Appendix

Detailed Specifications

1. Standards

The following standards are used in the document under the following abbreviations:

- **BASE32, BASE64, BASE64-URL**: Network Working Group: Request for Comments: 4648 – The Base16, Base32, and Base64 Data Encodings
- **CRT (ICM) Mode**: NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation
- **DER**: ITU-T X.690: Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)
- **JSON**: Internet Engineering Task Force (IETF): Request for Comments: 7159 – The JavaScript Object Notation (JSON) Data Interchange Format
- **JSON Web Signature**: Internet Engineering Task Force (IETF): Request for Comments: 7515 – JSON Web Signature (JWS)
- **SHA-256**: FIPS PUB 180-4 – Secure Hash Standard (SHS)
- **UTF-8**: Network Working Group: Request for Comments: 3629 – UTF-8, a transformation format of ISO 10646

2. Cash Register Algorithm Indicator

This indicator defines the algorithms used and the certification service provider (CSP). Once an algorithm used in the cash register algorithm indicator is no longer specified in the Appendix of the SigV 2008 and is therefore deemed to be non-secure, a new cash register algorithm indicator with secure algorithms must be defined and must not be used with existing cash registers. This indicator corresponds to a character string generated as follows:

RN-CM:

- "R": Fixed prefix
- "N": Index for the used algorithms suite starting with 1
- "-": Fixed separator
- "C": Country code of the CSP
- "M": Index for used CSP within the provided country code in accordance with ISO 3166-1 starting with 1

The following indicators are defined:

R1-CM:

- **CPS**: CM is regarded as a placeholder for the available CPSs. If a closed system is used in accordance with § 20, AT0 shall be specified as CPS.
- **Signature/hash algorithm**: For the creation of the receipt signature pursuant to no. 4, no. 5 of this Appendix. The ES256 algorithm is used in accordance with the JWA (JSON web algorithm) standard.
- **Hash algorithm for the concatenation of receipts and calculation of the IV, number N of the extracted bytes:** SHA-256 is used. The number of extracted bytes and the bytes transferred into the next receipt transferred is 8 (N=8)
- Compression algorithm for a compact representation of the receipt: this algorithm is based on the following methods:
    - Preparation of data to be signed: in accordance with nos. 4 and 5 of this Appendix:
    - Preparation of the machine-readable code: pursuant to nos. 12 and 13 of this Appendix:

3. Export Format Data Collection Protocol

The export format of the data collection protocol corresponds to the following JSON data structure:

- **Receipts group**: The value of this field is a JSON array. The number of the elements in this JSON array corresponds to the number of the signature certificates that were used for the signature of the receipts to be exported. An element of this list therefore corresponds to the following JSON data structure:

- **Signature certificat**e: The value of this field is the BASE64 encoded value of the signature certificate encoded in the DER format.
- **Certification positions**: The value of this field is a JSON array. The elements of the JSON array correspond to the string of all the certification positions that have been used to issue the signature certificate. The value of an element corresponds to the BASE64 encoded value of the certificate encoded in the DER format.
- **Receipts compact**: The value of this field is a JSON array. The elements correspond to the signed receipts shown in the compact representation of the JSON Web Signature Format (pursuant to no. 6 of this Appendix). The sequence of the receipts matches the filing sequence in the DCP. It must be ensured that the concatenation of the receipt signature is entered at position x with the receipt at position x+1 (see field "sig-previous-receipt" in no. 4 of this Appendix).

4. Plain Text Data for the Signature Format for Signing Through a Signature Creation Device

The signature creation is carried out based on the JSON Web Signature (JWS) standard. A receipt is shown in a JSON data structure that contains the data specified in § 9 Sec. 2 nos. 1 to 7 as a minimum. If necessary, the receipt format be can be extended by another JSON data. For the transformation procedure, which is required to create the signature and the machine-readable code, only the receipt data specified in § 9 Sec. 2 nos. 1 to 7 is relevant, which is represented in a JSON data structure as follows:

