Specification

"Real-time notifications"

Version 1.0 from 17.07.2019

Final Version
Table of contents

1. INTRODUCTION 3

2. TECHNICAL ARCHITECTURE OF THE PROCESS 4
   2.1. Using the WebSocket Protocol for Real-Time Notifications 4
   2.2. Data for establishing a connection on the client side 5
   2.3. Header description for the https handshake 7

3. STRUCTURE OF REAL-TIME NOTIFICATIONS 8
   3.1. Notification of new data on the EBICS bank server 8
   3.2. General information 10

4. DISTINCTION EBICS SPHERE - WEBSOCKET CONNECTION 12
   4.1. Actions on bank side 13
   4.2. Actions on customer side 13
1. Introduction

In conjunction with the introduction of Instant Payments it has become apparent that the possibility of processing payments in real time is not only a significant gain in convenience for private individuals, for example through payment options with a smartphone app, but that it also offers immense advantages for corporate customers.

While payment processes are still predominantly executed as time-consuming batch processes with cut-off times to be completed most ERP systems are already real-time systems today. In addition to the finality of a payment in real time it is the immediate feedback regarding the payment success that is particularly attractive for corporate customers on the basis of which further actions can be triggered in a business process chain (e.g. delivery of goods). Therefore, it is essential that not only the payment is processed in real time, but the notification of the payment success, too. A solution for this must be created for corporate customers.

Typically, corporate customers use the EBICS standard in electronic payment transactions with banks which is supported by all banks and savings banks and guarantees the secure transmission of payments and account information via internet. Designed as a client-server procedure, the initiative for transmission always starts from the customer or client side. Customers send payments and/or collect data. The bank side never sends data to the customer on its own initiative. For the desired real-time notification of customers in connection with instant payments this means that customers must actively ask via EBICS whether corresponding information is available for collection via EBICS. This inevitably leads to the problem that EBICS customer systems poll the bank servers at regular short intervals (polling), which can lead to an overloading of the bank systems due to the high and unnecessary overhead communication. To prevent this it could be a solution to modify EBICS in such a way that information (push service) can be actively transmitted to customers via EBICS. However, such a solution would mean that banks would have to establish active EBICS connections to the customer’s sphere which would entail a multitude of security questions to be solved and would lead to a complex bank-side administration of EBICS customer systems.

In order to avoid a complex and fundamental change of the EBICS standard on the one hand and to prevent the creation of more institute-specific solutions for the transmission of real-time information that are not compatible with each other on the other hand, the German banking industry has decided to set up a standardization on the basis of the combination of two systemically independent processes. For this purpose the use of the secure and established EBICS procedure for the collection of banking data should remain and modern web technology should be used in order to be able to trigger the EBICS collection by the customer from received events. For this the established internet standard "WebSocket" is used which offers a permanent bi-directional channel between corporate customer and bank, built from the client side and secured by TLS. Via this channel the banks can notify the confirmation of position (available for collection) of real-time information. The customer can then download the banking-relevant real-time information securely via the proven EBICS protocol.

The solution approach described above offers a combination of established internet standards with the electronic banking standard EBICS established in Europe for the secure transmission of real-time notifications to corporate customers. First and foremost, this procedure is intended for the
transmission of the credit advice which informs the corporate customer about the success of a real-time payment in real time. However, the procedure is also suitable for supporting other future business processes that require the transmission of real-time notifications.

Due to the systemic separation of EBICS and WebSocket functionality the design of the solution specified in more detail below basically offers the possibility of using existing IT-infrastructures, both in the banking and in the customer sphere. The secure EBICS infrastructure is used without a change for the transmission of sensitive banking data.

2. Technical architecture of the process

2.1. Using the WebSocket Protocol for Real-Time Notifications

The wss protocol is used for the possibility of sending messages to a client on the server side (push functionality). The concrete description of the wss protocol can be found in RFC 64551.

While the server reacts to requests from a client with a https connection (Client Request / Server Response), the client opens the (or a) connection to the server with the WebSocket protocol. The server can then use this open connection to actively (i.e. without a concrete request from the client) deliver information to the client. The procedure is bidirectional, i.e. the client can also send replies. For the application described here, however, only messages from the server to the client are sent. Internet encryption TLS (wss connection) is used for secure data transmission.

The number of sessions (connections) for a customer with his bank is not limited. Several sessions are required, for example, for separate customer systems. However, the notifications are customer-oriented, i.e. if there are several connections between a customer and a bank, messages will also be transmitted several times to the customer. Messages to a customer are always sent to the WebSocket connections of the customer existing at the time of sending. If a customer does not have an open connection at the time a message is sent, the bank will decide whether to send accumulated push messages as soon as interrupted connections are available again.

