HandHeld-Device (HHD)

for the generation of an OTP

HHD enhancement for optical interfaces

Issuer:

Bundesverband deutscher Banken e.V., Berlin

Deutscher Sparkassen- und Giroverband e.V., Bonn/Berlin

Bundesverband der Deutschen Volksbanken und Raiffeisenbanken e.V., Berlin

Bundesverband Öffentlicher Banken Deutschlands e.V., Berlin

Version: V 1.5.1

Status as of: 5.06.2018

Status: Final

Reference: chipTAN ab Version 1.2 or higher

This specification was developed on behalf of the Deutsche Kreditwirtschaft (German Banking Industry Committee – GBIC). It is hereby released for implementation in customer and bank systems.

Contents

Table of figures

References

[HHD 1.3] Schnittstellenspezifikation für die ZKA-Chipkarte - HandHeldDevice (HHD) zur TAN-Erzeugung, Version 1.3, 26.10.2007, Final Version, Zentraler Kreditausschuss

[HHD 1.3.2] Schnittstellenspezifikation für die ZKA-Chipkarte - HandHeldDevice (HHD) zur TAN-Erzeugung, Version 1.3.2 Final Version, 02.02.2009, Zentraler Kreditausschuss

[HHD_UC 1.0.1] HHD-Erweiterung für unidirektionale Kopplung, Version 1.0.1 Final Version, 02.02.2009, Zentraler Kreditausschuss

[HHD 1.4] Schnittstellenspezifikation für die ZKA-Chipkarte - HandHeldDevice (HHD) zur TAN-Erzeugung, Version 1.4 Final Version, 07.05.2010, Zentraler Kreditausschuss

[Belegung 1.4.1] ZKA-TAN-Generator – Belegungsrichtlinien für die Dynamisierung der TAN, Version 1.4.1 Final Draft, 04.10.2013, Die Deutsche Kreditwirtschaft

[Belegung 1.5] Belegungsrichtlinien für das chipTAN-Verfahren, Version 1.5, 1.Draft, Stand 16.02.2018, Die Deutsche Kreditwirtschaft

[S3G-Ctn] Specifications of the Secoder 3G, Secoder Application chipTAN, Version 1.2, 21.09.2016 (or higher)

[ISO18004] ISO/IEC 18004:2000: Information Technology – Automatic Identification and data capture techniques – Bar code symbology – QR code

A. INTRODUCTION

This specification describes the transmission of data in the form of a unidirectional optical coupling (hereinafter referred to as HHD_{UC} for HHD – unidirectionally coupled). The device control itself as well as the type of visualisation is not part of this document, but of the respective specification of a HandHeldDevice (HHD)[1] (see [S3G-Ctn]).

The HandHeld-Device can be used via a HHD_{UC} application interface specified in [Belegung 1.4.1] for FinTS customer products or the specific Internet banking applications of credit institutions.

Devices with a display and keyboard that meet the specifications of the respective HHD specification serve as the hardware basis for the HandHeldDevice or Secoder. With HHD_{UC}, regardless of the connection type, the data is only transmitted in one direction, namely from the access device to the HHD/Secoder and is then confirmed by the operator there. A concrete example for a unidirectional coupling is the optical transmission by means of an animated graphic.

The specification is divided into three sections:

- General specifications which are valid independently of the transmission protocol.
- Particular specifications for the use of an optical coupling method using animated graphics.
- Particular specifications when using methods based on "photographing a matrix code".

The aim of this standardisation attempt is to create, on the basis of as few variants as possible, a method which may ensure that every Internet banking application and every FinTS customer system can be used with every HHD_{UC} available on the market and that manufacturer-specific characteristics can be avoided.

The specifications are supplementary to the Secoder or HHD specification. All other mechanisms and features not explicitly mentioned are maintained as described in the corresponding specifications.

[1] The term HandHeldDevice (HHD) is still used in this version of the document due to historical reasons, but in this version it is used for a Secoder 3 reader with the Secoder application chipTAN (ctn). The specifications for the Secoder 3 form the basis for the corresponding reader devices.

B. GENERAL SPECIFICATIONS FOR HHD_{UC}

The use of devices with unidirectional coupling and the associated characteristics are described in the respective HHD standard. This specification, however, describes the (unidirectional) transmission protocol. This version of the HHD_{UC} specification describes the transmission using the concrete example of HHD V1.4, as it was also adopted in the specifications for Secoder 3 (see [S3G-Ctn]). The specifications for HHD V1.3 can be found in the HHD_{UC} specification HHDUC V1.0.1 and are no longer subject of this consideration. However, the description of HHD_{UC} V1.4 is strictly backward compatible with HHD_{UC} V1.01 with regard to the HHDUC devices, i.e. no specifications are made which are inconsistent with HHD_{UC} V1.0.1 or HHD V1.3. Furthermore, the protocol mechanisms of HHD_{UC} V1.0.1 are used as unmodified as possible.

The processes for manual and coupled operation differ in a number of basic properties.

More specifically, this means that only user guidance information is displayed on the customer's device, while the actual transaction data is transmitted via a communication link to the HHD_{UC}, where it is shown to the customer on the display.

The use of a HHD_{UC} therefore requires some enhancements compared to manual operation:

- The HHD_{UC} must be able to transmit the entire Challenge data from the customer device in one or more communication steps and then individually show it to the customer via display and keyboard.
- If several communication steps are required for the transmission of the challenge, the entire challenge must be assembled internally from the individual communication steps before the actual processing of the challenge is continued.
- The customer must be informed of the operating status and the status of the transmission.
- Since on the one hand transmission components have to be operated and on the other hand there is no direct connection to the customer's device to charge a rechargeable battery, there must be ways of minimising the energy requirement for the transmission components.
- There are cancellation scenarios that may especially occur in coupled operation.

B.1 Standard procedures

The following standard procedures result from the specifications of the HHD specification and are intended to illustrate the differences between manual and coupled operation using HHD V1.4 as an example.

The codes and key assignments used refer to this HHD specification and may only be considered as examples (see section „B.2.1“).

B.1.1 HHD V1.4 Procedure for manually entering start code and data

The following procedure shows the process of entering start code and transaction data using HHD V1.4 as an example. In this specific case, the start code "104xxxx" is used to select the template "104" to confirm an individual domestic transfer:

Procedure	Display indication
Inserting the smart card	none or text
Pressing the "TAN" key	Start-Code ▌
Pressing numeric keys (max. 12), confirming with " confirm " key	Start-Code 104xxxx
Accepting with „confirm“ key	Überweisung Inland
Input of further values depending on the selected template	Konto Empf.: ▌ BLZ Empf.: ▌ Betrag ▌
Press the "confirm" key to confirm the entry and the TAN will be displayed	Überweisung TAN: 361620

Characteristic of the standard HHD procedure is that the start code and the transaction data have to be entered in separate steps, in which the customer must actively enter and confirm data each time. The output in the first line of the data entry screen depends on the first two to nine digits of the previously entered start code.

While the customer can take the start code from the screen of his device, he is required to take the transaction value(s) from his payment receipt.

As a result, a TAN is generated, which is shown on the HHD display and which the customer must enter in the corresponding field of his device.

B.1.2 HHD V1.4 Procedure for coupled operation

The communication sequence shown below illustrates the HHD_{UC} sequence extended by the transfer function using the example of an optical coupling using animated graphics.

Procedure	Display indication
Inserting the smart card	none or text
Starting the transfer functio	Übertragung aktiviert
during data transmission	Übertragung ▌▌▌▌▌▌
All data bytes transferred, check digit OK: The transmitted start code is not displayed, but is transparently included in the TAN calculation later.	Übertragung erfolgreich
Accept with "confirm" key	Überweisung Inland
Confirmation of further values depending on the selected template	Konto Empf.: 12345678 BLZ Empf.: 70020245 Betrag 22,45
Approve by pressing the "confirm" key, then the TAN is displayed	Überweisung TAN: 472733

As shown in the figure, the process is divided into the transmission phase and the confirmation phase.

Transmission phase

At the beginning of the transmission phase, the transmission unit is activated by pressing the start button of the transmission function ("F" or "TAN" button, see [S3G-Ctn]) once. During transmission, status information is shown on the HHD_{UC} display. The successful transmission phase is completed with the text "Transmission successful" shown on the display.

When using a procedure based on photographing a matrix code, the transfer phase consists of photographing the matrix code. In this case, the successful transmission

phase is also completed with the text "Transmission successful" shown on the display.

The transmission phase includes the verification of the correct transmission according to section B.2.4. If a transmission error is detected, the transmission phase is canceled and the display text "Transmission canceled" is shown. See also section B.3.1.

Confirmation phase

In the subsequent confirmation phase, the transaction data is displayed element by element. The customer's "input data" is taken from the transmitted data and the customer only needs to confirm it after verifying it with the original voucher.

The start code, which creates Freshness and controls the dialog flow, has no other business relevance and is therefore not displayed to the customer in the standard case (exceptions see [S3G-Ctn]); however, it is taken into account in the subsequent TAN calculation.

B.2 Data transfer protocol

The data transfer protocol is kept very compact so that there are no excessive transmission times even when using narrow-band transmission protocols.

As for HHD V1.3.2, the data protocol follows a fixed structure comprising start code and two data elements including the respective length fields (see [HHD_{UC} 1.0.1]). The data transfer protocol according to HHD_{UC} V1.0.1 is no longer part of this specification, but completely supported to ensure backward compatibility.

With HHD_{UC} Version 1.4, the structure of the protocol has changed to allow data structures with different structures to be transferred. Now a ControlByte follows the field for the length of the start code, which can branch to data patterns of different structure. A defined bit combination in the length field of the start code determines the validity of the ControlByte. This also ensures compatibility with HHD_{UC} V1.0.1.

From HHD V1.4, the data structures can be defined individually to a certain extent:

Start criterion	HHD_{uc} data block	AMS data block (optional)

The start criterion depends on the concrete transmission procedure. If an AMS data block exists, it directly follows the HHD_{UC} data block, which means that there must be no start criterion between the two data blocks.

The HHD_{UC} data block has the following structure:

HHDuc lenght	Start code lenght	Control Byte HHDuc	Data record structure analog Control	Check digit
– HHDuc data block –				

Content and structure of the HHD_{UC} data block is described in the document [S3G-Ctn] (section 9.3).

The optional AMS data block has the following structure:

AMS lenght	Control Byte AMS	Sequence Counter	MAC
– AMS data block –			

Content and general structure of the AMS data block is also described in the document [S3G-Ctn] (section 9.3). The exact coding of individual fields may depend on the specific transfer protocol.

B.2.1 Degrees of freedom and restrictions in data record setup from HHD V1.4

The control of different data structures via the ControlByte results in the following:

- Security medium
  When using HHD, a bank smart card with the SECCOS operating system is used. The DK EMV TAN generator (DK EMV AC application) is used to generate the TANs.
- Visualisation
  When using HHD with ControlByte 0x01, the general visualisation concept for HHD is used (see [S3G-Ctn]), alternatively a visualisation structure corresponding to the ControlByte is used.
- Included fields and their content
  The data structure can contain any fields which have to be documented in the description. It must also be defined how the fields shall be filled depending on the purpose.
- Data lenght
  The maximum physical data length of the HHD_{UC} data block is 255 bytes. The respective logical maximum value must be defined in the description of an HHD variant defined by the ControlByte.
- An ASCII character set (cf. [S3G-Ctn]), which the HHD supports, must be used; the character set can be adapted to the respective purpose.
- The HHD_{UC} application interface (see [Belegung 1.4.1]) can be used optionally. In case usage is required, a corresponding specification of the elements "Challenge" and "Challenge HHD_{UC}" is required.

The meaning of the individual components of the HHD_{UC} data block is described in the following sections.

B.2.2 HHD_{UC} header and trailer from HHD V1.4

As of version 1.4.2 of this document, the contents of this section are included in document [S3G-Ctn] (section 9.3).

B.2.3 HHD_{UC} Body for HHD V1.4 (Control = 0x01)

As of version 1.4.2 of this document, the contents of this section are included in document [S3G-Ctn] (section 9.3).

B.2.4 Checksum calculation for HHD$_{UC}$

As of version 1.4.2 of this document, the contents of this section are included in document [S3G-Ctn] (section 9.3).

B.3 Further protocol characteristics

B.3.1 Protocol specifications

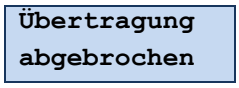
After receiving the start criterion, the HHD$_{UC}$ waits until the entire HHD$_{UC}$ data block has been received completely and without interruption. The length of the HHD$_{UC}$ data block results from its first byte (HHD$_{UC}$ length). If further data follows after the HHD$_{UC}$ data block (i.e. a start criterion does not follow directly), the AMS data block is also read in. The length of the AMS data block results from its first byte (AMS length).

After receiving the HHD$_{UC}$ data block, it is reviewed for errors using the procedure described in Section B.2.4.

If the result of the recalculation is negative, the terminal waits for the detection of the start criterion to repeat the process. The display of the transmission status also starts at 0% again after detecting the start criterion (and the directly following challenge length). If the HHD$_{UC}$ data block has to be read in again, the AMS data block is also read in again.

This process is canceled if the CheckSum is recalculated correctly or after a total of five unsuccessful attempts.

If an error occurs, the following message is displayed:



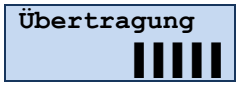
The display of the respective message can be interrupted by pressing the "Confirm" key or the "Cancel" key and the device is switched off; otherwise the device is switched off after a timeout of approx. 30 seconds.

B.3.2 Status informationen

In order to be able to quantitatively show the progress of the transmission on the display, the device must initially be able to determine the total length of the data to be transmitted. The AMS data block (if available) is not taken into account.

The data to be transferred contains both the total length of the HHD$_{UC}$ data block and the length of the individual elements. These allow the HHD$_{UC}$ to calculate the progress of the transmission in relation to the total length of the HHD$_{UC}$ data block.

With displaying the message



the customer is informed about the transfer progress.

B.3.3 Energy management

The communication module must be in permanent operating mode during operation, which results in a correspondingly high power requirement. Thus, the communication block should only be activated consciously when required, and deactivated at the earliest possible time (e.g. after successful check of the CheckByte). The operator can therefore decide at the beginning whether he wants to carry out the TAN generation by entering the context-sensitive data via the device keyboard (initiate the process by pressing the TAN key of HHD V1.4 in the standard layout) or to use the communication module for this purpose (initiate the process by pressing the "F" key of HHD V1.4 in the standard layout). The respective HHD specification describes which keys or functions are specifically used.

B.3.4 Cancelation scenarios

By pressing the "Cancel" key, the process is cancelled. It is irrelevant if the transmission has just been activated, is still active (display of the percentage value of the transmission status) or if the transmitted data is already shown on the reader's display.

When the "Cancel" button is pressed, the following is always shown:

```
Vorgang
abgebrochen
```

The display of this message can be interrupted by pressing the "Confirm" key or the "Cancel" key and the device is switched off; otherwise the device is switched off after a timeout of approx. 30 seconds.

This means that an individual correction of the data transmitted via the communication interface is not possible.

If the HHD_{UC} is removed from the reception range during data transmission, a timeout of approx. 5 seconds is started. Within this time, the HHD_{UC} can be repositioned in the reception area to resume the data transmission process. In this case the HHD_{UC} waits for the detection of the start criterion again. After detecting the start criterion (and the directly following length of the HHD_{UC} data block) the display of the transmission status starts at 0% again.

If the timeout expires without resuming the transmission, the following message will be displayed:

```
Übertragung
abgebrochen
```

The same error message is displayed for each physical transmission error, even if, for example, a length field is incorrect. If the check digit is incorrect, cancelation follows after five unsuccessful attempts (cf. Section B.3.1)

C. PARTICULAR SPECIFICATIONS FOR OPTICAL HHD_{UC}-COUPLING WITH ANIMATED GRAPHICS (HHD_{OPT})

In the following, additional specifications regarding the use of an optical coupling based on animated graphics between the customer device and HHD_{UC} - hereinafter referred to as HHD_{OPT} - are defined. In this case, the communication link consists of an optical connection between the two parties. A dynamic graphic is displayed on the screen of the customer's device in which the data to be transmitted is coded so that it can be read by the optical receiver elements in the HHD_{OPT}. HHD_{OPT} is a form of an optical coupling process. See section D for an alternative based on matrix codes. Further procedures can be added and described in a comparable way at a later date if the corresponding technologies are available.

With regard to the protocol, no additional specifications to the HHD_{UC} specifications described in section B are required. When coding the data in the AMS data block (if available), however, the specifications in section C.5 must be met.

The optical coupling of the two devices is determined by a number of framework conditions that are intended to enable the manufacturer-independent operation of a HHD_{OPT} with optical coupling.

C.1 Physical frameworks

To optically transfer the determined transaction data to HHD_{OPT}, a graphic compatible with the customer device must be generated dynamically. The following criteria are considered to be decisive:

- Dynamic processing must be fast and resource-saving; the resulting graphic must be as small as possible in terms of data volume.
- The medium used must allow the graphics to be transmitted as quickly as possible via the optical coupling path, i.e. the blink frequency must be as high as possible, with the result that the coordination between processor, graphics card, screen and presentation programme must be selected properly. The minimum and maximum blinking frequency to be supported is 2 Hz and 20 Hz.
- The generated graphic must be able to be displayed on any display, independently of . . .
    - the type of screen/display (CRT monitor, TFT, plasma, ...)
    - the size of the screen and
    - the chosen screen resolution.
- The standard HHD_{OPT} must have two markings which indicate the position of the outer optical elements. These correspond to the marks in the standardised graphic (see section C.3) and facilitate the positioning by the customer.
- Depending on the implementation, the most suitable implementation method must be chosen. For sending transaction data using an animated graphic, the following options are available:

Depiction	System requirements on customer side
Adobe® Flash®	Adobe® Flash® player must be installed
JavaScript	JavaScript must be enabled in your browser
Animated GIF	none
Sun Java®	Java® Virtual Machine must be installed

More detailed information on the characteristics of the individual implementation options can be found in section C.6.2.