- **Register ID**: The value of this field corresponds to the value specified in § 9 Sec. 2 no. 1, JSON format *string* UTF-8 encoded.
- **Receipt number**: The value of this field corresponds to the value specified in § 9 Sec. 2 no. 2, JSON format *string UTF-8 encoded*.
- **Receipt Date Time:** The value of this field corresponds to the value specified in § 9 Sec. 2 no. 3, JSON format *string* UTF-8 encoded. The date and the time is saved in the ISO 8601 format without specifying the time zone ("YYYY-MM-DD'D'hh:mm:ss", e.g., 2015-07-21D14:23:34). The time is always based on Austrian local time (CET).
- **Amount Rate Standard**: The value of this field corresponds to the value specified in § 9 Sec. 2 no. 4, JSON format *number* with two decimal places. If no amount has this VAT, then 0.00 is entered.
- **Amount Rate Reduced 1:** The value of this field corresponds to the value specified in § 9 Sec. 2 no. 4, JSON format number with two decimal places. If no amount has this VAT, then 0.00 is entered.
- **Amount Rate Reduced 2:** The value of this field corresponds to the value specified in § 9 Sec. 2 no. 4, JSON format number with two decimal places. If no amount has this VAT then 0.00 is entered.
- **Amount Rate Zero**: The value of this field corresponds to the value specified in § 9 Sec. 2 no. 4, JSON format *number* with two decimal places. If no amount has no VAT then 0.00 is entered.
- **Amount Rate Special**: The value of this field corresponds to the value specified in § 9 Sec. 2 no. 4, JSON format *number* with two decimal places. If no amount has this VAT then 0.00 is entered.
- **Status Sales Counter AES256 ICM**: The value of this field corresponds to the value specified in § 9 Sec. 2 no. 5, JSON format *string*. BASE64 encoded value of the encrypted total sales (pursuant to nos. 8 and 9 of this Appendix).
- **Certificate Serial Number**: The value of this field corresponds to the value specified in § 9 Sec. 2 no. 6, JSON format *string*. UTF-8 encoded.
- **Sig Previous Receipt**: The value of this field corresponds to the value specified in § 9 Sec. 2 no. 7. JSON format *string*. This value is calculated through the cryptographic hash function defined in the cash register algorithm indicator. The result of the signature creation is used as an input of this hash function pursuant to no. 6. For the recognition of the first cash sale, the value of the "Register ID" is used as an input of this hash function. From the hash function result, the N bytes are extracted and BASE-64 encoded starting with byte 0.

The number of the extracted bytes (N) is also defined through the cash register algorithm indicator. The use of access control methods must ensure that the concatenation is displayed correctly via the signature values, even when receipts are created in parallel.

5. Signature Format for Signing Through a Signature Creation Device

The data of a receipt to be signed is specified in § 9 Sec. 2 nos. 1 to 7. This data is transferred to a compressed representation to allow the data to be signed to be represented in a compact form. The transformation is carried out in accordance with the sequence defined in § 9 Sec. 2 nos. 1 to 7. The individual fields are UTF-8 encoded and joined with the character "_" and are saved in a character string. The following representation results from the use of the abovementioned identifier and the notation value (JSON field) for the extraction of the value from the JSON data structure of the receipt.

"Value(Register ID)_Value(Receipt Number)_Value(Receipt-Date-Time)_Value(Amount-Rate-Standard)_Value(Amount-Rate-Reduced-1)_Value(Amount-Rate-Reduced-2)_Value(Amount-Rate-Zero)_Value(Amount-Rate-Special)_Value(Status-Sale-Counter-AES256-ICM)_Value(Certificate-Serial Number)_ Value(Sig-Previous-Receipt)"

The prefix "_RKA_" is then added to the resulting character string. "RKA" represents a placeholder for the cash register algorithm indicator. These indicators are shown in a list (in accordance with no. 2 of this Appendix) and identify the following components:

- Signature/hash algorithm for the creation of the receipt signature
- Certification service provider (CSP) who issued the signature certificate
- Hash algorithm for the concatenation of the receipts and the number of bytes N that have been extracted from the calculated hash value.
- Compression algorithm that was used for the machine-readable code.

