Specification

EBICS

(Electronic Banking Internet Communication Standard)

Version 3.0.2

Final Version, June 27th 2022

This specification is valid from December 30th, 2022.

Amendment history

The following table provides an overview of the essential changes from revision version 3.0.1 to the revision version 3.0.2.

Chapter	Type*	Description
1.1	A	Update of the introductory chapter
4.4.1.3	C	Clarification that a bank can act as a "private CA" exclusively for its customers.
10.2.3.1	Ext	Reintroduction of positive and negative end label instead of "neutral" end label.
On top of that, various minor changes (i.e. editorial adjustments, update of references and corrections of obvious errors) were made throughout the document. In particular, final references to PTK, which is no longer included in the EBICS standard, have been removed.		

* E = Error; A = Amendment; C = Clarification; Ext = Extension; D = Deletion

Contents

The XML schema (H005, H000 and S002) can be found on
https://www.ebics.org/en/technical-information/ebics-schema

1 Overview and objectives of EBICS

1.1 Objective of the cooperation

The German banking sector represented by Die Deutsche Kreditwirtschaft (DK) and the French banking sector represented by Comité Français d'Organisation et de Normalisation Bancaires (CFONB) founded a company (EBICS SC) on the joint use of EBICS in 2010. Meanwhile, the EBICS community consists of four countries as the Swiss banking industry, represented by SIX Interbank Clearing and the Austrian banking sector, represented by Payments Services Austria (PSA) also joined the EBICS SC

EBICS was originally developed by the German banking industry and enables corporate clients to conduct their banking business flexibly, securely and efficiently and to select the most suitable services provider for their individual needs. EBICS also has “multi-bank capability”, meaning that in general corporate clients can reach any bank supporting the standard using the same software.

Principally, this specification is valid in general unless an instruction is specified for a particular country relating to a special application of the specification.

Any optional functionality can be supported in one country (and rendered mandatory) and, at the same time, not supported in another country.

The specific use of optional functionalities is described in detail in a common Implementation Guide (chapter 3 of this guide).

By now EBICS is not only used for communication between (corporate) customers and banks. It is also used for the exchange of information between financial market infrastructures.

1.2 General objectives of EBICS

This EBICS (“Electronic Banking Internet Communication Standard”) detailed specification describes the functionality of multi-bank capable, secure communication via the Internet.

EBICS does not present any special requirements of the concrete architecture of the customer’s systems; stand-alone desktop applications can be connected just as easily as e.g. client/server applications or applet solutions.

At the application level, the process “Remote data transmission with customer” is augmented by the concept of Electronic Distributed Signature (EDS), which allows chronologically and spatially-independent authorisation of orders from all customers.

The fundamental features of the EBICS standard are:

- Transmission of professional data (commercial transactions) using established bank-specific formats

- Possibility of the “Electronic Distributed Signature (EDS)”
- Specification of the EBICS-specific protocol elements in XML
- Transmission of messages via http (“Internet-based”); utilisation of TLS for basic transportation security between the customer’s and the bank’s systems as well as between financial market infrastructures, using TLS server authentication
- Cryptographic safeguarding of each individual step of a transaction via encryption and digital signatures at the application level.

The EBICS detailed specification is the basis for the development of customer and bank systems that communicate using the EBICS protocol. As such, it contains manufacturer-independent process descriptions and thereby guarantees interaction between customer and bank systems from different manufacturers.

This detailed specification incorporates the EBICS protocol description and all details relating to code management, EDS and the XML schemas for the order data of the administrative (technical) EBICS order types. The complete XML schemas are stored as separate HTML documents.

The detailed specification only limits the processing freedom of the customer and bank systems with specifications and provisions where this is necessitated by security considerations or processes beyond the scope of the EBICS communication. In contrast to the EBICS Implementation Guide, implementation alternatives will not be indicated in the detailed specification.

2 Definitions

2.1 Terms

The following terms in small capitals have a special meaning in the protocol definition:

- **MUST:** denotes a compelling requirement; only those implementations that fulfil this requirement are deemed to be EBICS-conformant.
- **SHALL/SHOULD:** denotes requirements that are to be followed under normal circumstances; however, individual exceptions are possible for technical or professional reasons.
- **CAN/MAY:** denotes unbinding recommendations or optional features.

Functionalities or features of the EBICS protocol are designated as **optional** if they do not have to be supported by the financial institution. Customers do not have a legal claim to the corresponding functionality from the financial institutions.

Functionality or features of the EBICS protocol in a particular version are designated as **planned** if they are being prepared for subsequent versions but may not yet be used in the present version.

This specification is addressed to software vendors. The terms mentioned obove refer to the requirements for implemention supporting EBICS functionality. It's no requirement for the formulation of contracts between customer and bank.