The optical characteristics of the HHD_{OPT} are determined by the following parameters:

- The physical characteristics of the selected optical receiving elements (e.g. photo transistors) with regard to energy consumption, characteristic curves and protection against crosstalk and ambient light. The manufacturer shall choose a proper solution.
- It is irrelevant whether the optical receiver elements are discrete or integrated components and the decision therefore depends on the manufacturer's cost and reliability considerations.
- Derived from the above, there are minimum distances between the optical receiving elements or the device size and their quantity. The device size can be determined by the manufacturer to a certain extent (see section C.1.1 "Calibration of the animated graphic"). For the number of optical elements, a quantity of 5 has proven to be the optimum in dialogue with different manufacturers and is used as fixed value in the specification.

Since a manufacturer-independent design of the optical coupling is to be obtained, the following chapters describe the operation of an HHD_{OPT}:

Before a product can be used as HHD_{OPT}, it must be ensured that the dynamic graphics shown below can be reliably interpreted in the supported formats of the design type.

C.1.1 Calibration of the animated graphics

The objective of the presentation of the animated graphic is the proper presentation on any screen for any product without manual customer intervention. However, since the size of the devices is not fixed and not every graphic format (e.g. Animated GIF) is scaleable, the customer may have to adjust the animated graphic to the size of the HHD_{OPT} or the screen resolution.

Depending on the display format used, this calibration can be performed locally in the browser (e.g. JavaScript) or on the web server (e.g. Animated GIF) (see appendix).

Depending on the implementation, the calibration can be carried out by "dragging" the graphic on specially marked fields or by using icons such as a large and a small magnifying glass.

C.1.2 Required limitation of display duration and size

In order to avoid that the customer may suffer from epileptic symptoms, the following general conditions must be adhered to when displaying the animated graphic:

Display duration:

The display duration must not exceed 60 seconds. After this time has elapsed, the animation must be stopped automatically and the customer must be offered a suitable control element for a restart.

Size limitation:

The size of the animated graphic must be adjustable within specific limits in order to support products of different designs. If the size is increased by more than 50% of the initial size of the animated graphic, a corresponding warning is to be issued and only after the customer has confirmed this warning he can continue with further enlargement.

C.2 General HHD_{OPT} definitions

The following specifications apply to the signals used and their meaning.

Designation	Information
CLK	Data channel with clocking for data transmission; at each transition "1" → "0" data is accepted.
SYNC	SYNC pattern that serves as start-detection.
Data 0 … 3	Bit values of a half-byte to be transmitted in a data channel

Also, the following applies:

white (high) = ’1’
black (low) = ’0’

For each data byte, the least significant half-byte is transmitted first.

The values of the data fields related to the respective half-byte are as follows:

Data 0: Value = 2^0

Data 1: Value = 2^1

Data 2: Value = 2^2

Data 3: Value = 2^3

Synchronisation

A defined SYNC pattern is used to detect the beginning of a message in idle mode.

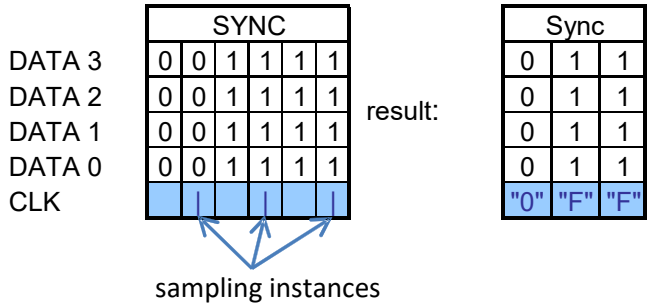


Figure 1: SYNC pattern

Since the data is only accepted at the sampling time  with a falling edge, the character sequence "0FF" results.

This pattern "0FF" must not occur in the area of the length fields, data and check digit. For the data of the HHDuc data block this is guaranteed by its construction and coding of the data elements contained. If the data to be transmitted contains an AMS data block this has to be ensured by the implementation. See section C.5 for details how this has to be handled.

During synchronisation, the CLK signal continues to operate to allow continuous clocking of the internal processes. The SYNC pattern shown in the figure above ensures that the synchronisation point can be located reliably, since within the byte sequence a defined change from "1" to "0" occurs in all data channels.

C.3 Composition of the Animated Graphic for HHD_{OPT}

For HHD_{OPT}, the optical elements used for receiving the data are arranged on one of the four sides, as the following illustration shows using the example of an integration at one of the sides:

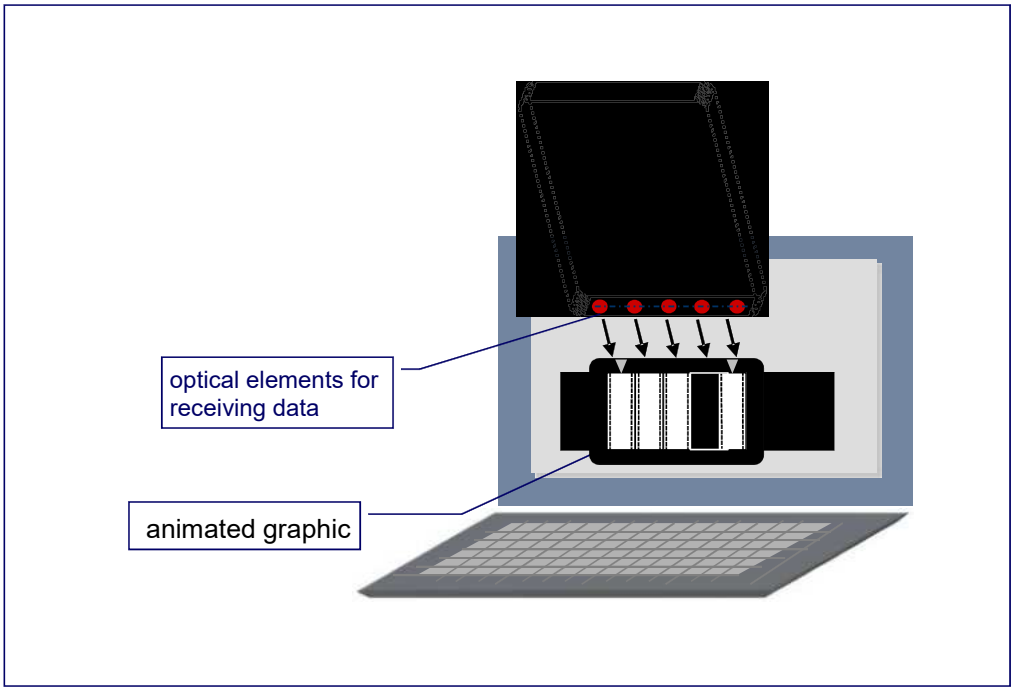


Figure 2: Example positioning of the standard HHD_{OPT} on the screen

HHD_{OPT} graphic structure

The following figure shows the structure of the animated graphic with exemplary dimensions in millimetres. The real dimensions are optimised on the basis of the screen resolutions and devices to be supported and may deviate from this example. The size of the graphic can be adapted to the physical dimensions of the specific device at the selected screen resolution using the calibration function (see section C.1.1).

The actual animated graphic is framed by a black frame in order to achieve a smoother appearance on the one hand and to achieve a defined end of the measuring range on the other hand. Two white markings are integrated in the frame to facilitate the positioning of the HHD_{OPT} (see also section C.1).

The five animated graphic components are separated by black bars to reduce crosstalk effects.

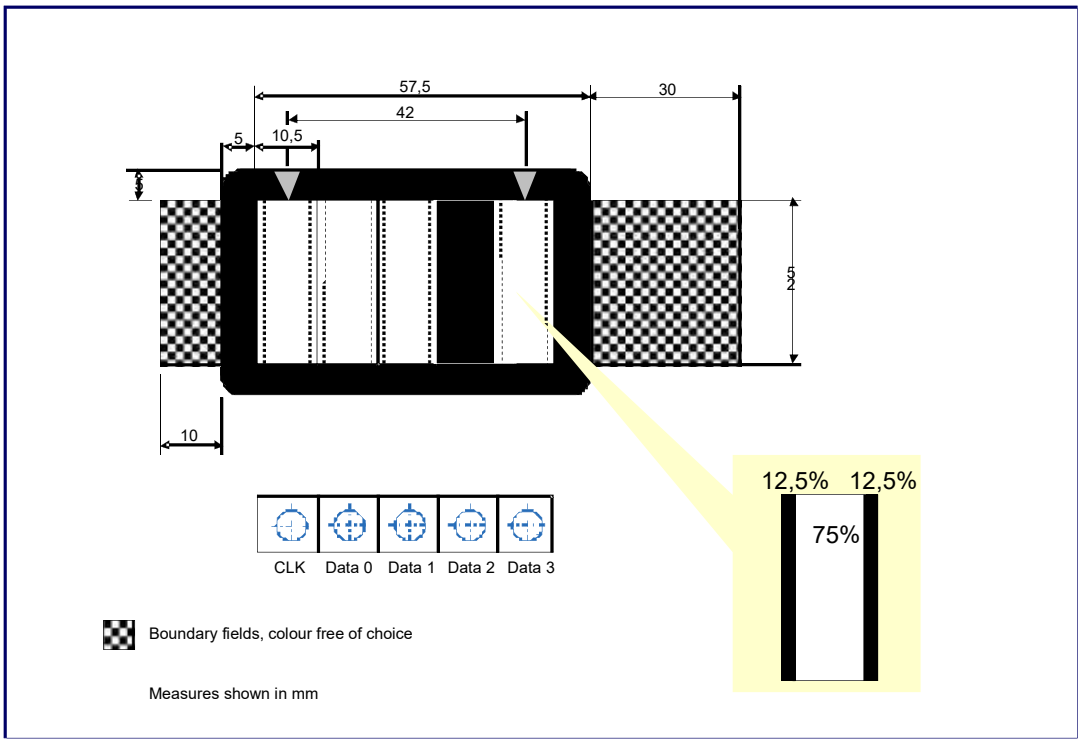


Figure 3: Structure of the animated HHD_{OPT} graphics

Procedure

The clock pulse (change between black and white) is constantly controlled by the frequency of the system (Flash®, JavaScript, Animated GIF, monitor repetition frequency, graphics card, etc.).

The sequence begins with a SYNC pattern, as described in Section C.2.

With the following rising edge of the CLK signal, the individual data bit areas are set to the desired value ("1" or "0") and scanned by the device with a certain delay (suppression of the transient response), e.g. with the falling edge of the CLK signal.

While the quantity and significance of the optical elements as well as the time sequence are the subject of this specification and must be guaranteed by the implemented animated graphic, the procedures for measurement for reading the animat-

ed graphic (e.g. measurement with rising and/or falling edge) will be specific for each manufacturer of the HHD_{OPT} device.

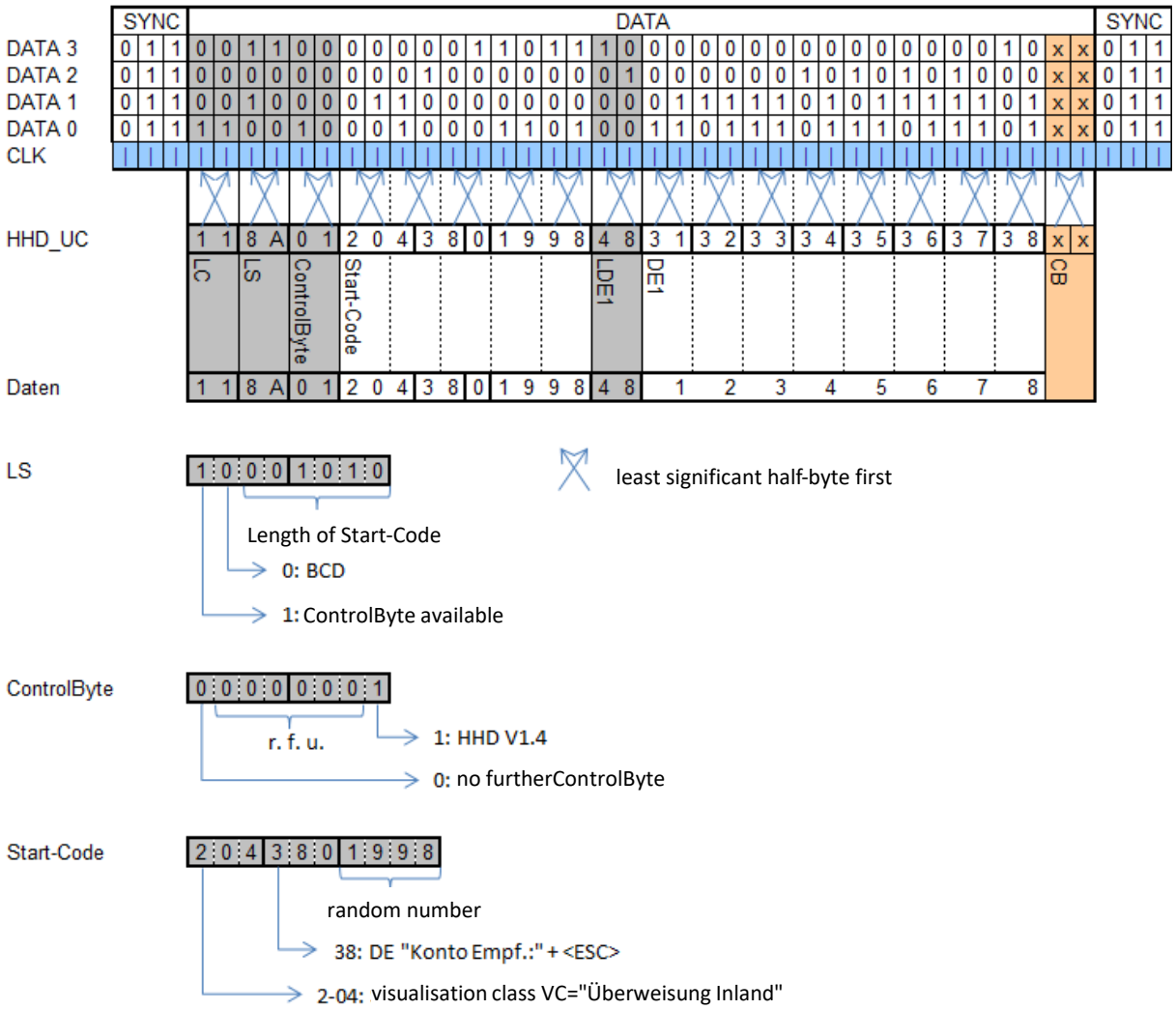


Figure 4: Time sequence example for the standard HHD_{OPT} (i.e. without AMS data)

C.4 HHDuc data block restrictions

If the control byte of the HHDuc data block (field control at position 3) has the value 0x01, then this is data for the HHD procedure according to version 1.4.

When using HHD_{OPT}, the following restrictions apply in this case:

Not more than one of the data elements contained in the data block has a maximum length of 36 characters. The maximum length of this data element in bytes is 18 bytes for BCD coding and 36 bytes for ASCII coding. The other data elements have a maximum length of 12 characters. The maximum length of these other data elements in bytes is 6 bytes for BCD coding and 12 bytes for ASCII coding.

C.5 Coding of the AMS data block

Since the pattern "0FF" is used as a SYNC pattern (start recognition), this pattern must not be part of the AMS data block. However, this pattern can theoretically occur in the sequence counter and the MAC. To avoid this while using HHD_{OPT} in the extended HHDuc data block, the fields in position 8 (sequence counter) and position 9 (MAC) must not be encoded in 8-bit binary but in 7-bit binary.

To convert the fields in position 8 and 9 to a 7-bit encoding, both must be converted together (18 bytes) as described in section 9.3.1 of [S3G-Ctn]. The result then is 21 bytes long and replaces the fields at positions 8 and 9 in the extended HHDuc data block. Accordingly, AMS_Control.b5=1 and AMS_LC=22 must be set in the extended HHDuc data block (see the description of the extended HHDuc data block in section 9.3.1 of [S3G-Ctn]).

The described conversion of the extended HHDuc data block must be carried out before creating the animated graphic.

Note: If a Secoder receives an extended HHDuc data block with AMS_Control.b5=1, it internally resets the described conversion before further processing.

C.6 Annex

C.6.1 Example for the checksum calculation

As of version 1.4.2 of this document, this example is included in the document [S3G-Ctn] (Section 9.3).

C.6.2 Characteristics of the possible graphic formats for optical coupling

The following describes the characteristics of the possible graphic formats that can be used for optical coupling.

The method used in the respective customer situation depends on the specific implementation and is decided by the respective processing software on behalf of the bank or with regard to the customer product. If necessary, a decision tree can also be used, which, for example, selects the appropriate procedure on the basis of the identified browser settings. In any case, the choice of graphic format should always be transparent to the customer and not linked to administrative activities. The same applies to customer products that either support the standards / products natively or integrate corresponding browser libraries.

C.6.2.1 Adobe® Flash®

This method allows the generation of the smallest and fastest graphics. However, the Adobe® Flash® player must be installed first. Due to the mentioned characteristics and the good performance behaviour during the dynamic generation of a one-time created class by parameterisation, this method allows an optimal implementation of the optical coupling.