The resulting character string corresponds to the *signature format for signing through a signature creation device* and is signed through the JSON Web Signature standard with the signature certificate provided and the selected hash algorithm. In the JWS format, this data is referred to as "JWS Payload".

6. The Result of the Signature Creation

The result of the signature creation is the compact representation as defined by the JWS standard. This character string consists of three BASE64-URL encoded elements separated by the character ".". The three elements correspond to the elements in the sequence provided

1. the meta information on the hash or signature algorithm used
2. the signed data (JWS Payload) and
3. the calculated signature value.

If no digital signature can be created due to a failure of the signature creation device, the UTF-8 encoded character string "safety device failure" BASE64-URL encoded is entered in place of the calculated signature value (third element of the compact JWS representation).

7. Note Regarding the Change of the Signature Certificate

If the signature certificate is changed, it must be ensured that the following receipts may no longer be signed with the certificate that was used before the change.

8. Encryption Method Sales Counter

For the encryption of the sales counter, the AES-256 in ICM (CTR) mode (Integer Counter Mode) is used without padding. Pursuant to no. 9 of this Appendix, the initialization vector contains a calculated hash value which takes into account the receipt number and the register identification number in its calculation. The sales counter in plain text is transferred in an appropriate representation that can be also be reconstructed later without padding information.

The binary data of the AES key is BASE64 encoded to disclose the AES key via FinanzOnline.

9. Encryption

The following procedure applies to encrypt the encoded sales counter:

- **Algorithms**: The AES-256 in ICM (CTR) mode is used. No “padding” is used for the encryption.
- **Initialization vector**: The initialization vector (IV) for the encryption algorithm is a byte array with a length of 16. The UTF-8 encoded cash register identification number (value of the field “Register ID” in accordance with no. 4 of this Appendix) and the UTF-8 encoded receipt number (value of the field “receipt number” pursuant to no. 4 of this Appendix) are joined together in this sequence to calculate the IV. The result is an UTF-8 encoded character string, which is used as the input value for the hash function defined in the cash registration algorithm indicator. The result of the hash function is the hash value displayed in a byte array and bytes 0-15 are extracted from that and used as an IV.

  Note: for each encrypting operation carried out with a provided AES key, it must be ensured that the same IV is never used.
- **Coding the sales value**: The block size of AES-256 corresponds to a byte array with a length of 16. A byte array with a length of 16 is used to code the sales counter in plain text. Each element of the byte array is initialized with 0. The sales counter with the byte number “N” is saved in BIG-ENDIAN format as a “complement on two” representation (“signed”), starting with byte 0. “N” corresponds to the number of bytes required to code the sales counter. Sales counters that are at least 5-byte long must be used.

The result of the encryption is a byte array with a length of 16. Starting with byte 0, N bytes are extracted from the array, BASE-64 encoded and saved in the receipt.

10. Decryption

The decryption is carried out as follows:

- **Algorithms**: see nos. 8 and 9 of this Appendix
- **Initialization vector**: see no. 9 of this Appendix
- **Processing the encrypted sales counter**: A byte array with a length of 16 is created. Each element of the byte array is initialized with 0. Starting with byte 0, the BASE64 decoded byte array of the encrypted sales counter is saved in the created 16-byte long array.

The processed data is decrypted with the AES algorithm and the AES-256 key. The result of the decryption is a byte array with a length of 16. Starting with byte 0, N bytes are extracted from the array, and correspond to the decrypted sales counter. The format corresponds to the “coding the sales value” specified during the encryption.

11. Transfer Format for Data Collection Protocol

Receipts that are transferred to the data collection protocol correspond to a JSON data structure that must have at least the following values/data. The manufacturer can optionally add additional data here. Each receipt must use a minimum of the following data that is saved in a JSON data structure:

- **JWS compact**: The value of this field corresponds to the compact representation of a signature pursuant to JWS standard (in accordance with no. 5 of this Appendix), JSON format *string.*
- **Signature certificate (optional)**: The value of this field is the BASE64 encoded value of the signature certificate encoded in the DER format, JSON format *string*.
- **Certificate positions (optional)**: The value of this field is a JSON array. The elements of the JSON array correspond to the string of all certification positions that have been used to issue the signature certificate. The value of an element corresponds to the BASE64 encoded value of the certificate encoded in the DER format.