1 https://tools.ietf.org/html/rfc6455
2.2 Data for establishing a connection on the client side

In order to establish a wss connection to the server, the customer must be provided with the necessary connection data.

One possibility for EBICS clients is to retrieve the data via the EBICS download standard request (without specifying a period start date). However, the banks are free to communicate the access data to the customers via alternative channels. For the EBICS customer in the considered initial use case, the use of the secure EBICS communication for the transmission of access data makes sense. As an EBICS response, the client receives the user data in JSON format.

The following EBICS business transaction is defined:

**EBICS V 3.0 / BTF**: The assignment of the element group `<Service>` is as follows:

<table>
<thead>
<tr>
<th>Service-Name</th>
<th>Scope</th>
<th>Option</th>
<th>MsgNm</th>
<th>Container</th>
<th>Description</th>
<th>Up-/Download</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTH</td>
<td>DE</td>
<td></td>
<td>wssparam</td>
<td></td>
<td>Retrieving data for establishing a connection to a wss session in JSON format</td>
<td>D</td>
</tr>
</tbody>
</table>

**EBICS V 2.5 / Technical OrderType**: The order type WSS is used.

<table>
<thead>
<tr>
<th>OrderType</th>
<th>Direction of transfer</th>
<th>Text</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>WSS</td>
<td>D</td>
<td>Retrieving data for establishing a connection to a wss session in JSON format</td>
<td>JSON-Format</td>
</tr>
</tbody>
</table>

The response contains (regardless of the EBICS version used) under `ebicsResponse/body/DataTransfer/OrderData` a JSON file with the following contents, which is zipped and encrypted according to the EBICS specification:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Explanations</th>
<th>Occurences</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>URL of the access</td>
<td></td>
<td>[1..1]</td>
</tr>
<tr>
<td>TOKEN</td>
<td>Security token Usable during period of validity</td>
<td>Password for the wss connection setup. The requirement is that the password must be unique. It is recommended to use a UUID (Version 4, i.e. random number) according to RFC 4122 with the length of 16 bytes.</td>
<td>[1..1]</td>
</tr>
<tr>
<td>OTT</td>
<td>One time token</td>
<td>Can this token be used only once (value: Y)?</td>
<td>[1..1]</td>
</tr>
</tbody>
</table>

| **VALIDITY** | end of validity to set up the wss connection by access token | If the connection is terminated, this can be restarted with the security token as long as the end of validity is not reached. The format is ISO DateTime with time zone UTC | [1..1] |
| **PARTNERID** | Customer reference | For EBICS customers we recommend the EBICS Customer ID | [1..1] |
| **USERID** | User reference | For EBICS customers we recommend the relevant EBICS Subscriber ID. In case of a request using a SystemID (technical user), the Subscriber ID of a bank-technical User is entered here - if available, otherwise the SystemID. | [0..1] |

All values in JSON-file are character strings.

Example:

```
{
    "URL": "https://bankmitwebsocket.de",
    "TOKEN": "550e8400-e29b-11d4-a716-446655440000",  "OTT": "N",
    "VALIDITY": "2019-03-21T10:35:22Z",  "PARTNERID": "K1234567",  "USERID": "USER4711"
}
```

The usual EBICS error codes for a standard download are possible. In particular, this business transaction is not set up for the "historical" download.
2.3 Header description for the https handshake

Premise: Due to the high compatibility with the different programming languages (Java, Javascript, Ajax,...) and High Level Frameworks like Spring-Boot, one should use http-Basic Authentication RFC 2617\(^3\) when submitting credentials.

The following header is defined there:

Authorization: Basic <base64encoded Credential>

The credential usually consists of User:Password and is therefore made up of the Key values of PARTNERID, if applicable USERID and TOKEN in the form PARTNERID_USERID:TOKEN (see section 2.2).

If only one customer reference is used, the separator "_" is also omitted.

For the example in section 2.2, the following results for the header:

- Credential = K1234567_USER4711:550e8400-e29b-11d4-a716-446655440000
- base64encoded Credential= SzEyMzQ1NjdVvVNFUjQ3MTE6NTUwZTg0MDAtZTI5Yi0xMWQ0LWE3MTYtNDQ2NjU1NDQwMDAw
- Authorization: Basic SzEyMzQ1NjdVvVNFUjQ3MTE6NTUwZTg0MDAtZTI5Yi0xMWQ0LWE3MTYtNDQ2NjU1NDQwMDAw

3 Structure of real-time notifications

Real-time notifications delivered to the customer within a wss session, are encoded in JSON format. There are two classes of messages:

1. notification of new data on the EBICS bank server as well as
2. general information (broadcast messages) for customers

The contents and structure in JSON format are described in the following subchapters.