However, the use of the Adobe® Flash® method requires the activation of dynamic components in the browser. If this is not desired or if the use of Flash components is not recommended by an institute, either of the two other methods must be used.

C.6.2.2 JavaScript

The use of JavaScript requires the activation of this option in the browser of the access device. In this case, JavaScript also offers sufficiently good characteristics with regard to the specified requirements.

C.6.2.3 Animated GIF

Since this method does not require any system requirements or browser settings, this graphic format can be used independently from the browser settings.

However, the downside of Animated GIF is the fact that a GIF graphic must be built completely from the web server and no dynamic functions in the browser can be used, e.g. for changing the graphic size depending on the screen resolution. In addition, the operating speed of this method is comparatively low.

C.6.2.4 Sun Java®

Sun Java® meets the required performance and speed requirements, but requires the installation of a Sun Java® virtual machine and its loading at runtime, which makes the use of this method as HHD_{OPT} only useful in Java applications.

D. PARTICULAR SPECIFICATIONS FOR THE USE OF MATRIX CODES

If a chipTAN reader has a camera, the HHDuc data block and, if available, the AMS data block can also be transferred to the chipTAN reader by photographing a corresponding matrix code instead of using an animated graphic. This procedure is called HHD_{QR}, HHD_{FM} or HHD_{MC} (depending on the specific method used to generate the matrix code). When using a matrix code as part of the chipTAN procedure, the following aspects must be considered.

D.1 Reader requirements

The chipTAN reader for the HHD_{QR}, HHD_{FM} or HHD_{MC} procedure must be implemented as a stand-alone device in accordance with the specifications for Secoder 3 and contain the Secoder application chipTAN [S3G-Ctn]. The manufacturer must confirm this by means of an appropriate manufacturer's declaration.

The chipTAN reader must never be implemented as a simulation on a smartphone or any other device.

After extracting the HHD_{UC} data block and, if applicable, the AMS data block from the matrix code, further processing of the data must take place as described in [S3G-Ctn] or Chapter B of this specification.

D.2 QR Codes[1] (HHD_{QR})

If the matrix code is generated according to the specification of a QR code (see www.qrcode.com), the procedure is referred to as HHD_{QR}. In this case, the following requirements must be met when generating the QR code.

D.2.1 General features

D.2.1.1 HHDuc data volume and QR Code size

The maximum size of a HHD_{UC} data block including AMS is 150 bytes for HHD V1.4. Four BQR bytes and one byte AMS indicator are added, see section D.2.1.3. This is the scale for the required capacity and the associated version 8 of the QR code at ECC level L. If HHD V1.3.2 without AMS is used for a minimum consideration, the maximum size is 47 bytes (+ 4 BQR bytes + 1 AMS indicator byte).

Based on these sizes, a chipTAN reader must support the following versions (source: www.qrcode.com).

Version	Module	ECC Level	Binary data	Usage data (DE = data element)
4	33 x 33	L	78	HHD V1.3.2 + AMS
		M	62	HHD V1.3.2 / max. 1 DE + AMS, max. 2 DE
5	37 x 37	L	106	HHD V1.3.2 + AMS
		M	84	HHD V1.3.2 + AMS
6	41 x 41	L	134	HHD V1.4
		M	106	HHD V1.3.2 + AMS
7	45 x 45	L	154	HHD V1.4
		M	122	HHD V1.3.2 + AMS
8	49 x 49	L	192	HHD V1.4 + AMS
		M	152	HHD V1.4

Figure 5: Data capacity and versions of QR Codes

Note: For the calculation shown in the table, the parameters ECI mode according to [ISO18004] with value 1 for ISO-8859-1 character set and 8-bit byte mode for the user data were taken into account for generating the QR code.

Based on the required capacity, a smaller version with ECC level L(ow) (7%) should be preferred to a larger version with ECC level M(edium) (15%). In incorporation of logos (see section D.2.1.5) may require a larger version.

The QR code versions 4 to 8 and the ECC level L / M can be combined as desired.

When displaying the QR code, it is essential to ensure that each module of the QR code ALWAYS uses the same amount of pixels when it is displayed, i.e. there must be no mix of modules with e.g. 3 pixels and 4 pixels within a QR code.

[1] QR-Code is a registered trademark of DENSO WAVE Inc., it is free to use, but it must be referenced. See also:

https://register.dpma.de/DPMAregister/marke/registerHABM?AKZ=013123104&CURSOR=4

This may vary for different QR codes. However, at least 3 pixels per module should be used.

D.2.1.2 Positioning aid

For the most convenient user experience, a QR code reader may include a frame in the camera window to assist the user with positioning the camera. This may help the user to center the camera on the QR code. The necessity as well as the concrete design of the positioning aid is the responsibility of the chipTAN reader manufacturer.

D.2.1.3 Identifier for Banking QR-Code (BQR)

Before starting the interpretation of the QR code, e.g. by verifying the optional AMS MAC - for this purpose an inserted smart card would have been accessed already - an identification mark for a banking QR code (abbreviation "BQR") is introduced. This identifier consists of a constant "DK" (2 bytes) at the beginning of the data stream and a 2 byte long CRC16 test value at the end. The resulting message is structured as follows:

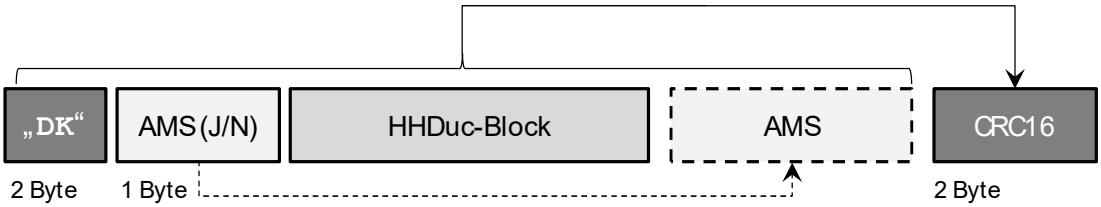


Figure 6: Structure of a banking QR-Code

The constant "DK" always marks a Banking QR code. If this constant occurs, the CRC16 test value must then be calculated and compared using the block described above. This concludes the BQR identification.

To determine the CRC check value, the following CRC-16 polynomial, starting with the least significant bit (LSB), is to be used:

$$x^{16} + x^{15} + x^2 + 1 = (x + 1)(x^{15} + x + 1)$$

The following specifications shall apply:

Initial value	0
Final XOR value	0
Data bytes inversion	N
CRC result inversion before final XOR	N

The smart card may only be accessed if the verification of BQR is successful. This avoids unnecessary access to the smart card.

The new CRC-protected AMS indicator introduced in the Banking QR code with the values "J" or "N" clearly indicates whether the data stream contains an AMS data structure or not.

Notes:

- The use of the Banking QR code does not serve to increase security, but protects a possibly inserted smart card from unnecessary accesses and increases the battery life of the reader device.
- A chipTAN reader can optionally be used to read standard QR codes (i.e. those without a BQR identifier) and show the result on the display. It is up to the manufacturer of the chipTAN reader if and in which form this function is made available.
  If the chipTAN reader cannot display a standard QR code, the following message must be shown and processing must be canceled if the verification of the BQR fails:

  Kein Banking
  QR-Code

D.2.1.4 Banking QR-Code scrambling

If the QR code is valid according to section D.2.1.3, which is the result of the BQR verification, the HHD_{UC} data must be made illegible before generating the QR code and transmitting it to the access device. Otherwise a user could discover financial data in the data stream when using a standard smartphone app, which could lead to irritation and uncertainty.

For making the data illegible, the data has to be XORed (two bytes by two bytes) with the fixed bytes “DK” (i.e. ’44 4B’).

This scrambling starts after the constant "DK" at the 3rd byte of the banking QR code and ends after the CRC test value.

Notes:

- This function does not increase security either.

D.2.1.5 Integrated image and logo support

In order to mark specific chipTAN QR codes as such, an operator can optionally integrate small images or logos.

In any case, the type and size of such images can negatively affect the reading behaviour of the QR code and the battery life. The respective institute is responsible for ensuring that the highest possible reading process quality is achieved regarding all reader products used.

D.2.2 Procedure for generating a banking QR-Code

The following graphic shows the procedure for generating a banking QR code with all mandatory and optional substeps:

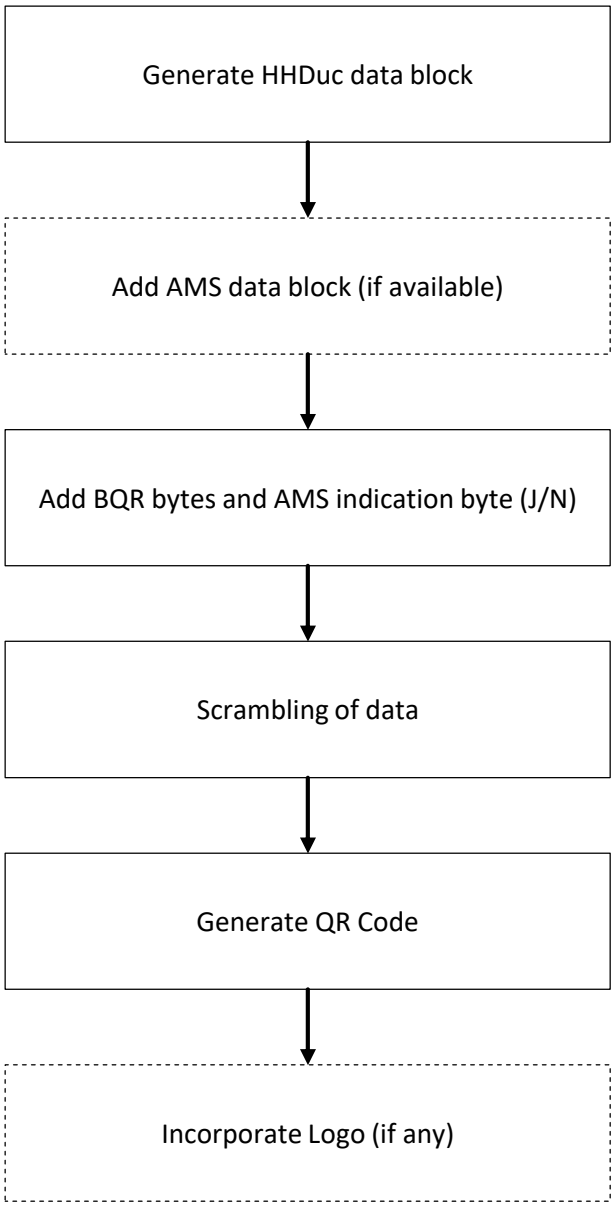


Figure 7: Procedure for generating a chipTAN QR code

The QR code generated on the server-side is transferred as a graphic (e.g. with MIME-Type=PNG) to the client and displayed there.

D.2.3 Definition of QR-Code parameters

In addition to the specifications in the previous section, the following parameters have been taken into account in the HHD_{QR} procedure:

Parameter	to be supported by a QR-Code chipTAN reader (optional functions are shown in italics)
QR Code specification	ISO/IEC 18004:2006
QR Code Model	Model 2
Type	static
Data type	binary data (8-bit byte mode)
ECI-Header	011100000001' 12 bit consisting of 4 bit identifier for ECI mode and 8 bit identifier for character set (1 = ISO-8859-1)
Error correction	ECC Level L *ECC Level M*
QR Code Version	Minimum: Version 4 - 33 x 33 Module Maximum: Version 8 - 49 x 49 Module
Colours	black / white
Quiet Zone	- 4 Module - white
Additional optional functions	*- Support of integrated images / logos* *- Scanning of non-banking QR codes*
MIME-Types[2]	PNG, GIF, JPG
Displayed image size	Minimum: 35 x 35 mm Maximum: not defined
Recording time of a QR code, beginning with the QR code being shown on the screen of the access device and ending when the calculated data is shown on the chipTAN reader display.	max. 300 ms
Camera for display delay in focusing process before QR code is read.	max. 200 ms

[2] For the use of server-based QR code generation

Parameter	to be supported by a QR-Code chipTAN reader (optional functions are shown in italics)
QR code example for version 7 with ECC level L for transmitting a maximum of 154 bytes of binary data (e.g. HHD V1.4 without AMS)	 Version 7: 45 x 45 Module ECC = L 35 x 35 mm 4 mm Quiet Zone ECI = 1

D.2.4 Example for generating a chipTAN-QR Code

The following steps show the structure of the individual data structures up to the generation of chipTAN-QR codes as described in section D.2.2.

D.2.4.1 HHDuc data block for QR-Code calculation

In this example, the following HHD_{UC} structure is used for an individual bank transfer containing the data elements "account/IBAN" (the last 10 digits) and "amount":

	Element	Content	Hexadecimal representation
HHDuc-Block	LC		1D
	LS		C8
	Control		01
	Start-Code	„821 12345“	38 32 31 31 32 33 34 35
	L(DE1)		
	DE1	„0123456789“	30 31 32 33 34 35 36 37 38 39
	L(DE2)		
	DE2	„100,00“	31 30 30 2C 30 30
	Checkbyte		02

This leads to the following data stream for the BQR determination:

1D C8 01 3832313132333435 4A 30313233343536373839 46 3130302C3030 02

D.2.4.2 BQR determination

The following data are added for the BQR determination:

	Element	Content	Hexadecimal representation
BQR-1	DK	„DK“	44 4B
	AMS-Flag	„N“	4E
HHDuc-Block			
BQR-2	CRC-16		42 35

This leads to the following result, which is then used for the scrambling:

444B 4E 1D C8 01 3832313132333435 4A 30313233343536373839 46 3130302C3030 02 4235

Note: It should be considered that the data for the formation of the CRC-16 are processed exactly as shown. Many of the routines available online first convert the data stream to ASCII and then calculate CRC-16 from this data, which leads to an incorrect result.

D.2.4.3 Banking QR-Code scrambling

For the banking QR code scrambling, the transferred data structure is manipulated via XOR using the constants "DK" (0x'44 4B'). In this case, it is started at the 3rd byte, so that the constant value "DK" is retained in plain text. The operation results in the following data structure:

444B 0A 56 8C 4A 7C79757A7678707E 0E 7B7579777F717D73737D 0D 757B7467747B 46 0971

D.2.4.4 QR-Code generation

With the resulting data string for a scrambled banking QR code, a suitable QR code can now be generated.

The following QR codes were generated using parameter version = 7 and ECC Level = L.

Part of the generation process is the application of a mask according to section 8.8 of [ISO18004]. There are 8 different masks available. The mask is selected automatically using the procedure described in [ISO18004]. Tests have shown, however, that different masks are used for specific implementations. The resulting QR code can therefore be quite different. There are no issues with the contained data and readability though, since the mask used is part of the QR code. However, the visual comparison reveals differences.

The following table shows the different QR codes for the example shown above:

Mask	**QR-Code**
0 (000)	
1 (001)	
2 (010)	
3 (011)	

Mask	**QR-Code**
4 (100)	
5 (101)	
6 (110)	
7 (111)	

D.3 Colour matrix code (HHD_{FM})

Since the beginning of 2017, the color matrix code method, based on CrontoSign Copyright © 2016 VASCO Data Security, together with the SECODER 3G chip-TAN 1.1, has been used in the Cooperative Financial Group. The Deutsche Genossenschafts-Verlag eG has taken over the further development, testing and coordination. Companies wishing to integrate and use the colour matrix code method based on SECODER 3G must consult the Deutsche Genossenschafts-Verlag or VASCO Data Security in order to establish the necessary framework conditions for its implementation.

In the following, the basic characteristics of the colour matrix code method are described.

D.3.1 Colour matrix code format and structure

This chapter describes the basic characteristics of the coloured matrix code.

The colour matrix code consists of a square image in the colours red, green, blue or white. The image is framed by a black frame with a width of two units containing the actual user data. Inside the frame there is a white zone (padding) with a width of one unit. Within this zone the actual coloured data content is visualised with 25 to 25 units. Each individual unit (colour dot) must have a minimum size of 3 pixels. The total image size is about 4 x 4 cm.

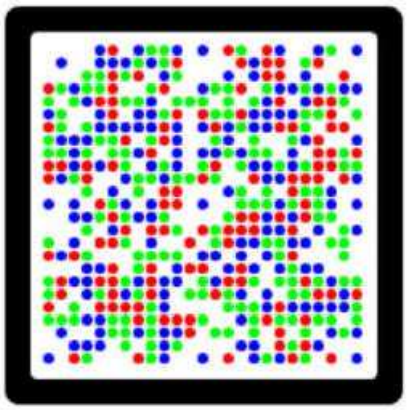


Figure 8: Colour matrix code

D.3.2 Colour matrix code dynamic content

The colour matrix code data content consists of three elements.

Payload	Error correction	Error detection

The elements are defined as follows.

Property	**Length**	**Purpose**
Payload	101 bytes	Payload with padding and obfuscation
Error correction	38 bytes	Reed-Solomon error correction.
Error detection	3 bytes	CRC-24

D.3.2.1 Payload

The payload starts with a version byte. If the HHD_{UC} block is <= 100 bytes, a single colour matrix code is generated. In case the HHD_{UC} data block is > 100 bytes, two colour matrix codes are generated. When interpreting the colour matrix code visually, sequential reading and merging of the string must be supported.

Content:

Single colour matrix code (If HHD_{UC} data block <= 100 bytes)

0x05	HHD_{UC} Data Block

Double colour matrix code (If HHD_{UC} data block > 100 bytes)

0x15	0x00	HHDuc Data Block (bytes 0 -99)
0x15	0x21	HHDuc Data Block (bytes 100 -LC)

Padding

A colour matrix code has a fixed number of bits. The bits not used are padded.

