	Basis URL of the IdP
"headers": ["Authorization: Bearer czZCaGmF0M2JW"]	Header fields, if provided
}	
}	
}	

After authentication (see next chapter), the invoice issuer's software can request the SWP API endpoint in order, for example, to upload invoices or fetch registrations. The available API endpoints are defined and documented in the YAML file.

For normal activations with prepared software, no test operation is necessary. In cases requiring communication with the NWP's test system (often called an integration system), a test onboarding JSON must be requested from the NWP. In this file, the element "is_test" is set to true, and the endpoints therein refer to the test system.

Authorization

Authorization works with the Authorization Code grant type from OAuth 2.0. The basic idea is that the client authorizes itself with the API server using an access token. Access tokens are intended to have a relatively short period of validity (smaller window of opportunity for attackers).

Shortly before or after an access token expires, a refresh token can be used to obtain a new access token from the IdP server. In some cases, refresh tokens therefore have a much longer lifetime than access tokens or, if the NWP wishes, no defined expiry date. If the refresh token has expired, however, the onboarding process must be carried out again.

As a rule, the client should only persist refresh tokens and obtain a new access token if necessary.

Note for OAuth 2.0 specialists
With regard to authorization, there is a deviation from the SIX NWP API, because NWPs, in contrast to on-premises installations, can have a truly secure client secret. Without this, the Client Credentials grant type cannot really be used safely. This is why the Authorization Code grant type of RFC 6749[14] is used for the SWP API. As the use of a browser is not expedient in all cases, the first part of the Authorization Code grant type is replaced by the onboarding JSON described above. The second part (from "D" in RFC) is as standard.

This leaves the question how the client gets the first refresh token. In the eBill SWP API, the client requests one from the IdP using the initial authorization code.

For the client, there are two different implementation versions, one with and one without OAuth 2.0 client library.

Version 1: without an OAuth client library

Implementation without an OAuth client library is very simple and is recommended if no other circumstances require a different solution. The procedure is as follows:

1. Fetch the first refresh and access token with the initial authorization code:

```
HTTPS request:

URL:
<auth_url>

Method:
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>
```

Example with data from the above 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 example:

```
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
```

The response is as follows:

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

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

In this example, the returned access token is valid for 600 seconds. It is advisable to fetch a new token a few seconds (e.g. 30) before expiry. This works as follows:

```
HTTPS request:

URL:
<auth_url>

Method:
POST

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

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

Example:

```
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 example:

```
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.mynwp.ch/test/oauth/v1/token
```

The answer is the same as for the initial request, except in this request the refresh_token in the response is 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
}
```

or:

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

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

Version 2: with an OAuth client library

The SWP can utilize an OAuth client library and use it from “D” of the above-mentioned RFC. In this case, the “issuer” URL of the onboarding JSON is used as input. The library will then request “/.well-known/openid-configuration” and see the same URL in the “token_endpoint” element as in the “token_endpoint” of the onboarding JSON file.

Further steps for both versions

To summarize again: The authorization code and the refresh tokens must be sent only to auth_url, the access tokens only to nwp_api_endpoint.

If nwp_api_endpoint is requested with an expired access token, the client gets back the HTTP status code 401 (unauthorized).

Please note: In the IdP responses, the refresh token is optional except for in the first request. The rule is that the requesting software always saves a received refresh_token and uses it again. If the NWP does not also send a new refresh token with every access token, it must ensure that a new refresh token is transmitted in due time in the case of typical usage patterns.

Similarly, each NWP can decide for itself whether to accept only the very last refresh token or also the n last or all that were issued within a defined period. This can be useful for on-premises installations, for example, if a backup has to be reloaded. Each NWP can decide for itself the value of n or if there is even a validity period for refresh tokens, and how long it is.

The returned tokens can be signed JWTs but do not have to be. The requesting software should simply treat the tokens as a string and forward them unchanged in the corresponding requests. In general, it is recommended not to make assumptions about the maximum length of refresh or access tokens. If this is unavoidable, however, assuming a maximum size of 16 KB will probably put you on the safe side for the next few years.

In the eBill SWP API, refresh tokens have a guaranteed minimum lifetime of 90 days so that associations, for example, which might only download new registrations once a month and only send invoices once a year, do not have to request a new onboarding JSON from the NWP each time.

The lifetime of the access tokens is much shorter; 10 minutes is recommended. This means the SWP must be able to cope with the fact that a new access token must be requested and used during an upload of 50,000 invoices, for example.

B2B messages between NWPs

Some NWPs also support the B2B exchange of invoices and credits. In this API recommendation, this is explicitly optional, with individual NWPs being able to decide only to permit sending, others also receipt.

SWPs that want to send B2B invoices and credit notes do not have to do anything specific. The recipient's PID then denotes a B2B recipient instead of an eBill participant. It should be noted that the PIDs entered into the master data are validated according to the above definition, and not only "4101" and "4110" are permitted. And it is strongly recommended to use a delivery format that can contain item data.

SWPs that can also receive invoices can request the corresponding endpoints if the is_b2b_receiver element in the onboarding JSON is set to true:

"is_b2b_receiver": true,

The corresponding endpoints are defined in the YAML file. With the bcFormat header field, the SWP can define which format it wants to receive. A PDF is also delivered in all cases.

Next steps, future versions

The working group intends to promptly add future expansions of the SIX NWP API for new functions to this interface recommendation. In addition, it is considering offering additional functions, for example with regard to separate attachments, information about the formats supported by the NWP, and invoice issuer activation. The latter is especially, but not exclusively, useful for online invoicing solutions.

The working group expresses its heartfelt thanks for the support from:

SIX, Swico and GS1.

Members of the working group:

Versions of this document

Version	Date	Description	Authors
0.1	30 Oct. 2019	Initial version	Nicolas Guillet, Abacus Research AG
0.4	31 Jan. 2020	Additions regarding onboarding	Nicolas Guillet, Abacus Research AG, with support from Pierre Arnaud, Epsitec SA
0.5	18 Feb. 2020	B2B added	Eva Sediki, Nintu Informatik AG
0.6	21 Apr. 2020	Authorization added	Nicolas Guillet, Abacus Research AG
0.7	10 Jul. 2020	Minor corrections	Eva Sediki, Nintu Informatik AG

[1] https://github.com/swico/ebill-swp-api
[2] https://www.ebill.ch/de/home/network-partners.html#technical-documentation
[3] By e-mail from 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] To follow soon: 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