All values in the JSON file are strings (UTF-8).

3.1 Notification of new data on the EBICS bank server

<table>
<thead>
<tr>
<th>Level</th>
<th>Name</th>
<th>Description</th>
<th>Explanations</th>
<th>Occurences</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MCLASS</td>
<td>Notification class as an array with the key information NAME VERS TIMESTAMP</td>
<td>For the current class all the time &quot;EBICS-HAA&quot; With the administrative EBICS order type HAA you could call the the same information - the JSON file is therefore the Push counterpart to the appropriate administrative EBICS business transaction</td>
<td>[1..1]</td>
</tr>
<tr>
<td>2</td>
<td>NAME</td>
<td>Name of notification class</td>
<td></td>
<td>[1..1]</td>
</tr>
<tr>
<td>2</td>
<td>VERS</td>
<td>Format version of this message class</td>
<td>This specification begins for the “EBICS-HAA” class with 1.0</td>
<td>[1..1]</td>
</tr>
<tr>
<td>2</td>
<td>TIMESTAMP</td>
<td>Time of delivery</td>
<td>In the case that no active wss connection to the customer is available, and however data has to be delivered later when it becomes available, this is the time of the first attempt by the bank to make the data available. The format is ISO DateTime with time zone UTC.</td>
<td>[1..1]</td>
</tr>
<tr>
<td>1</td>
<td>PARTNERID</td>
<td>EBICS customer reference</td>
<td>The unique EBICS identification of the customer.</td>
<td>[1..1]</td>
</tr>
<tr>
<td></td>
<td>USERID</td>
<td>EBICS user reference</td>
<td>The clear and unambiguous EBICS identification of the user on the EBICS bank server</td>
<td>[0..1]</td>
</tr>
<tr>
<td>---</td>
<td>--------</td>
<td>----------------------</td>
<td>---------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>1</td>
<td>BTF</td>
<td>List of BTF parameter groups as arrays</td>
<td>BTF parameter groups to which Information on collection (by EBICS version 3.0) are already available</td>
<td>[0..1]</td>
</tr>
<tr>
<td>2</td>
<td>SERVICE</td>
<td>The names for these keywords are sent to the corresponding Data element/Attribute names based on the EBICS specification.</td>
<td>Values for these keys result in from the EBICS specification (especially external BTF code list)</td>
<td>[1..1]</td>
</tr>
<tr>
<td>2</td>
<td>OPTION</td>
<td></td>
<td></td>
<td>[0..1]</td>
</tr>
<tr>
<td>2</td>
<td>CONTTYPE</td>
<td></td>
<td></td>
<td>[0..1]</td>
</tr>
<tr>
<td>2</td>
<td>MSGNAME</td>
<td></td>
<td></td>
<td>[0..1]</td>
</tr>
<tr>
<td>2</td>
<td>VARIANT</td>
<td></td>
<td></td>
<td>[0..1]</td>
</tr>
<tr>
<td>2</td>
<td>VERSION</td>
<td></td>
<td></td>
<td>[0..1]</td>
</tr>
<tr>
<td>2</td>
<td>FORMAT</td>
<td></td>
<td></td>
<td>[0..1]</td>
</tr>
<tr>
<td>1</td>
<td>ORDERTYPE</td>
<td>List of Ordertypes as array</td>
<td>Order types for which Information on collection (by EBICS version 2.5). [&quot;OrderType1&quot;, &quot;OrderType2&quot;, ... &quot;OrderTypeN&quot;]</td>
<td>[0..1]</td>
</tr>
</tbody>
</table>

**Note on the occurrence of BTF and ORDERTYPE:** At least one of these optional Arrays must be delivered with the message class EBICS-HAA!