2.2 Notation

2.2.1 XML

2.2.1.1 XML schema

The following symbology is used for graphical representation of XML schemas:

- Elements are placed in rectangles.
- Attributes are also placed in rectangles and are surrounded by an "attributes" box.
- Elements, attributes and other declarations that belong to a complex type are surrounded by a dashed box that is highlighted in yellow.
- A "branch" (corresponds to `choice` in XML schema) is shown as an octagon containing a switch symbol for three possible switch positions. The connecting lines to the possible alternatives branch out on the right of the symbol.
- A "sequence" (corresponds to `sequence` in XML schema) is shown as an octagon containing a line symbol with three points on it. The connecting lines to the individual sequence elements branch out on the right of the symbol.

- Symbols with solid edges denote mandatory use, and in the XML schema correspond to the attribute `minOccurs="1"` for elements or `use="required"` for attributes.
- Dashed symbols denote optional use, and in the XML schema correspond to the attribute minOccurs="0" for elements or use="optional" for attributes.
- Crossed-out symbols denote planned usage, and in the XML schema correspond to the attribute combination `minOccurs="0" maxOccurs = "0"` for elements or `use="prohibited"` for attributes
- "m..n" in the right lower corner of an element symbol restrict the use of the element to m- to n-times occurrence, and in the XML schema correspond to `minOccurs="`*m*`" maxOccurs="`*n*`"`; correspondingly, where "m..∞" `minOccurs="`*m*`" maxOccurs="unbounded"`
- Element groups are represented by octagons, and correspond to the `group` declaration in the XML schema
- Attribute groups are surrounded by boxes with the respective group names and correspond to the `attributeGroup` declaration in the XML schema.