Obfuscation

To reduce the visibility of static or repeated data patterns in the colour matrix code, data scrambling is used after padding the content.

D.3.2.2 Error correction

Reed-Solomon codes are used for error correction.

D.3.2.3 Error detection

For error detection, a checksum of the payload is created and added at the end of the byte string.

Further technical details should be clarified with the above-mentioned contact persons.

D.4 Generation of further matrix codes (HHD_{MC})

If none of the methods described above is used to generate the matrix code, the method is referred to as the HHD_{MC} method.

As part of the HHD_{MC} procedure, the customer product must generate the matrix code on the access device. The output data is available as HHD_{UC} data block and in some cases as AMS data block as described in chapter B.

The matrix code generated shall be displayable on any display of an access device, independently of

- the type of screen (CRT monitor, TFT, plasma, ...)
- the size of the screen and
- the selected screen resolution.

In addition, the DK makes no further specifications for the HHD_{MC} method as to how the matrix code is to be generated. It is the responsibility of the manufacturer of the chipTAN reader and the provider of the HHD_{MC} method to ensure that all patent requirements for the use of the selected matrix code are met.

Handbook

pushTAN 2.0
(pushTAN decoupled)

Sparkassen-Finanzgruppe

finanz informatik

Contents

1 | Änderungshistorie

Version	Date	Revision
0.1	14.09.2020	Initial version
0.2	22.10.2020	Internal review
1.0	28.10.2020	Internal review and approval
1.1	23.11.2020	Update / clarification to read status
1.2	14.12.2020	Update/ clarification not supported APP versions
1.3	14.01.2021	Scheduling concretized

2 | Management Summary

Das pushTAN-Verfahren wird angepasst, sodass Endkunden bei Freigabe eines Auftrags künftig in der pushTAN-App keine TAN angezeigt bekommen, die in der Banking-Anwendung (bzw. Drittdienste-Anwendung) manuell eingegeben werden muss. Stattdessen kann der Auftrag in der pushTAN-App durch eine Schaltfläche direkt freigegeben werden.

Die neue pushTAN-Variante wird als „**pushTAN 2.0**“ bezeichnet.

Der sogenannte DECOUPLED-Approach wird nur für die TAN-Eingabe angeboten.

2.1 | Zielgruppe des Dokuments

Zielgruppe sind alle Drittdienstleister, die den **Embedded SCA Approach** verwenden. Die hier beschriebenen Änderungen betreffen nur den authentication-Type PUSH_OTP (**pushTAN**).

Für den authenticationType CHIP_OTP und SMS_OTP ergeben sich keine Änderungen. Gleiches gilt für alle Drittdienstleister die den SCA Approach REDIRECT verwenden.

2.2 | Terminplanung

Seit dem 18. November 2020 steht die Sandbox für Tests zur Verfügung. Die Echtschnittstelle wird pushTAN 2.0 ab dem 09.03.2021 für alle Institute unterstützen.

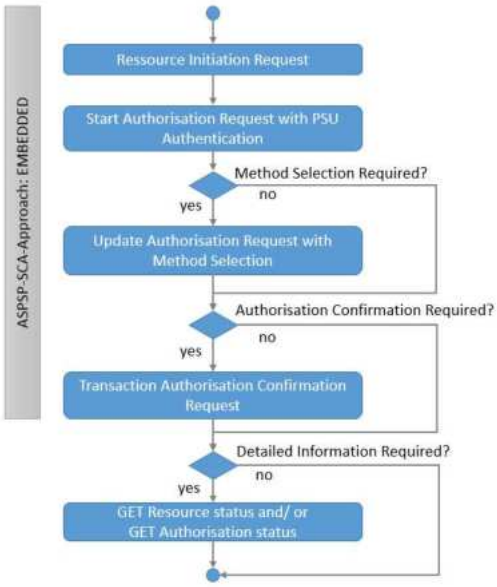
Die Aktivierung der pushTAN-Variante "pushTAN 2.0" (pushTAN decoupled) erfolgt unabhängig von der Softwareänderung in der XS2A-API durch die jeweiligen Institute. Das hat für die XS2A-Schnittstelle die Auswirkung, dass nur sehr vereinzelte Kunden mit der pushTAN-Variante 2.0 über die Schnittstelle zugreifen werden.

Der Flächen-Rollout ist für das 2. Quartal 2021 geplant. Ausführliche Informationen stellen wir Ihnen zeitgerecht im Vorfeld u. a. im Rahmen einer separaten Information zur Verfügung.

Wir bitten Sie daher im Sinne eines positiven Endbenutzer-Erlebnisses bis Ende März 2021 die Minimalanpassung vorzunehmen und empfehlen Ihnen bis Ende Juli 2021 den DECOUPLED-Approach zu unterstützen.

3 | Veränderungen am Request Flow

Bisher haben Sie eine Ressource (Payment oder Consent) initiiert, den Autorisierungs-Prozess gestartet, je nach Rückmeldung dann die Methodenauswahl vorgenommen und abschließend optional dann die TAN-Challenge beantwortet. Eine explizite Abfrage des Status einer Ressource oder einer Autorisierung erfolgte durch Sie wahrscheinlich nur im Fehlerfall.



Mit der Einführung der pushTAN-Variante„pushTAN 2.0“ (pushTAN decoupled) in der Sparkassen-Finanzgruppe wird sich dieser Prozessablauf ändern. Es wird ein DECOUPLED-Verfahren eingeführt, dass die TAN-Eingabe außerhalb Ihrer Applikation ermöglicht.

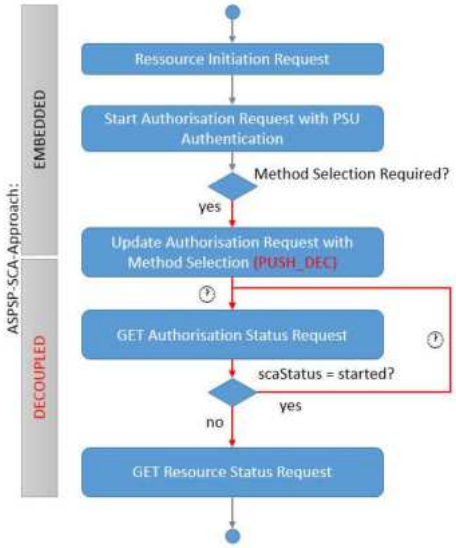
Sofern es sich um einen pushTAN-Kunden handelt, der pushTAN decoupled bereits nutzen kann (weil seine pushTAN-App in der Lage ist das Verfahren zu unterstützen sowie das Verfahren grundsätzlich nutzbar ist), werden Sie immer zur Methodenauswahl aufgefordert. Angeboten werden dann der bisher bekannte authenticationType PUSH_OTP und zusätzlich der neue authenticationType PUSH_DEC.

Wenn Sie oder der Benutzer eine authenticationMethodId vom Typ PUSH_OTP wählen, dann folgt die weitere Verarbeitung nach dem bisher bekannten Prozess. Damit können Sie flexibel den Zeitraum für die Anpassung an das neue Verfahren definieren. Nach Ende der Übergangsphase entfällt die künstlich eingeführte Methodenauswahl, wenn der Benutzer nur eine scaMethod hat.

Wenn Sie oder der Benutzer eine authenticationMethodId vom Typ PUSH_DEC wählen, dann signalisieren wir in der Response das Sie in den ASPSP-SCA-Approach DECOUPLED gewechselt sind (Response-Header: ASPSP-SCA-Approach = DECOUPLED). Die weitere Autorisierung erfolgt entkoppelt über die S-pushTAN-App. Der Transaction Authorisation Confirmation Request wird von Ihnen nicht mehr ausgelöst.

Um im neuen Verfahren identifizieren zu können, ob die Autorisierung bzw. die Ressource erfolgreich angelegt/ ausgeführt werden konnte und Sie das bankfachliche Ergebnis anzeigen können, muss der Request „GET status“ für die Autorisierung ggf. mehrfach aufgerufen werden. Wenn die Autorisierung einen positiven finalen Status hat, dann kann „GET status“ für die Ressource genutzt werden, um deren Status zu ermitteln.

Der Benutzer hat maximal 12 Minuten Zeit für die Durchführung der Autorisierung. In diesem Zeitraum kann sich der Status ändern.



In den folgenden Unterkapiteln werden die Requests noch einmal technisch detailliert beschrieben.

3.1 | Start Authorisation Request with PSU Authentication

Die Rückantwort (**scaMethods**) wird erweitert um den authentificationType PUSH_DEC. Für pushTAN Benutzer werden in der Übergangsphase immer PUSH_OTP und PUSH_DEC geliefert. Wenn Sie Ihren Prozess zu Beginn noch nicht angepasst haben, dann filtern Sie bitte den Eintrag PUSH_DEC aus und wählen Sie nur authenticationMethodIds vom Typ PUSH_OTP. Oder auch umgekehrt: Wenn Sie den neuen Prozess unterstützen, dann filtern Sie bitte authenticationMethodId PUSH_OTP aus.

	Authentication Type	Authentication Version	Authentication MethodId
	SMS_OTP	-	Bezeichnung des Benutzer-Endgerätes
ALT	PUSH_OTP	-	Bezeichnung des Benutzer-Endgerätes
NEU	PUSH_DEC	-	Bezeichnung des Benutzer-Endgerätes
	CHIP_OTP	HHD1.3.2	MANUAL
	CHIP_OTP	HHD1.3.2OPT	OPTICAL
	CHIP_OTP	HHD1.3.2QR	QR

Im nächsten Schritt führen Sie die Methodenauswahl durch.

Beispiel-Request:

```
curl -X POST \
  https://...xs2a-api/{bankcode}/v1/{resource}/{resourceID}/authorisa-
tions/{authorisationID} \
  -H 'Accept: */*' \
  -H 'Content-Type: application/json' \
  -H 'PSU-ID: Test123' \
  -H 'X-Request-ID: 97e8bd97-41ab-4761-a498-e847dec28f39' \
  -d '{
            "psuData": {
            "password": "Geheim"
  }
}
'
```

Beispiel-Response:

```
HTTP/1.1 201
status: 201
Content-Type: application/json;charset=utf-8
Location:  https://...xs2a-api/>/{bankcode}/v1/{resource}/{resourceID}/authori-
sations/{authorisationID}
ASPSP-SCA-Approach: EMBEDDED
X-Request-ID: 97e8bd97-41ab-4761-a498-e847dec28f39
{
   "scaStatus": "psuAuthenticated",
   "authorisationId": "{authorisationID}",
   "scaMethods": [
      {
         "authenticationType": "PUSH_OTP",
         "authenticationVersion": "",
         "authenticationMethodId": "Classic - Privat",
         "name": "pushTAN | Privat (******9387)"
      },
      {
         "authenticationType": "PUSH_OTP",
         "authenticationVersion": "",
         "authenticationMethodId": "Classic - Firma",
         "name": "pushTAN | BW (******7890)"
      },
      {
         "authenticationType": "PUSH_DEC",
         "authenticationVersion": "",
         "authenticationMethodId": "Privat",
         "name": "pushTAN | Privat (******9387)"
      },
      {
         "authenticationType": "PUSH_DEC",
         "authenticationVersion": "",
         "authenticationMethodId": "Firma",
```

```
        "name": "pushTAN | BW (******7890)"
      }
   ],
   "_links": {
      "scaStatus": {
         "href": "https://.../ xs2a-api/{bankcode}/v1/{resource}/{resourceID}/authorisations/{authorisationID}"
      },
      "selectAuthenticationMethod": {
         "href":   "https://.../xs2a-api/{bankcode}/v1/{resource}/{resourceID}/authorisations/{authorisationID}"
      }
   },
   "psuMessage": "Bedienungshinweis an den Endanwender."
}
```


3.2 | Update Authorisation Request with Method Selection

Der Request für die Methodenauswahl ist unverändert, aber abhängig von der getroffenen Auswahl verändert sich die Rückmeldung in der API:

- Das Header-Attribut ASPSP-SCA-Approach enthält entweder den Wert EMBEDDED oder DECOUPLED.
- Im Decoupled-Fall wird challengeData nicht ausgeliefert und es wird kein Link auf authoriseTransaction zurückgegeben, da die Freigabe komplett in der S-pushTAN-App erfolgt.

Variante 1: Es wird eine authenticationMethodId mit authenticationType": "PUSH_OTP" gewählt.


```
curl -X PUT \
  https://.../xs2a-api/{bankcode}/v1/{resource}/{resourceID}/authorisations/{authorisationID} \
  -H 'Accept: */*' \
  -H 'Content-Type: application/json' \
  -H 'X-Request-ID: 85dd4796-103d-4aa1-89fd-c5a7a32fdce9' \
  -d '{
          "authenticationMethodId": "Classic - Firma"
}
'
```


Response:


```
HTTP/1.1 200
status: 200
Content-Type: application/json; charset=utf-8
ASPSP-SCA-Approach: EMBEDDED
X-Request-ID: 85dd4796-103d-4aa1-89fd-c5a7a32fdce9
Location: https://.../xs2a-api/{bankcode}/v1/{resourceID}

{
   "scaStatus": "scaMethodSelected",
```

```
  "chosenScaMethod": {
    "authenticationType": "PUSH_OTP",
    "authenticationMethodId": "Classic - Firma",
    "name": "pushTAN | Classic - pushTAN_Med1"
  },
  "challengeData": {
    "otpMaxLength": 6,
    "otpFormat": "integer",
    "additionalInformation": "Bitte tragen Sie die TAN aus der S-pushTAN-App
ein."
  },
  "_links": {
    "authoriseTransaction": {
      "href":   "https://.../xs2a-api/{bankcode}/v1/{resource}/{resourceID}/au-
thorisations/{authorisationID}"
    },
    "scaStatus": {
      "href":   "https://.../xs2a-api/{bankcode}/v1/{resource}/{resourceID}/au-
thorisations/{authorisationID}"
    }
  },
  "psuMessage": " Bitte tragen Sie die TAN aus der S-pushTAN-App ein."
}
```


Variante 2: Es wird eine authenticationMethodId mit authenticationType": "PUSH_DEC" gewählt.

```
curl -X PUT \
 https://.../xs2a-api/{bankcode}/v1/{resource}/{resourceID}/authorisa-
tions/{authorisationID} \
 -H 'Accept: */*' \
 -H 'Content-Type: application/json' \
 -H 'X-Request-ID: 6dcf9f3f-f4d6-47f6-b7fe-1a08e57ede17' \
 -d '{
        "authenticationMethodId": "Firma"
}
'
```

Response:


```
HTTP/1.1 200
status: 200
Content-Type: application/json; charset=utf-8
ASPSP-SCA-Approach: DECOUPLED
X-Request-ID: 6dcf9f3f-f4d6-47f6-b7fe-1a08e57ede17
Location: https://.../xs2a-api/{bankcode}/v1/{resourceID}

{
  "scaStatus": "started",
  "chosenScaMethod": {
    "authenticationType": "PUSH_DEC",
    "authenticationMethodId": "Firma",
    "name": "pushDecTAN | Firma"
  },
```

```
  "_links": {
    "scaStatus": {
      "href": "https://.../xs2a-api/{bankcode}/v1/{resource}/{resourceID}/authorisations/{authorisationID}"
    }
  },
  "psuMessage": "Bitte bestätigen Sie die Transaktion mit ihrer PushTAN-APP."
}
```


Die folgende Tabelle stellt die Antwortwortvarianten nochmals gegenüber:

	Variante 1: “PUSH_OTP”	Variante 2: “PUSH_DEC”
Header: ASPSP-SCA-Approach	EMBEDDED	DECOUPLED
Body: scaStatus	scaMethodSelected	started
Body: chosenScaMethod	{ "authenticationType": "PUSH_OTP", "authenticationMethodId": "BW_Old", "name": "pushTAN | BW_old" }	{ "authenticationType": "PUSH_DEC", "authenticationMethodId": "BW", "name": "pushTAN decoupled | BW" }
Body: challengeData	{ "otpMaxLength": 6, "otpFormat": "integer", "additionalInformation": "Bitte tragen Sie die TAN aus der S-pushTAN-App ein." }	entfällt
Body: _links	authoriseTransaction scaStatus	scaStatus

3.3 | Folgerequests

3.3.1 Bei Auswahl von PUSH_DEC

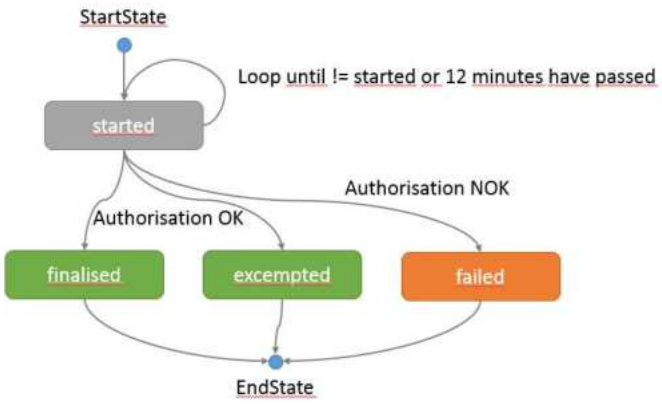
Sie rufen ab jetzt zyklisch den Endpunkt „GET authorisation“ auf, um die Information zu erhalten, ob der Auftrag durch den Kunden in der S-pushTAN-App freigegeben wurde. Nur wenn sie ein positives Ergebnis erhalten machen weitere Abfragen auf die eigentliche Ressource Sinn.

```
curl -X GET \
  https://.../xs2a-api/{bankcode}/v1/{resource}/{resourceID}/authorisations/{authorisationID} \
  -H 'Accept: application/json' \
```

```
-H 'X-Request-ID: 97e8bd97-41ab-4761-a498-e847dec28f39' \
```

Das Attribut scaStatus kann ab diesem Prozessschritt folgende Ausprägungen haben:

- started (Neu): Die Autorisierung wurde gestartet. Der Benutzer muss die Freigabe über die S-pushTAN-App vornehmen. Die Freigabe ist noch nicht erfolgt.
- finalized: Die Autorisierung wurde erfolgreich mit einem zweiten Faktor abgeschlossen.
- failed: Die Autorisierung wurde nicht erfolgreich abgeschlossen. Im Decoupled-Approach heißt das konkret: Entweder die Zeit ist abgelaufen oder der Benutzer hat nicht bestätigt.
- exempted: Die Autorisierung wurde erfolgreich ohne einen zweiten Faktor abgeschlossen.