The values for the signature certificate and the certification positions remain constant for a long period. They do not therefore have to be transferred for each receipt, but can be made available to the DCP in a different form. It must be ensured that

1. For each receipt, the DCP can create an allocation for the appropriate signature certificate and the certification positions of the signature certificate and
2. All the certificates in the DCP are available to enable the export of the signed receipt data.

When using access control methods it should be ensured that, when transferring the receipt data to the DCP, the concatenation is displayed correctly via the signature values (pursuant to no. 4 of this Appendix) even if the receipt created is carried out in parallel.

12. Preparation details of the Data Contained in the Machine-Readable Code to Verify the Signature Value of a Cash Sale

The data processed for the machine-readable code is represented by a character string containing the following elements:

- **Signed receipt data**: This data corresponds to the UTF-8 encoded character string of the signature format that was transferred to the signature creation device (in accordance with no. 5 of this Appendix). The character string can be extracted from the JWS Payload field of the compact JWS representation (result of the signature creation).
- **Signature value**: The signature value in BASE64 coding is extracted from the compact JWS representation (result of the signature creation). It must be ensured that the signature value in the compact representation of the JWS standard is BASE64-URL encoded to simplify the use in web applications. However, this representation is not suitable for the QR code representation, because it also contains the character "_", which is used for the separation of the elements of the data to be signed. Therefore, the BASE64-URL encoded signature value must be decoded and encoded in the standard BASE64 format.

These two elements are assembled in the sequence specified with the character "_", UTF-8, encoded and processed in a machine-readable code.

13. Verification of the Machine-Readable Code:

The verification of the signature saved in a machine-readable code is processed as follows:

1. **Reading the machine-readable code**: The read UTF8 encoded character string contains the "signed receipt data" and the "signature value".
2. **Extraction of the "signed receipt data" and the "signature value"**: The "signed receipt data" and the "signature value" are extracted from the UTF-8 encoded character string using the separating character "_". The BASE64 encoded signature value is BASE64 decoded.
3. **Processing the Compact Representation Based on the JWS-Signature Standard**: The compact representation (in accordance with no. 5 of this Appendix) is reconstructed from the machine-readable code as follows. The individual elements are joined using the character ".".
   a) **JWS Protected Header:** The signature/hash algorithm of the JWS protected header can be reconstructed through the cash register algorithm indicator. The JWS protected header is saved UTF-8 encoded in the character string at the first position BASE64-URL encoded.
   b) **JWS Payload**: The JWS Payload corresponds to the previously extracted receipt data and is saved in the character string at the second position BASE64-URL encoded.
   c) **JWS Signature**: This value corresponds to the previously extracted signature value and is saved in the character string at the third position BASE64-URL encoded.
4. **Verifying the Signature**: The processed compact representation based on the JWS standard is verified with the appropriate signature certificate.

14. Creation of the OCR-Enabled Character String

Owing to the difficulty of having all the possible characters of a BASE64 character sequence automated and for the realistic lighting conditions and camera properties to be securely recognized, for the OCR-enabled character string the BASE32 representation of binary data is selected in place of the BASE64 representation of the following elements in accordance with nos. 4 and 12 of this Appendix.

- Signature value
- Sig-Previous-Receipt
- Status-Sales-Counter-AES256-ICM

The resulting character string is printed in the OCR-A font on the receipt.

15. Verification of the OCR-Enabled Character String

To verify the OCR-enabled character string, the BASE32 encoded elements must be recoded to the BASE64 coding. The subsequent test procedure is equivalent to a verification of the machine-readable code.

16. OID

The OID identifier for use in the signature certificate is 1.2.40.0.10.1.11.1
"Austrian Tax Authorities Cash Registers Owners".

The OID is assigned from the subtree 1.2.40.0.10.1.11, the "Federal Ministry of Finance Subtree".