**Example 1:**
Advice of a real-time transfer receipt:

```json
{
    "MCLASS": {
        "NAME": "EBICS-HAA", "VERS": "1.0",
        "TIMESTAMP": "2019-05-13T12:21:50Z"
    },
    "PARTNERID": "K1234567", "USERID": "USER471",
    "BTF": {
        "SERVICE": "REP",
        "SCOPE": "DE",
        "CONTTYPE": "ZIP", "MSGNAME": "camt.054"
    },
    "ORDERTYPE": ["C5N"]
}
```
Example 2:
New account notifications and a payment status report for instant payments from the customer

```json
{
  "MCLASS": [  
    {"NAME": "EBICS-HAA", "VERS": "1.0",  
  ],  
  "PARTNERID": "K1234567", "USERID": "USER471",  
  "BTF": [  
    {"SERVICE": "REP",  
      "SCOPE": "DE",  
      "CONTTYPE": "ZIP", "MSGNAME": "camt.052"  
    },  
    {"SERVICE": "REP",  
      "SCOPE": "DE", "OPTION": "SCI",  
      "CONTTYPE": "ZIP", "MSGNAME": "pain.002"  
    }  
  ],  
  "ORDERTYPE": ["C52","CIZ"],
}
```

### 3.2 General information

<table>
<thead>
<tr>
<th>Level</th>
<th>Name</th>
<th>Description</th>
<th>Explanations</th>
<th>Occurences</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MCLASS</td>
<td>Notification class as an array with the key information NAME VERS TIMESTAMP</td>
<td></td>
<td>[1..1]</td>
</tr>
<tr>
<td>2</td>
<td>NAME</td>
<td>Name of notification class</td>
<td>For the current class all the time &quot;INFO&quot;</td>
<td>[1..1]</td>
</tr>
<tr>
<td>2</td>
<td>VERS</td>
<td>Format version of this message class</td>
<td>This specification begins for the &quot;INFO&quot; class with 1.0</td>
<td>[1..1]</td>
</tr>
<tr>
<td>2</td>
<td>TIMESTAMP</td>
<td>Time of delivery</td>
<td>In the case that no active wss connection to the customer is available, and however data has to be delivered later when it becomes available, this is the time of the first attempt by the bank to make the data available. The format is ISO DateTime with time zone UTC.</td>
<td>[1..1]</td>
</tr>
<tr>
<td>1</td>
<td>INFO</td>
<td>Free text as array with key information LANG FREE</td>
<td>The array must contain at least one key pair LANG / FREE</td>
<td>[1..1]</td>
</tr>
<tr>
<td>2</td>
<td>LANG</td>
<td>Language of the free text</td>
<td>ISO 2A</td>
<td>[1..1]</td>
</tr>
<tr>
<td>2</td>
<td>FREE</td>
<td>Free text message</td>
<td>The message can theoretically be of any length</td>
<td>[1..1]</td>
</tr>
</tbody>
</table>
Example 3:
Example for general information:

```json
{
  "MCLASS": [  
    {"NAME": "INFO", "VERS": "1.0", "TIMESTAMP": "2019-03-25T12:25:34Z"}
  ],
  "INFO": [  
    {"LANG": "EN", "FREE": "The EBICS-Service is limited on 30.03.2019 from 10:00 a.m. - 11:00 a.m. due to maintenance work "}
  ]
}
```
4. Distinction EBICS Sphere - WebSocket Connection

Via the WebSocket connection an EBICS customer can be informed in real time about new available files on the EBICS bank server (see section 3.1).

Thus the WebSocket connection is a supplement to the EBICS communication but is technically completely separated from the EBICS procedure.

The structure of the WebSocket connection is shown in the following graphic with the single actions numbered in chronological order and for the various spheres displayed in different colors (blue = EBICS, red = WebSocket, green = interaction WebSocket and EBICS in the customer sphere):

As an example of a real-time notification the following graphic shows the notification of an instant payment input.

Again, the single actions are numbered consecutively according to their chronological order and displayed in different colours for the different spheres (blue = EBICS, red = WebSocket, green = interaction between WebSocket and EBICS in the customer sphere or in the bank sphere):
4.1 Actions on bank side

The bank wishing to offer the real-time notification service must offer the OrderType WSS or the corresponding BTF parameters. It has to be possible that the wss access data is created in the JSON format described and then provided by the EBICS server.

The bank side must be able to manage the N potential WebSocket connections for their M customers. The corresponding logon tokens must also be managed, in particular it must be possible to compare them to the assigned customer IDs. In addition, the bank creates (near-time) JSON messages when new data is available on the EBICS server or general information needs to be sent quickly to the customer.

4.2 Actions on customer side

A customer wishing to receive real-time notifications must select the order type WSS or the corresponding BTF parameters. In particular, it must be ensured that the system is able to use the returned JSON structure from the EBICS response and can establish a corresponding WebSocket connection from it. In addition, the customer system must be able to process the push messages from the WebSocket connection and must be able to derive corresponding actions (especially triggers for corresponding download requests in EBICS).