Wenn der Status der Autorisierung OK ist (scaStatus „finalised“ oder „exempted“) wird auch die zu autorisierende Ressource aktualisiert und wechselt den Status.

Wenn der Status der Autorisierung „failed“ ist, bleibt die zu autorisierende Ressource im Status „received“ bzw. „RCVD“. In diesem Fall kann die Autorisierung nochmals gestartet werden. Weiterhin erhalten Sie durch den Aufruf von „GET Authorisation“ Informationen zum vorhandenen Fehler.

```
curl -X GET \
  https://.../xs2a-api/{bankcode}/v1/{resource}/{resoureceID}/authorisations/{authorisationID} \
```

Um den Status der Ressource abzufragen nutzen Sie bitte

```
curl -X GET \
  https://.../xs2a-api/{bankcode}/v1/{resource}/{resourceID}/status \
  -H 'X-Request-ID: 97e8bd97-41ab-4761-a498-e847dec28f39' \
```

Der Endpunkt antwortet mit einem http Status Code 200 und signalisiert den Zustand der Ressource über den Response Body:

consentStatus	transactionStatus	Erläuterung
received	RCVD (Received)	Diese Kombination kann es nur geben, wenn die Autorisierung fehlgeschlagen ist, aber trotzdem der Status der Ressource gelesen wird. Die Autorisierung kann nochmals gestartet werden beginnend mit Start Authorisation with PSU Authentication
rejected	RJCT (Rejected)	Die Ausführung/ Anlage der Ressource wurde abgelehnt aus Gründen die nichts mit der Autorisierung zu tun haben. Bsp.: Keine Deckung.
valid	ACCC (AcceptedSettlementCompleted) ACCP (AcceptedCustomerProfile) ACTC (AcceptedTechnicalValidation) PART (PartiallyAccepted)	Die Ressource wurde angelegt/ ausgeführt.
-	PATC (PartiallyAcceptedTechnical Correct)	Die Ressource wurde angelegt/ ausgeführt. Es handelt sich um ein Instant Payment das noch nicht komplett ausgeführt ist.

Der Endpunkt liefert zusätzlich eine psuMessage mit Detailinformationen aus denen der Grund (insbesondere der Ablehnungen) für den Endbenutzer ersichtlich ist: "psuMessage": "{Nr.}- {Text}."

3.3.2 Bei Auswahl von PUSH_OTP

Sie schließen wie gewohnt den Autorisierungs-Flow mit authorise Transaction ab:

```
curl -X PUT \
  https://.../xs2a-api/{bankcode}/v1/{resource}/{resourceID}/authorisations/{authorisationID} \
  -H 'Content-Type: application/json' \
  -H 'X-Request-ID: 97e8bd97-41ab-4761-a498-e847dec28f39' \
  -d '{
        "scaAuthenticationData": "123456"
}'
```

3.4 | Verhalten bei einer veralteten pushTAN-App

Sofern ein Kunde das pushTAN decoupled Verfahren prinzipiell nutzen kann, jedoch noch eine veraltete App nutzt, welche decoupled-Variante nicht unterstützt, ist mit dem folgenden Verhalten zu rechnen:

Wenn Sie oder der Benutzer eine authenticationMethodId vom Typ PUSH_DEC wählen, kann die Autorisierung nicht erfolgreich beendet werden. Wenn Sie nun den Endpunkt „GET authorisation“ aufrufen, erhalten Sie die folgende Rückmeldung:

```
{
  "scaStatus": "failed",
  "psuMessage": "3015- Abrufversuch durch inkompatiblen Client"
}
```

In diesem Fall müssen Sie die Autorisierung der Ressource erneut starten und die authenticationMethodId vom Typ PUSH_OTP wählen, dann folgt die weitere Verarbeitung nach dem bisher bekannten Prozess.

4 | Support

4.1 | Sandbox

In der von uns bereitgestellten Sandbox wurde das pushTAN decoupled Verfahren ebenfalls simuliert. Das Verhalten ist an eine PSU-ID gekoppelt. Die PSU-ID lautet: „pushDecTAN“.

Attribut	**Wert**
PSU-ID	pushDecTAN
password	okok1
scaAuthenticationData	111111

4.2 | Bei Rückfragen

Wenn Sie weitere Fragen haben wenden Sie sich bitte an unseren Support: https://xs2a.sparkassen-hub.com/home

4.3 | Anpassung der Dokumentation

Die bereitgestellte YAML im carepackage auf der XS2A Welcome-Page ist im Zuge der Anpassungen für das pushTAN decoupled Verfahren überarbeitet worden.

Handbook

Account Transactions Query

Sparkassen-Finanzgruppe

finanz informatik

Contents

1 | Revision History

Version	Date	Revision
0.1	10.07.2019	Initial version
1.0	10.07.2019	Internal review and approval

2 | Management Summary

2.1 | Purpose of this document

This document explains the function of the Read Transaction List endpoint, as this endpoint can be controlled by various options.

Call:
GET /v1/accounts/{account-id}/transactions {query-parameters}

For the full description, please refer to Chapter 6.6.4 of 03. NextGenPSD2 Access to Account Interoperability Framework - Implementation Guidelines V1.3.4_20190705.pdf and Chapter 6BG of XS2A Extended IG Additional Information 20190923.docx.

In the following chapters, the behavior of the endpoint as determined by

- query parameters
- header attributes and
- initial/subsequent calls

is described.

This document focuses on optional or conditional attributes. Mandatory attributes of the Berlin Group specifications must be supplied.

2.2 | Functional scope of the endpoint

Depending on the selected query, the endpoint will deliver booked and/or earmarked transactions in the camt.52 format or standing orders that are currently on hand in the json format.

2.3 | Organization of the document

Chapter 3 describes the query parameters, and chapter 4 describes the header attributes. In Chapter 5 we explain the possible as well as practical combinations of query parameters and header attributes. Finally, in chapter 6, we explain how this is implemented in the sandbox, as it does not mimic the behavior of the actual interface.

3 | Query Parameters

This chapter briefly describes all the essential information for the query parameters.

3.1 | Supported query parameters

dateFrom
During the initial retrieval of account transactions, a period of more than 90 days may be specified. Subsequent queries only allow a maximum of 90 days. The initial retrieval is understood to be the first call after the consent has been set up.

If no value is supplied, the date is automatically set to 90 days prior to the current date.

The institutions maintained by FI have the option of limiting the transaction history for the individual online access channels. This limitation works in all access channels, which is to say that it also works in the XS2A API. If your query period exceeds the configured time period, then you will receive the transactions for the maximum possible number of days.

dateTo
The value must be greater than or equal to the dateFrom. If the value is not supplied, then the current date will automatically be used.

bookingStatus
We support all characteristics mentioned in the Berlin Group specification:

- booked: delivers only booked transactions
- pending: delivers only earmarked transactions
- both: delivers booked and earmarked transactions
- information: delivers the standing orders currently on hand

deltaList
We always remember when account transactions were last retrieved and therefore offer the opportunity to make a delta call. You can switch from dedicated requests to the delta delivery at any time. If deltaList = true, then dateFrom is automatically set to the date of the last retrieval of booked transactions, and dateTo is set to the current date. Incidental values of both parameters are ignored.

3.2 | Unsupported query parameters

entryReferenceFrom
withBalance

4 | Header Attributes

This chapter briefly describes all the essential information for the header attributes.

4.1 | Supported header attributes

PSU-IP-Address
If the attribute is not supplied, then the request is interpreted as an automatic request. These are limited in the consent on a daily basis (frequencyPerDay).

Authorization
If the consent was created using the redirect SCA approach, a valid OAuth access token is required.

Accept
We accept if Accept is not sent. If it is sent, then only the following characteristics are allowed:

- */*
- application/xml
- application/json

4.2 | Unsupported header attributes

None.

5 | Possible Combinations

Not all possible combinations make technical sense, which is why we ignore some attribute characteristics. This behavior is documented here.

bookingStatus=pending
For requests with "bookingStatus=pending", we ignore dateFrom, dateTo and deltaList when these are supplied. Earmarked transactions technically always refer to the current booking day. Delivery is always in the camt.52 format (i.e., the Accept header must be application/xml). Queries with application/json are rejected.

bookingStatus=information
For requests with "bookingStatus=information", we ignore dateFrom, dateTo and deltaList if they are supplied. Standing-order inventories technically always refer to the current booking day. Delivery is always in the json format (i.e., the Accept header must be application/json). Queries with application/xml are rejected.

bookingStatus=both oder booked
The following table does not contain all possible combinations but does try to explain the basic characteristics.

	Query Parameters				behavior in the interface		
initial request	booking-Status	dateFrom	dateTo	deltaList	dateFrom	dateTo	comment
not relevant	booked or both	without	without	false	= Today - 90 days	= Today	If no date range is supplied, then the system automatically sets -90 days to today.
yes	booked or both	greater than 90 days	*not relevant*	*not relevant*	= chosen date		The initial call is not limited to 90 days.
no	booked or both	greater than 90 days	*not relevant*	*not relevant*	= Today - 90 days		Follow-up calls are limited to a maximum of 90 days backwards.
no	booked or both	date	date	true	= date last request	= Today	dateFrom and dateTo are ignored. If start and end dates are to be taken into account, then deltaList = true must not be supplied.
not relevant	both	date	date < Today	*not relevant*	= chosen date	= chosen date	The request will be answered, but you will not receive any pending transactions since pending transactions can only exist for the current day.

6 | Deviations from the Sandbox

The sandbox simulates the actual interface. The functionality of the sandbox will be expanded starting in November 2019. Up to this point, a fixed camt.52 statement would be delivered statically. In the future, the bookingStatus will also be supported in the sandbox. In contrast to the actual interface, there are the following simplifications:

- There is no distinction between initial and follow-up calls. All queries are firmly limited to 90 days prior to the present day.
- The starting balance is always 0.00 EUR. For each calendar day requested, an entry for 1 EUR will be listed in the "booked" status and the final balance calculated dynamically.
- The standing-order inventories and the earmarked transactions are static.

Finanz Informatik

XS2A-API

Implementer Options

Supported Features

Version	Date	Changelog
1.0	20.02.2019	First Version
1.1	07.03.2019	Smaller Changes, Layout
1.2	24.04.2019	Notice on delivery of multiple camt.052-files in read transaction
1.5	11.09.2019	Deletion of notice for read transaction (fixed)
2.0	29.10.2019	IO24: Change from NO to YES Changes for "BG XS2A Extended IG Additional Information" in - IO30 - IO31 - IO33
2.1	11.11.2019	IO34: Change balancetype openingBooked052 to openingBooked (typing error)
3.0	07.02.2020	Added: Support for instant bulk-payments in JSON and PAIN-message format Added: Hints to IO23
3.1	04.06.2020	Added: Changes with Major Release 20.1 - added: balanceType interimAvailable
3.2	15.01.2021	Added: pushTAN DECOUPLED (pushTAN 2.0)

ID	Description	Characteristic	Decision
IO1	Mandate the TPP to sign requests on application level	-	NO
IO2	Supported Single Payment products	sepa-credit-transfers	YES
		instant-sepa-credit-transfers	YES
		target-2-payments	YES
		cross-border-credit-transfers	YES
		pain.001-sepa-credit-transfers	YES
		pain.001-instant-sepa-credit-transfers	YES
		pain.001-target-2-payments	YES
		pain.001-cross-border-credit-transfers	NO
IO3	Supported Bulk Payment products	sepa-credit-transfers	YES
		instant-sepa-credit-transfers	YES
		target-2-payments	NO
		cross-border-credit-transfers	NO
		pain.001-sepa-credit-transfers	YES
		pain.001-instant-sepa-credit-transfers	YES
		pain.001-target-2-payments	NO
		pain.001-cross-border-credit-transfers	NO
IO4	Supported Periodic Payment products	sepa-credit-transfers	YES
		pain.001-sepa-credit-transfers	YES
IO5	(on principle) Supported SCA Approaches	Redirect	NO
		OAuth2	YES
		Decoupled	NO
		Embedded	YES
		Mixed Embedded/ Decoupled	YES
IO6	OAuth2 required as a pre-step for PSU authentication	-	NO
IO8	Support of TPP Messages on operational issues	-	YES
IO9	Risk management regarding the offering of SCA methods via the XS2A-Interface	-	PSU needs to be authenticated (either with a first relevant factor or an access token)
IO10	Transaction fees transported via the XS2A-Interface	-	NO
IO11	Supported SCA Methods	-	c.f. Table 'SCA method & approach (options)'
IO12	Configuration of supported SCA methods – applicable SCA Approaches	-	c.f. Table 'SCA method & approach (options)'
IO13	Configuration of supported SCA methods – TPP Redirect Preferred	-	c.f. Table 'SCA method & approach (options)'
IO14	*Authentication Requirements for the Decoupled SCA Approach*	-	*Not applicable (e.g. Decoupled not supported)*
IO15	PSU-ID required in message	Payment Initiation Request	NO
		AccountInformationConsentRequest	NO
		Payment Cancellation	NO
		Signing Basket	NO
IO16	PSU-ID-Type required in message	Payment Initiation Request	NO
		AccountInformationConsentRequest	NO
		Payment Cancellation	NO
		Signing Basket	NO
IO17	Support of multicurrency accounts	-	NO
IO18	Representation of an account	**Accounts are represented by the IBAN** (currently the only supported represantation considered for tests)	YES
IO19	PSU-Corporate-ID required in message, if a corporate account is affected	Payment Initiation Request	NO
		AccountInformationConsentRequest	NO
		Payment Cancellation	NO
		Signing Basket	NO
IO20	PSU-Corporate-ID-Type required in message, if a corporate account is affected	Payment Initiation Request	NO
		AccountInformationConsentRequest	NO
		Payment Cancellation	NO
		Signing Basket	NO
IO21	Support of future dated payments	-	YES
IO22	Support of SCA exemption	if creditor account belongs to PSU	YES
		if creditor is on a whitelist of the PSU	YES
		if instructed amount does not exceed a certain limit	YES
		Transaction Risk Analyse	YES
IO23	Support of sessions (combination of AIS and PIS)	Combined Services (combinedService Indicator)	NO
IO24	Support of PSU messages in relevant scenarios	Decoupled SCA initiated	YES
		SCA method chosen (Embedded)	YES
IO25	Grouping restrictions for Signing Baskets	Singning baskets for the same Payment product allowed (only individual payments)	NO

ID	Description	Characteristic	Decision
		Singning baskets for the various Payment products allowed (only individual payments)	NO
		Singning baskets for the same Payment product allowed (also payments with multi level SCA)	NO
		Singning baskets for the various Payment products allowed allowed (also payments with multi level SCA)	NO
		Singning baskets for Payments and Consent Establishment allowed (only individual payments)	NO
		Singning baskets for Payments and Consent Establishment allowed allowed (also payments with multi level SCA)	NO
IO26	SCA required for Payment Cancellation	-	YES
IO27	Multi level SCA supported for Use Cases	Payment Initiation	YES
		Consent Establishment	NO
		Signing Baskets	NO
		Payment Cancellation	YES
IO28	SCA approach supported for multi level SCA	Redirect	YES
		Embedded	YES
		Decoupled	YES
IO29	ASPSP enforces explicit start of authorisation	Redirect	Never
		Embedded	Always
		Decoupled	Always
IO30	Support of optional account Information access rights	all PSD2 related services for all accounts	NO
		only access rights in request, accounts handled between PSU and ASPSP afterwards	YES
		Consent Extension BG XS2A Extended IG "additional Account Information" necessary to get ownerName or standing orders? see EN_OwnerName	NO
		list of available accounts	YES
		list of available accounts with balances	YES
IO31	Support of formats for account information	XML: camt.052	YES
		XML: camt.053	NO
		XML: camt.054	NO
		JSON (transactions)	NO
		JSON (standing orders)	YES
		Text: MT942	NO
		Text: MT940	NO
IO32	Support of optional Endpoints for AIS	accounts?withBalance	YES
		accounts/{account-id}?withBalance	YES
		accounts/{account-id}/transactions?withBalance	YES
		accounts/{account-id}/transactions/{resourceId}	NO
IO33	Support of optional (values of) query parameters for AIS	entryReferenceFrom	NO
		bookingStatus=pending	YES
		bookingStatus=both	YES
		bookingStatus=information	YES
		deltaList	YES
IO34	Support of Balance Types	openingBooked~~052~~	NO
		expected	NO
		interimAvailable	YES
		forwardAvailable	NO
		nonInvoiced	NO
		closingBooked	YES
IO35	Conditions for delivery of a transaction list directly in the response		Always
IO36	Conditions for delivery of a transaction list as a separate download with only a link in the response		Never
IO37	Redirect after first SCA-Factor	Currently the test concept assumes that all ASPSPs will not demand "Redirect" as the SCA-Approach, when they already de-manded (and received) a first SCA-factor via the XS2A-Interface. (Please confirm this assumption)	YES
IO38	Implicit start of transaction authorisation supported	-	YES
IO39	API steering links of type “startAuthorisationWith…” supported (i.e. creation of authorisation sub-resources and delivery of missing data at the same time supported)		YES

ID	Description	Characteristic	Decision
IO40	PSU Authentication data delivered via the XS2A-Interface (Embedded Approach) shall be encrypted at application level		NO
IO41	*Access to Multicurrency Account Details*		*Not Applicable*
IO42	Card Number supported to identify subaccounts		NO
IO43	Support of payment Cancellation per payment product	Pre Authentication - Supported Single Payment products o sepa-credit-transfers o instant-sepa-credit-transfers o target-2-payments o cross-border-credit-transfers o pain.001-sepa-credit-transfers o pain.001-instant-sepa-credit-transfers o pain.001-target-2-payments - Supported Bulk Payment products o sepa-credit-transfers o pain.001-sepa-credit-transfers - Supported Periodic Payment products o sepa-credit-transfers o pain.001-sepa-credit-transfers	YES
IO43	Support of payment Cancellation per payment product	After Authentication - Supported Single Payment products o pain.001-sepa-credit-transfers - Supported Bulk Payment products o pain.001-sepa-credit-transfers - Supported Periodic Payment products o sepa-credit-transfers o pain.001-sepa-credit-transfers	YES
		pain.001-cross-border-credit-transfers	NO
IO44	Supported Formats of payment status response bodies for XML-based payments	XML	NO
		JSON	YES
		BOTH	NO
IO45	Processing of regular (not instant) Payments	Batch Booking	YES
		Realtime Booking	YES
IO46	Permission of Requests for Account Data Reading with PSU involvement and reference to a recurring consent.	YES, no restrictions (ASPSP accepts requests for account information with PSU involvement that refer to a recurring consent)	YES
		NO (ASPSP denies requests for account information with PSU involvement)	NO
		WITH RESTRICTIONS (Not completely defined in [XS2A-BP], therefore not completely analysed here)	NO
IO47	Counting the frequency of AIS requests	Each request (on a specific end-point) is counted.	YES
		Frequency is counted in an accumu-lated way (e.g. all requests within a given time frame are counted as one with regards to the frequency)	NO

SCA Method Configuration - Default						
SCA Method	Applicable SCA Approaches			TPP Redirect Preferred		
	Embedded	Decoupled	Redirect	Yes	No	Unused
smsTAN	YES	NO	YES	Redirect	Embedded	Redirect
pushTAN	YES	NO	YES	Redirect	Embedded	Redirect
chipTAN manuell	YES	NO	YES	Redirect	Embedded	Redirect
chipTAN optisch (animated)	YES	NO	YES	Redirect	Embedded	Redirect
chipTAN QR	YES	NO	YES	Redirect	Embedded	Redirect
chipTAN USB (not supported)	NO	NO	NO	Redirect	Embedded	Redirect
pushTAN 2.0 (mixed EMBEDDED/ DECOUPLED)* authenticationType": "PUSH_DEC"	NO	YES	NO	Redirect	Embedded	Redirect
SCA Method 7 (to be defined by the Implementer)	*NO*	*NO*	*NO*	*Redirect*	*Redirect*	*Redirect*
SCA Method 8 (to be defined by the Implementer)	*NO*	*NO*	*NO*	*Redirect*	*Redirect*	*Redirect*
SCA Method 9 (to be defined by the Implementer)	*NO*	*NO*	*NO*	*Redirect*	*Redirect*	*Redirect*
SCA Method 10 (to be defined by the Implementer)	*NO*	*NO*	*NO*	*Redirect*	*Redirect*	*Redirect*

* see DE_PushTAN decoupled_v1.3_20210114.pdf or EN_PushTAN decoupled_v1.3_20210114.pdf for more details.

YES/NO Decisions	SCA Risk Management	SCA Approaches	First Factor in Decoupled Approach
YES	PSU needs to be authenticated (either with a first relevant factor or an access token)	Embedded	One Factor Authentication via the XS2A-Interface required
NO	SCA methods may be offered before the PSU has been authenticated	Decoupled	Identification sufficient
	SCA methods are offered without any preconditions	*Redirect*	*Not applicable (e.g. Decoupled not supported)*

Handbook

Account Details: ownerName

Sparkassen-Finanzgruppe

finanz informatik

Contents

1 | Revision History

Version	Date	Revision
0.1	10.16.2019	Initial version
1.0	10.16.2019	Internal review and approval

2 | Management Summary

The Berlin Group will soon publish the BG XS2A Extended IG Additional Information specification. Based on this specification, we will supply the account holder name in the "Account Details" structure.

3 | Affected Endpoints

Currently the "Account Details" structure is supplied in the following endpoints:

- GET /v1/accounts {query-parameters}
- GET /v1/accounts/{account-id} {query-parameters}

4 | Structural Adjustments

The structure described in Chapter 14.18 Account Details in 03. NextGenPSD2 Access to Account Interoperability Framework - Implementation Guidelines V1.3.4 is changing as follows:
A new attribute called ownerName will be supplied. The new attribute is added after "currency" and before "name".

...

currency	*Currency Code*	*Mandatory*	*Account currency*
ownerName	Max70Text	Optional	Name of the legal account owner. If there is more than one owner, then multiple names might be noted here. For a corporate account, the corporate name is used for this attribute. Even if supported by the ASPSP, the provision of this field might depend on the fact whether the PSU coincides with the account owner and/or whether an explicit consent to this specific additional account information has been given by the PSU.
name	*Max35Text*	*Optional*	*Name of the account given by the bank or the PSU in online banking*

...

5 | Consent Permissions

Additional permissions on the consent are not required to retrieve the account owner name. The information is always supplied.

Handbook

pushTAN 2.0
(pushTAN decoupled)

Sparkassen-Finanzgruppe

finanz informatik

Contents

1 | Changelog

Version	Date	Revision
0.1	14.09.2020	Initial version
0.2	22.10.2020	Internal review
1.0	28.10.2020	Internal review and approval
1.1	23.11.2020	Update / clarification to read status
1.2	14.12.2020	Update/ clarification not supported APP versions
1.3	14.01.2021	Scheduling concretized

2 | Management Summary

The pushTAN procedure is being adapted so that when approving an order, end customers will no longer be shown a TAN in the pushTAN app that has to be entered manually in the banking application (or third-party application). Instead, the order can be approved directly in the pushTAN app with a button.

This new pushTAN variant is called "**pushTAN 2.0**".

This so-called DECOUPLED approach is only offered for TAN entries.

2.1 | Target Audience for this Document

The target audience for this document is all third-party service providers who make use of the **embedded SCA approach**. The changes described here only affect the authenticationType PUSH_OTP (**pushTAN**).

There are no changes to the authenticationType CHIP_OTP and SMS_OTP. The same applies to all third-party service providers who use the REDIRECT SCA approach.

2.2 | Schedule

The sandbox has been available for tests since November 18, 2020. The real interface will support pushTAN 2.0 for all institutes from March 9th, 2021.

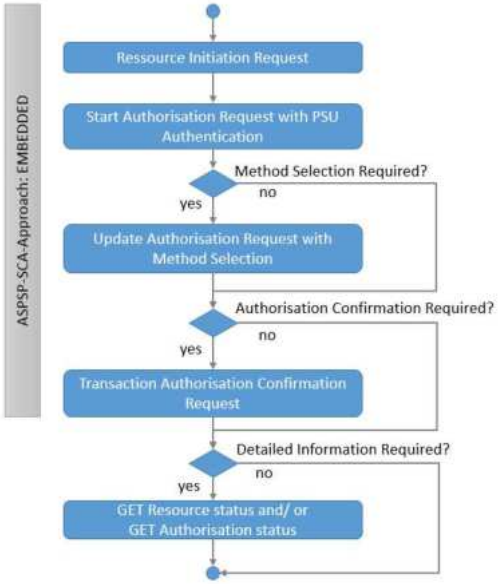
The activation of the pushTAN variant "pushTAN 2.0" (pushTAN decoupled) takes place independently of the software change in the XS2A-API by each individual institute. For the XS2A interface, this has the effect that only a small amount of customers will access the interface with the pushTAN variant 2.0.

The area rollout is planned for the second quarter of 2021.

We therefore ask you to make the minimum adjustment by the end of March 2021 in the interests of a positive end-user experience. We recommend that you support the DECOUPLED approach by the end of July 2021.

3 | Changes to the Request Flow

Up to now you have initiated a resource (payment or consent), begun the authorization process, then selected the method based on the feedback and, finally, optionally answered the TAN challenge. It is likely that you have only requested the status of a resource or an authorization explicitly in the event of an error.



With the introduction of the pushTAN variant known as “pushTAN 2.0” (pushTAN decoupled) among the Sparkassen-Finanzgruppe, this process flow will change. A DECOUPLED procedure will be introduced that enables the TAN to be entered outside of your application.

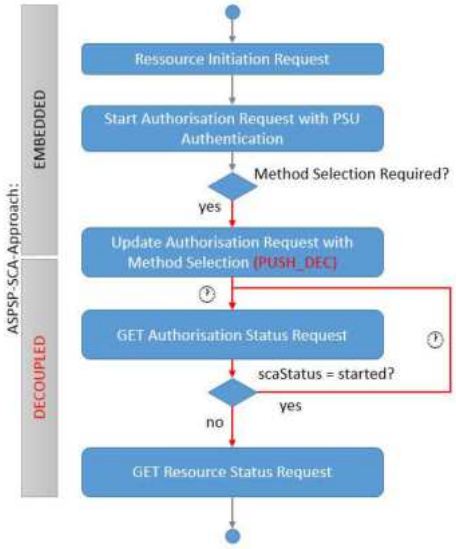
If a pushTAN customer is already able to make use of pushTAN decoupled (because their pushTAN app supports the process and the process can be used in theory), you will always be asked to select a method. The already familiar authenticationType PUSH_OTP as well as the new authenticationType PUSH_DEC are then offered.

If you or the user select the PUSH_OTP type of authenticationMethodId, subsequent processing continues to follow the familiar process. This allows you to flexibly set the period of time for adapting to the new procedure. After the transition phase ends, the artificially introduced method selection is no longer required if the user only has one scaMethod.

If you or the user select the PUSH_DEC type of authenticationMethodId, then we indicate in the response that you have switched to the DECOUPLED ASPSP-SCA approach (response header: AS-PSP-SCA-Approach = DECOUPLED). Subsequent authorization is decoupled via the S-pushTAN app. You will no longer trigger the transaction authorization confirmation request.

As part of this new procedure, to be able to identify whether the authorization or the resource could be successfully created/executed and you can display the proper banking result, the "GET status" request must in certain cases be invoked multiple times for the authorization. If the authorization has a positive final status, then "GET status" can be used for the resource to determine its status.

The user has a maximum of 12 minutes to carry out the authorization. The status can change during this period.



In the following subsections, the requests are described again in technical detail.

3.1 | Start Authorisation Request with PSU Authentication

The response (**scaMethods**) is being expanded to include the authenticationType PUSH_DEC. For pushTAN users, PUSH_OTP and PUSH_DEC will always be delivered during the transition phase. If you have not yet adapted your process at the start, then please filter out the PUSH_DEC entry and select only the PUSH_OTP type of authenticationMethodIds. Or vice versa: If you support the new process, please filter out authenticationMethodId PUSH_OTP.

	Authentication Type	Authentication Version	Authentication MethodId
	SMS_OTP	-	Description of the end user device
OLD	PUSH_OTP	-	Description of the end user device
NEW	PUSH_DEC	-	Description of the end user device
	CHIP_OTP	HHD1.3.2	MANUAL
	CHIP_OTP	HHD1.3.2OPT	OPTICAL
	CHIP_OTP	HHD1.3.2QR	QR

You will carry out the method selection in the next step.

Example request:

```
curl -X POST \
  https://...xs2a-
api/{bankcode}/v1/{resource}/{resourceID}/authorisations/{authorisationID} \
  -H 'Accept: */*' \
  -H 'Content-Type: application/json' \
  -H 'PSU-ID: Test123' \
  -H 'X-Request-ID: 97e8bd97-41ab-4761-a498-e847dec28f39' \
  -d '{
```

```
          "psuData": {
          "password": "Geheim"
 }
}
'
```

Example response:

```
HTTP/1.1 201
status: 201
Content-Type: application/json;charset=utf-8
Location: https://...xs2a-api/>/{bankcode}/v1/{resource>/{resourceID}/authorisations/{authorisationID}
ASPSP-SCA-Approach: EMBEDDED
X-Request-ID: 97e8bd97-41ab-4761-a498-e847dec28f39
{
  "scaStatus": "psuAuthenticated",
  "authorisationId": "{authorisationID}",
  "scaMethods": [
    {
      "authenticationType": "PUSH_OTP",
      "authenticationVersion": "",
      "authenticationMethodId": "Classic - Privat",
      "name": "pushTAN | Privat (******9387)"
    },
    {
      "authenticationType": "PUSH_OTP",
      "authenticationVersion": "",
      "authenticationMethodId": "Classic - Firma",
      "name": "pushTAN | BW (******7890)"
    },
    {
      "authenticationType": "PUSH_DEC",
      "authenticationVersion": "",
      "authenticationMethodId": "Privat",
      "name": "pushTAN | Privat (******9387)"
    },
    {
      "authenticationType": "PUSH_DEC",
      "authenticationVersion": "",
      "authenticationMethodId": "Firma",
      "name": "pushTAN | BW (******7890)"
    }
  ],
  "_links": {
    "scaStatus": {
      "href": "https://.../ xs2a-api/{bankcode}/v1/{resource}/{resourceID}/authorisations/{authorisationID}"
    },
    "selectAuthenticationMethod": {
      "href": "https://.../xs2a-api/{bankcode}/v1/{resource}/{resourceID}/authorisations/{authorisationID}"
```

```
    }
  },
  "psuMessage": "Bedienungshinweis an den Endanwender."
}
```

3.2 | Update Authorisation Request with Method Selection

The request for selecting the method remains unchanged, but the response in the API changes depending on the selection that has been made:

- The ASPSP-SCA-Approach header attribute contains either the value EMBEDDED or DECOUPLED.
- In the decoupled case, challengeData is not delivered and no link to authoriseTransaction is returned, since the approval takes place entirely within the S-pushTAN app.

Variant 1: An authenticationMethodId with "authenticationType": "PUSH_OTP" is selected.

```
curl -X PUT \
 https://.../xs2a-
api/{bankcode}/v1/{resource}/{resourceID}/authorisations/{authorisationID} \
 -H 'Accept: */*' \
 -H 'Content-Type: application/json' \
 -H 'X-Request-ID: 85dd4796-103d-4aa1-89fd-c5a7a32fdce9' \
 -d '{
        "authenticationMethodId": "Classic - Firma"
}
'
```

Response:

```
HTTP/1.1 200
status: 200
Content-Type: application/json; charset=utf-8
ASPSP-SCA-Approach: EMBEDDED
X-Request-ID: 85dd4796-103d-4aa1-89fd-c5a7a32fdce9
Location: https://.../xs2a-api/{bankcode}/v1/{resourceID}

{
  "scaStatus": "scaMethodSelected",
  "chosenScaMethod": {
    "authenticationType": "PUSH_OTP",
    "authenticationMethodId": "Classic - Firma",
    "name": "pushTAN | Classic - pushTAN_Med1"
  },
  "challengeData": {
    "otpMaxLength": 6,
    "otpFormat": "integer",
    "additionalInformation": "Bitte tragen Sie die TAN aus der S-pushTAN-App
ein."
  },
  "_links": {
```

```
      "authoriseTransaction": {
        "href": "https://.../xs2a-api/{bankcode}/v1/{resource}/{resourceID}/authorisations/{authorisationID}"
      },
      "scaStatus": {
        "href": "https://.../xs2a-api/{bankcode}/v1/{resource}/{resourceID}/authorisations/{authorisationID}"
      }
    },
    "psuMessage": " Bitte tragen Sie die TAN aus der S-pushTAN-App ein."
}
```


Variant 2: An authenticationMethodId with "authenticationType": "PUSH_DEC" is selected.


```
curl -X PUT \
  https://.../xs2a-api/{bankcode}/v1/{resource}/{resourceID}/authorisations/{authorisationID} \
  -H 'Accept: */*' \
  -H 'Content-Type: application/json' \
  -H 'X-Request-ID: 6dcf9f3f-f4d6-47f6-b7fe-1a08e57ede17' \
  -d '{
          "authenticationMethodId": "Firma"
}
'
```


Response:


```
HTTP/1.1 200
status: 200
Content-Type: application/json; charset=utf-8
ASPSP-SCA-Approach: DECOUPLED
X-Request-ID: 6dcf9f3f-f4d6-47f6-b7fe-1a08e57ede17
Location: https://.../xs2a-api/{bankcode}/v1/{resourceID}

{
    "scaStatus": "started",
    "chosenScaMethod": {
      "authenticationType": "PUSH_DEC",
      "authenticationMethodId": "Firma",
      "name": "pushDecTAN | Firma"
    },
    "_links": {
      "scaStatus": {
        "href": "https://.../xs2a-api/{bankcode}/v1/{resource}/{resourceID}/authorisations/{authorisationID}"
      }
    },
    "psuMessage": "Bitte bestätigen Sie die Transaktion mit ihrer PushTAN-APP."
}
```

The following table compares the response variants once again:

	Variant 1: "PUSH_OTP"	Variant 2: "PUSH_DEC"
Header: ASPSP-SCA-Approach	EMBEDDED	DECOUPLED
Body: scaStatus	scaMethodSelected	started
Body: chosenScaMethod	{ "authenticationType": "PUSH_OTP", "authenticationMethodId": "BW_Old", "name": "pushTAN | BW_old" }	{ "authenticationType": "PUSH_DEC", "authenticationMethodId": "BW", "name": "pushTAN decoupled | BW" }
Body: challengeData	{ "otpMaxLength": 6, "otpFormat": "integer", "additionalInformation": "Bitte tragen Sie die TAN aus der S-pushTAN-App ein." }	n/a
Body: _links	authoriseTransaction scaStatus	scaStatus

3.3 | Follow-up Requests

3.3.1 When PUSH_DEC Is Selected

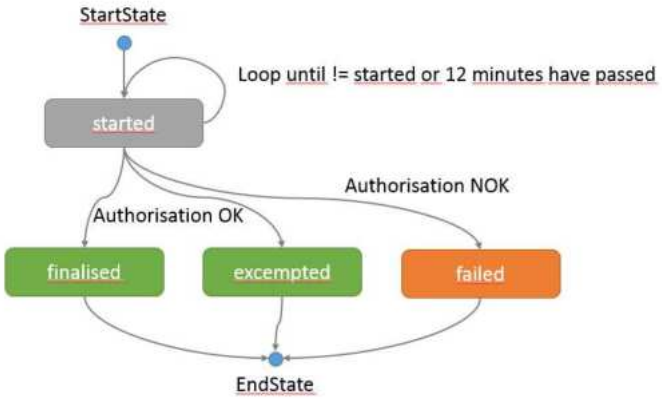
Going forward, you cyclically call the "GET authorization" endpoint to receive information about whether the order has been approved by the customer in the S-pushTAN app. Further requests on the actual resource only make sense if you receive a positive result.

```
curl -X GET \
  https://.../xs2a-
api/{bankcode}/v1/{resource}/{resourceID}/authorisations/{authorisationID} \
  -H 'Accept: application/json' \
  -H 'X-Request-ID: 97e8bd97-41ab-4761-a498-e847dec28f39' \
```

Beginning with this process step, the scaStatus attribute can have the following characteristics:

- started (New): The authorization has begun. The user has to perform the approval via the S-pushTAN app. Approval has not yet taken place.
- finalized: The authorization was successfully completed with a second factor.
- failed: The authorization was not completed successfully. In the decoupled approach, this means that either the time has expired or the user has not confirmed.

- exempted: The authorization was successfully completed without a second factor.



If the status of the authorization is OK (scaStatus "finalised" or "exempted"), the resource to be authorized is also updated and its status changes.

If the status of the authorization is failed, the resource to be authorized remains in the “received” or “RVCD” status. In this case, the authorization can be started again. You get further informations about the error by calling “GET Authorization”.

```
curl -X GET \
  https://.../xs2a-
api/{bankcode}/v1/{resource}/{resoureceID}/authorisations/{authorisationID} \
```

To query the status of the resource, please use:

```
curl -X GET \
  https://.../xs2a-api/{bankcode}/v1/{resource}/{resourceID}/status \
  -H 'X-Request-ID: 97e8bd97-41ab-4761-a498-e847dec28f39' \
```

The endpoint responds with an http status code 200 and signals the status of the resource via the response body:

consentStatus	transactionStatus	Comment
received	RCVD (Received)	This combination can only exist if the authorization has failed, but the status of the resource is still being read. The authorization can be started again beginning with Start Authorization with PSU Authentication
rejected	RJCT (Rejected)	Executing/creating the resource was rejected for reasons that have nothing to do with the

consentStatus	transactionStatus	Comment
		authorization (e.g., insufficient funds).
valid	ACCC (AcceptedSettlementCompleted) ACCP (AcceptedCustomerProfile) ACTC (AcceptedTechnicalValidation) PART (PartiallyAccepted)	The resource was created/executed.
-	PATC (PartiallyAcceptedTechnical Correct)	The resource was created/executed. It involves an instant payment that has not yet been fully executed.

The endpoint also provides a psuMessage with detailed information for the end user: "psuMessage": "{No.} - {Text}." Reasons for rejection can be found here.

3.3.2 When PUSH_OTP Is Selected

As usual, you conclude the authorization flow with authoriseTransaction:

```
curl -X PUT \
  https://.../xs2a-
api/{bankcode}/v1/{resource}/{resourceID}/authorisations/{authorisationID} \
  -H 'Content-Type: application/json' \
  -H 'X-Request-ID: 97e8bd97-41ab-4761-a498-e847dec28f39' \
  -d '{
          "scaAuthenticationData": "123456"
 }'
```

3.4 | Behavior by an outdated pushTAN-App

If a costumer can use pushTAN decoupled in principle, but is still using an outdated app that does not support pushTAN decpoupled, the folloeing bevavior is to be expected:

If you or the customer select an authenticationMethodId of the type PUSH_DEC, the authorization can’t be completed successfully. If you call “GET authorization” you get the following response:

```
{
   "scaStatus": "failed",
   "psuMessage": "3015- Abrufversuch durch inkompatiblen Client"
}
```

In this case, you have to restart the authorization with an authenticationMethodId of the type PUSH_OTP. Further processing is carried out according to the known procedure.

4 | Support

4.1 | Sandbox

The pushTAN decoupled process is also simulated in the sandbox we have provided. The behavior is linked to a PSU ID. The PSU ID is called "pushDecTAN".

Attribute	Value
PSU-ID	pushDecTAN
password	okok1
scaAuthenticationData	111111

4.2 | Questions?

Should you have any further questions, please contact our support team:
https://xs2a.sparkassen-hub.com/home

4.3 Adaptation of the documentation

The provided YAML in the carepackage on the XS2A Welcome-Page has been revised in the course of providing pushTAN decoupled.

finanz informatik

Welcome to the Sparkassen PSD2 Sandbox

Version 1.2

September 6, 2019

Table of Contents

Information contained in the downloaded documents

This document is designed to assist you in setting up the XS2A interface. The other files in the download contain documents that explain the use of the interface and provide the parameters required to call the interface.

Also included are interface descriptions in the OpenAPI/Swagger format as well as basic REST client setup with Postman in example.

EN_ReadMe_vx.x_YYMMDD.pdf

This document contains a list of all parameters available in the Sparkassen sandbox with a description of their implementation. The parameters that are described in this document are also used as environment variables in the Postman collections for greater ease of use.

EN_Implementer Options_vx.x_YYMMDD.pdf

The Implementer Options illustrate the scope of the sandbox.

EN_bankcodelist_ vx.x_YYMMDD.pdf

This file lists all supported bankcodes for the production environment.

Additional Documents

If there are additional Documents contained in the carepackage, their use is described in a Management Summary at the beginning of the document.

OpenAPI or Swagger Files: /swagger/*.json

Swagger (https://swagger.io) or OpenAPI (https://swagger.io/docs/specification/about/) is a description language for specifying REST interfaces. The description contains all the information required for the call, including descriptions, and thereby provides a starting point for human and machine to interact with the specified REST services.

For humans, there are Swagger editors, which generate a superficial interface from the description that enables the interface to be called and also aids usability through the descriptions.

During development, Swagger code generators can be used to generate code for clients or servers from the specifications. In this way development is accelerated and the quality of the software is increased through the consistent quality of the generated code.

The download contains the Swagger documents for the Berlin Group (https://www.berlingroup.org/nextgenpsd2-downloads).

finanz informatik

Setup for sandbox access

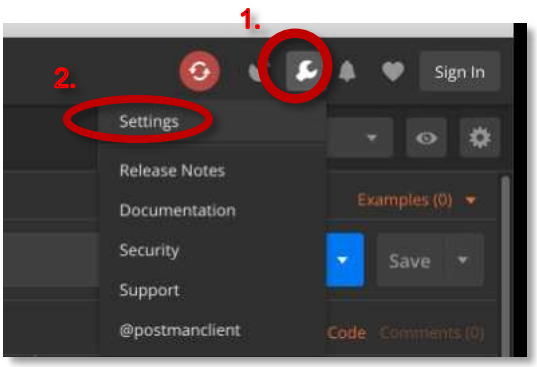
The following paragraphs describe how to setup client certificates to access our XS2A sandbox. Postman client is used for example, any other REST client like curl or Insomnia need to be setup in a similar way.

Setting up client certificates in Postman

One requirement for using the XS2A interface is the use of the client certificate. How to set that up in Postman is described below.

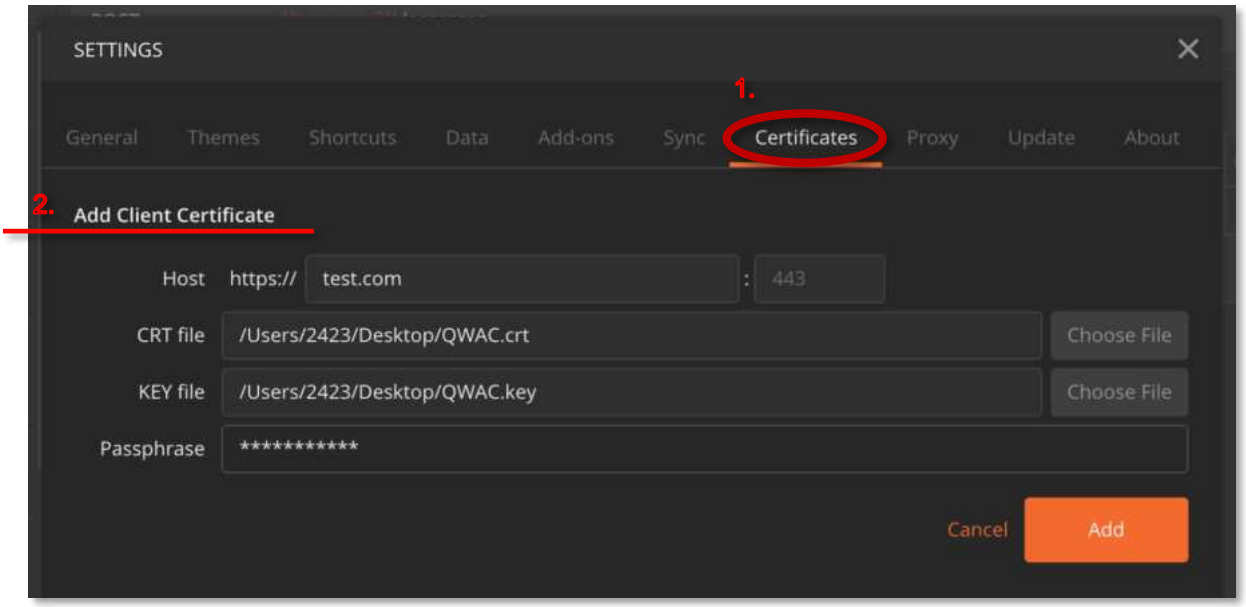
If the client certificate is in the PEM format, the same certificate and key file can be used in Postman and in CURL.

The following explanation is based on:
https://learning.getpostman.com/docs/postman/sending_api_requests/certificates/



Client certificates must be saved in the settings. First click on the *wrench* icon and then select *Settings*.

Attention:
If you are executing Redirect Collection “Automatically follow redirects” has to be switched OFF.



Next, enter the data for the certificate under the Certificates tab:

1. Enter the host for which the certificate should be imported and, if necessary, the port number as well.
2. This is the path to the certificate file.
3. This is the path to the key file.
4. Provide the password, if one exists, for the client certificate.

Apply the settings by clicking *Add*.

There is no need to adjust further settings. When Postman sends a request to the host you have entered, the certificate is sent automatically.

In the dialog box shown above, you can also remove the certificate. After saving a certificate, a "Remove" link appears for this purpose.

To check whether the certificate has been sent, the console can be opened with the keyboard shortcut **`CMD / CTRL + ALT + C`**. After submitting a request, which certificates have been selected and sent will be listed under *Client Certificate*.

Determining institution-specific URLs as well as available versions

To determine the correct URL for the selected version, use the following request:

Parameter	Value	Description
URL	`https://xs2a-sandbox.f-iapim.de:8444/fixs2a-env/xs2aapi/v1/bc/<BANKTLEITZAHL>/versions`	Endpoint for retrieving the version. A valid routing number must be transferred.
HTTP Method	`GET`	The request is made via HTTP GET.

A call with CURL therefore looks like this:

```
curl -X GET https://xs2a-sandbox.f-i-apim.de:8444/fixs2a-env/xs2a-api
/<BANKTLEITZAHL>/versions \
  --key '/Users/youruser/Desktop/QWAC-KEY.pem' \
  --cert '/Users/youruser/Desktop/QWAC.pem'
```

A successful response delivers an HTTP/200 "OK" status code and the following message body:

```
{
    "versions": [
        {
            "name": "v1",
            "href": "https://xs2a-sandbox.f-i-apim.de:8444/fixs2aenv/xs2a-
api/<BANKLEITZAHL>/v1",
            "spec": "1.3.0",
            "revision": "1"
        }
    ] }
```

It contains detailed information about the available versions and an institution-specific URL. From this point forward, all available role-based features can be used via this URL.

Handbook

XS2A Redirect

Sparkassen-Finanzgruppe

finanz informatik

Contents

1 Revision History

Version	Date	Revision
1.0	08.28.2019	Initial publication
1.1	10.02.2019	Chapter 4: Correction attribute name clientID and reference to space after scope added. Chapter 7: Note on different spelling clientID/ client-id added.
1.2	29.10.2019	Chapter 4: Correction Scope-Coding
1.2.1	15.11.2019	Chapter 4:
1.2.2	13.10.2020	Chapter 4: Correction attribute names for Redirect-Parameter
1.3	02.06.2021	Chapter 4: Add attribute restrictions
1.4	12.01.2024	Correction in Chapter 5 and 7

2 Intro

With the September 2019 release, we support authorization using the redirect approach in combination with OAuth2. The bank-offered consent is also supported in this context whereby the user/PSU determines the authorization of the consent via our online banking system.

Subsequent access to authorized resources using the redirect approach must then be performed with a mandatory OAuth access token.

The authorization using the redirect approach takes place via the Internet presence (online banking) of the Sparkassen/financial institutions.

This documentation describes the use of our redirect solution. Basic information is provided in the Berlin Group Specification and RFC 6749/8414 and will not be described redundantly here.

3 Creating a Resource

When you create a resource (consent, payment, payment cancellation), the TPP-Redirect-Preferred header attribute determines whether to authorize using the redirect or the embedded approach. If you want to authorize the resource using the embedded approach, you must specify this explicitly (TPP-Redirect-Preferred=false). If the attribute is missing, we automatically assume that the authorization should take place using the redirect approach.

In the redirect approach the TPP-Redirect-URI header attribute is a mandatory field. However, we also recommend that you use TPP-Nok-Redirect-URI so that you can distinguish between success and error events in the sequence. To be able to assign the received authorization code to your process during a redirect, please use the "state" variable (see the following section as well). Because of this, it is not necessary to include custom parameters in one of the redirect URIs; OAuth2 framework parameters are used, which are specially designed for this application. You can nevertheless optionally specify additional parameters in the redirect URL. These are transmitted unchanged during a redirect.

You can change the approach when canceling payments. In other words, a payment authorized using the embedded approach can be canceled using the redirect approach and vice versa.

If you have created the resource, you will obtain a "scaOAuth2" link instead of the "startAuthorisationWithPsuAuthentication" link.

4 Redirecting the PSU

Important. Please note:
If you create Consents or Payments using the redirect mode the link "scaOAuth2" instead of "startAuthorisationWithPsuAuthentication" is returned.

This link is the link to the OAuth Well Known endpoint.

For the first step, you have to determine the authorization server's metadata using the OAuth-Well-Known endpoint (see "scaOAuth2" link). This endpoint provides RFC-8414-compliant URLs for the authorization endpoint (Sparkassen Internet presence/online banking) and for the endpoint in order to create tokens.

Example:

```
{
  "issuer": "https://xxx",
  "authorization_endpoint": "https://yyy ",
  "token_endpoint": "https://zzz"
}
```

In the second step, you must complete the OAuth parameters for the resource and redirect the user there.

Parameter	Description	Required/Optional
response_type	Indicates what kind of response the client expects. It is possible to request tokens directly. In this implementation, however, only authorization codes (=code) can be requested.	Required Fix: code
client_id	Your client ID of the TPP according to the Berlin Group Specification protocol. Specifically, this concerns the subjectNcaId from your certificate.	Required Max: 128
scope	Scope for which access is being requested (i.e., for a consent or payment). Allocated according to the Berlin Group Specification as follows: "PIS“ + Colon („%3A“) + <PaymentID>"	Required Max: 40

	or "AIS:“ + Colon („%3A“) + <ConsentID>"	
state	Variable returned during a redirect that allows you to assign the authorization code you received to your original process.	Required Max: 255
code_challenge	Challenge per PKCE (Proof Key for Code Exchange RFC 7636)	Required Max: 128
code_challenge_method	Method per PKCE (RFC 7636)	Required Fix: S256

The following attributes are limited in the character set:

- client_id
- state
- code_challenge

Allowed: ("^[a-zA-Z0-9_-]+.*$")

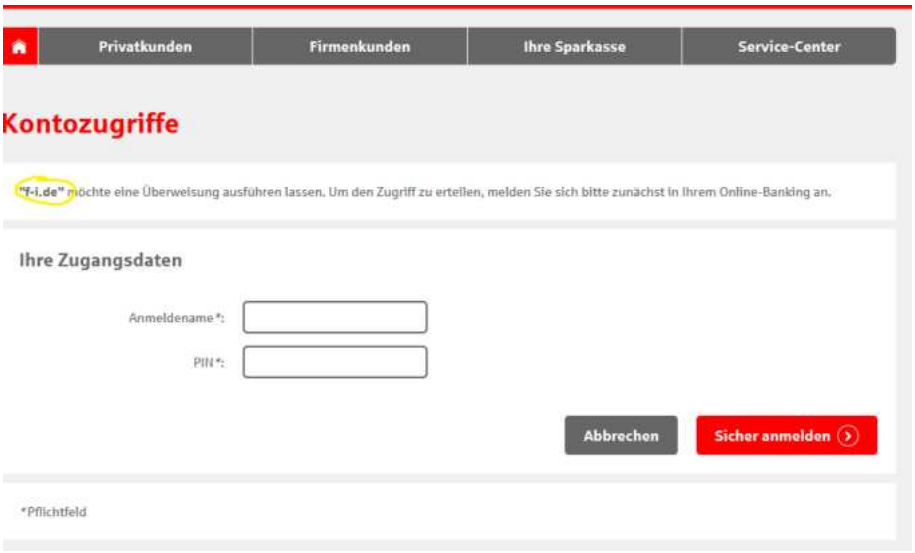
Illustrative example URL for redirecting a user:

```
https://www.sparkasse-xxx.de/.../xs2a/authorize?
        response_type=code&
        client_id=<TPP Registration number>&
        scope=AIS%3A<consentId>&
        state=S8NJ7uqk5fY4EjNvP&
        code_challenge_method=S256&
        code_challenge=vXVXiMA4CQ_Buik94dCNpfIfveWdNxMEwVtxGDz7xWg
```

5 Sparkassen Internet presence

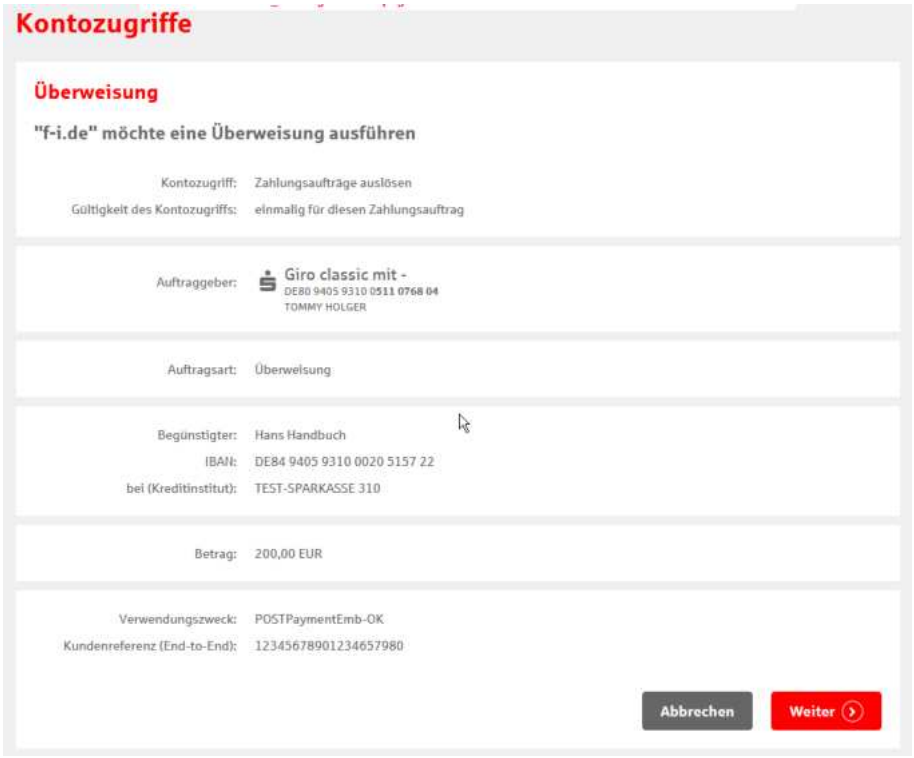
Even before logging in, the user receives information about which service provider would like to begin an authorization.

Example with a TPP named f-i.de:



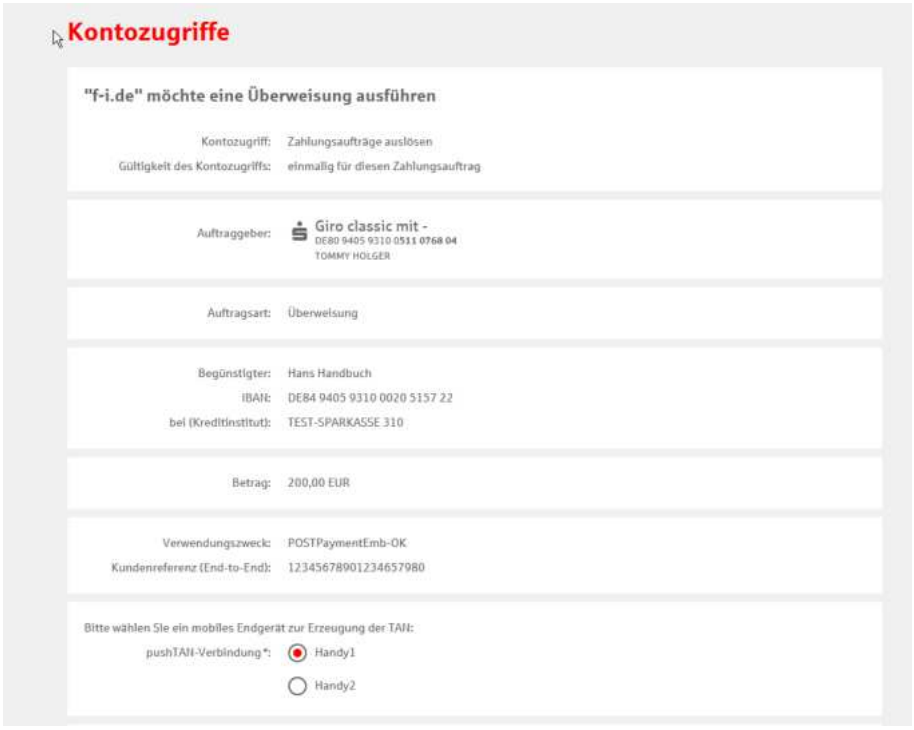
The user then logs on and receives detailed information about the resource (consent, payment, payment cancellation).

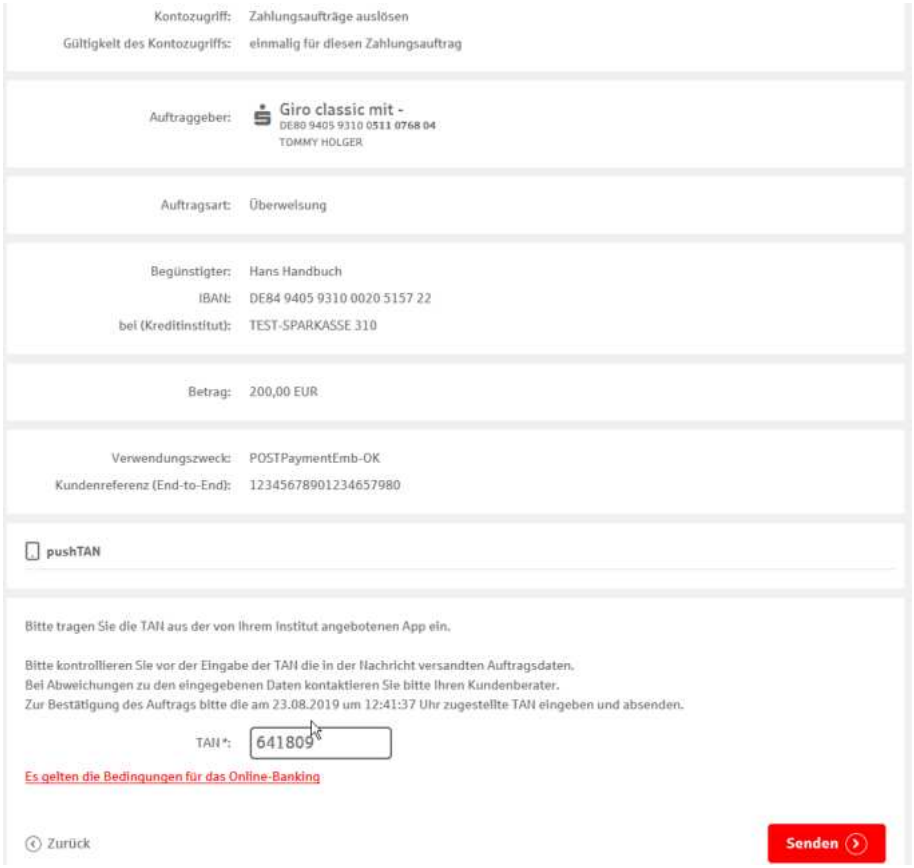
Example with a scheduled transfer:



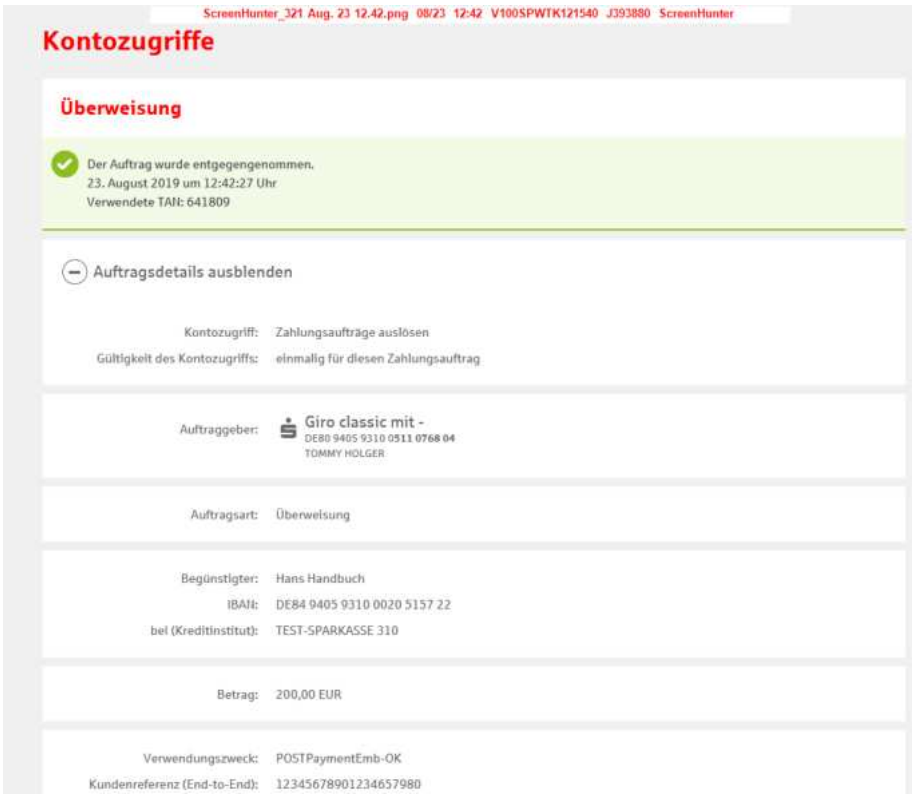
Subsequently, if several TAN methods are available, the user must then select the TAN method and enter the TAN. All TAN methods are supported here (pushTAN and chipTAN) at the same time as all other access channels.

Example selecting a pushTAN connection:





If successful, the user is asked to log off and then returned to the URI (TPP-Redirect-URI) you have saved. In the event of an error, we will redirect to the TPP-Nok-Redirect-URI as long as it has been saved or alternatively to the TPP-Redirect-URI.



Use of the Internet presence is limited to the authorization of the resource you have presented. This is to ensure that the return to your application indeed takes place.

Also, the version of the Internet presence that is designed for redirects does not support management transactions such as PIN changes, changing TAN methods and so forth.

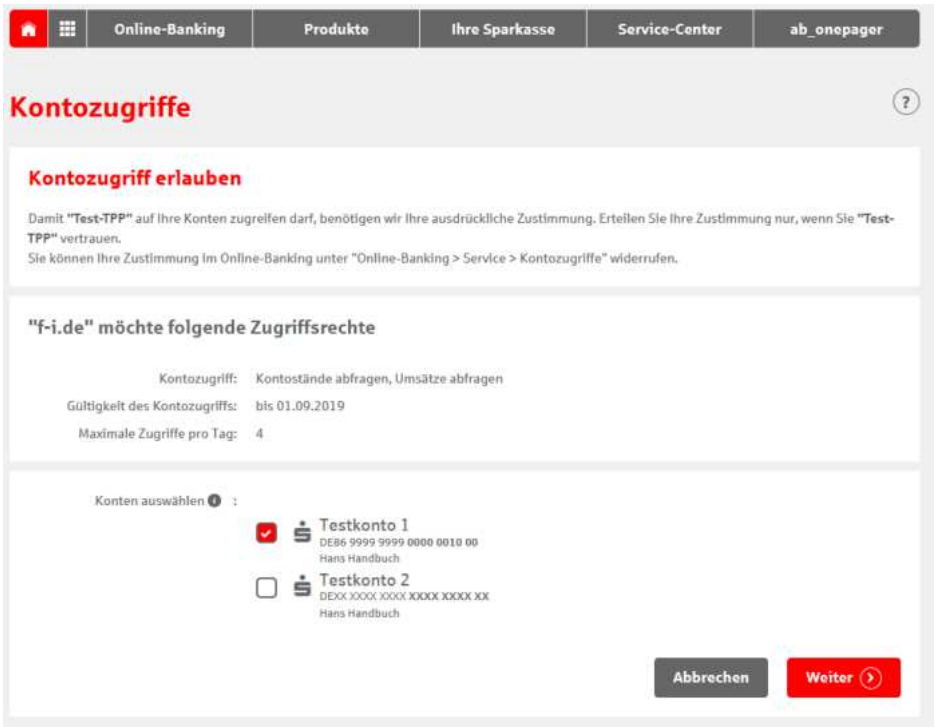
If the user wants to make use of other features, he or she must re-login to the standard Internet presence of his or her Sparkasse/financial institution.

5.1 Creating a bank-offered consent

With a bank-offered consent, the customer may stipulate the accounts to which he or she would like to grant the consent. This is done via the Internet presence.

In the sandbox, the selection by the customer is simulated. An account (DE86 9999 9999 0000 0010 00) is selected by default.

Example of account selection by the customer:



6 Redirecting with AuthCode

If the authorization has been completed successfully, you will receive an AuthCode: <TPP-Redirect-URI>?state=<state>&code=<authorization_code>. You must then use this to retrieve an access token. The procedure is time-sensitive (10 minutes). The access token has to be retrieved immediately in succession.

In the event of an error, no access token is provided.

Exception for a second signature (order confirmation): If a payment has to be authorized by two or more users, you will only receive an AuthCode when the entire authorization is complete.

7 Exchanging AuthCode for OAuth Token

We issue RFC-compliant access and refresh tokens. You can obtain the URI with the "token_endpoint" link using the OAuth-Well-Known endpoint.

Example body of a request:

'code=XJAy0ajuvQfv5FwqAi9lmP_LjllXDVtwbtUECAuTlIA&client_id=meinTpp&code_verifier=E6vTY2ZMFctVP16Ei7617SOeg1gexjIpELwIwOOiVE4&grant_type=authorization_code'

Note: In this endpoint, the attribute is called client_id, while in the URL it is called clientId. We will correct this promptly and then support both spellings.

An example response:

{
"access_token":
"eyJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJQU0QyIiwiU2NvcGUiOiJQSVM6IDEyMzQtd2VydGlxLTk4MyIsImlzcyI6IkZpbmFueiBJbmZvcm1hdGlrIiwiZXhwIjoxNTY2NTUyMzI1fQ.JG0EGwIRgEfiq8mfdoocjEJzGEsMXfetDekSMLI6a4ku4jXdCDk-sN663fNQ4UaXC1SEebs0xAm5IYvMLneedpPcBvAsNLf5vch4RjgeUymv1bYPlaVawXt7wsudhJJCT5iT0_L_Nf_Uwli2G5dcm_WMo4l0JSJgqhSX25gG-giaggLZ-sury3GGC335zJVef8GtH9a42aYMba6KjQtdLbKGIz492hBBVswJe4khaURZ7QyiZ0RnI5huSKnesdHes9C1EL9UL6v3ue8et9yYrO7hIO9lrIc-lacnKweAXCgD8r2EGkmUHsxgruLoD8JBDdtJ5JiUqdU-4W08rmqoAA",
  "token_type": "Bearer",
  "expires_in": "3600",
  "refresh_token": "mnllysEZDis-teXu6S4MItliLhvb8nnVlGwtKvXSc5M"
}

An access token that we have issued is currently valid for one hour (3,600 seconds). You can also derive the validity from the token.

In the initial version, the refresh token is valid for 180 days. In the future we will adjust the validity of the refresh token to match the validity of the resource.

Information about the code challenge and the verifier: Proof Key for Code Exchange (PKCE) support eliminates the risk of an attacker being able to exchange an intercepted authorization code for a token because it also requires the verifier to create tokens. Details on this are published under RFC 7636.

8 Using OAuth Tokens

The OAuth token is required only for accessing successfully authorized resources. This allows you to query the status of resources or authorizations even without an AuthCode or tokens.

The token is mandatory for the following endpoints:

- Read account list
- Read transaction list
- Read balance
- Read account details
- Payment cancellation request (if the payment in question has already been successfully authorized)

The token is optional for all other endpoints.

If a token is sent, then it must exist and be valid. The lifecycle of the access/refresh token is not synchronized with the lifecycle of the XS2A resource. In other words, the status of the resource is always verified, and the request is rejected despite the token being valid. Example: A user (PSU) revokes a consent.