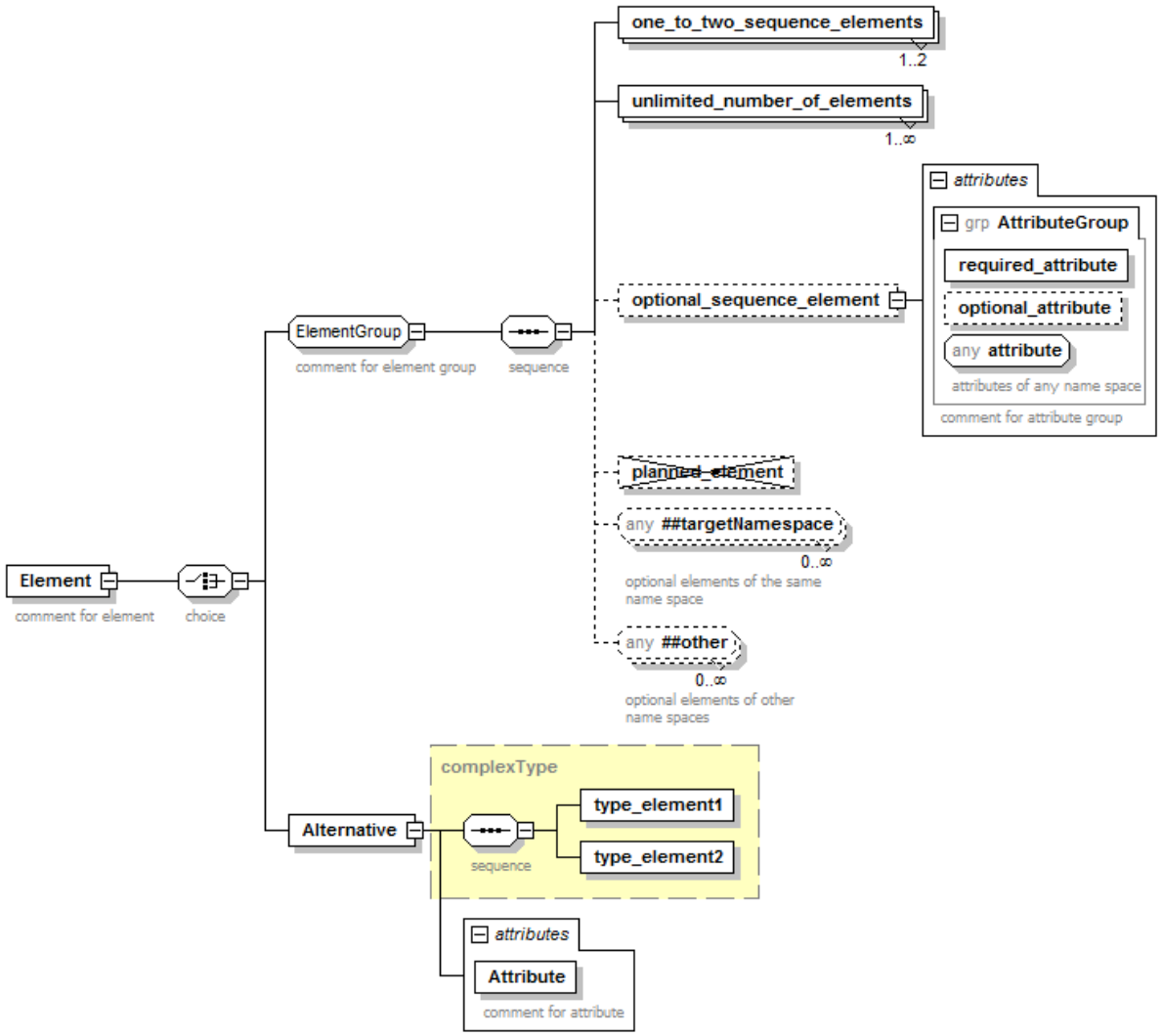


Diagram 1: XML schema symbols

2.2.1.2 XML documents

Individual code segments are shown in the Courier font.
If an element name or type does not fit completely onto a line, the symbol » is used to direct the reader to the next line.

```
Complete examples of code are shown in Courier 8pt and are surrounded by a frame.
```

2.2.2 Flow diagrams

Processes are represented with the help of UML 2.0 activities. In this document they receive a start and an end node. A start node is the starting point of a process, the end node marks the end of an entire process.

For the sake of simplicity, activities will be nested within one another. Actions that contain an activity will be marked with a fork symbol. Activity A in Diagram 2 comprises three process steps (actions). Step_2 is itself an activity comprising 2 process steps. Hence the activity Step_2 is called up within activity A, i.e. run through from the start node of Step_2 to the end node of Step_2.

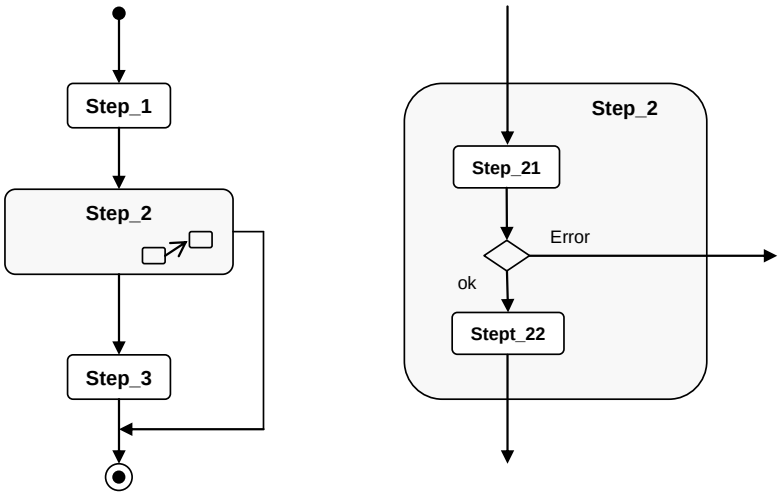


Diagram 2 Nesting of activities

2.2.3 Other notation

In the naming of new administrative order types, the appended tag “[mandatory]” denotes that the financial institution MUST support this administrative order type. On the other hand, the appended tag “[optional]” means that the financial institution CAN support this administrative order type.

Similarly, the tag “[planned]” is appended to planned features or functions.

2.3 Data types

The XML schema defines a set of primitive and derived data types that can be used to form your own data types.

The following primitive data types are primarily used in conjunction with EBICS:

- **string**: string of characters with unrestricted length and structure
- **boolean**: boolean truth value with the characteristics “true” (=1) or “false” (=0)
- **decimal**: decimal numbers to any degree of accuracy
- **dateTime**: time stamp with date and time in accordance with ISO 8601
  The structure is as follows: YYYY-MM-DDTHH:MM:SS.sss**Z**. The character Z

indicates that date and time have been converted to UTC. If the date string does not correspond to this structure, the EBICS message has to be declined or the ES verification has to be rated as negative, respectively.

- **date**: date in accordance with ISO 8601
- **hexBinary**: hexadecimal value with unrestricted length
- **base64Binary**: data type to record base64-coded binary data
- **anyURI**: uniform resource locator (e.g. URL, IP address).

The following pre-defined data types are derived from primitive data types and are used in the EBICS standard:

- **normalizedString**: string of characters that has spaces (blanks) removed at the start and end
- **token**: a normalizedString that contains no line feeds and no multiple spaces in succession
- **language**: nationality label in accordance with RFC 1766
- **nonNegativeInteger**: non-negative integer values
- **positiveInteger**: positive integer values.

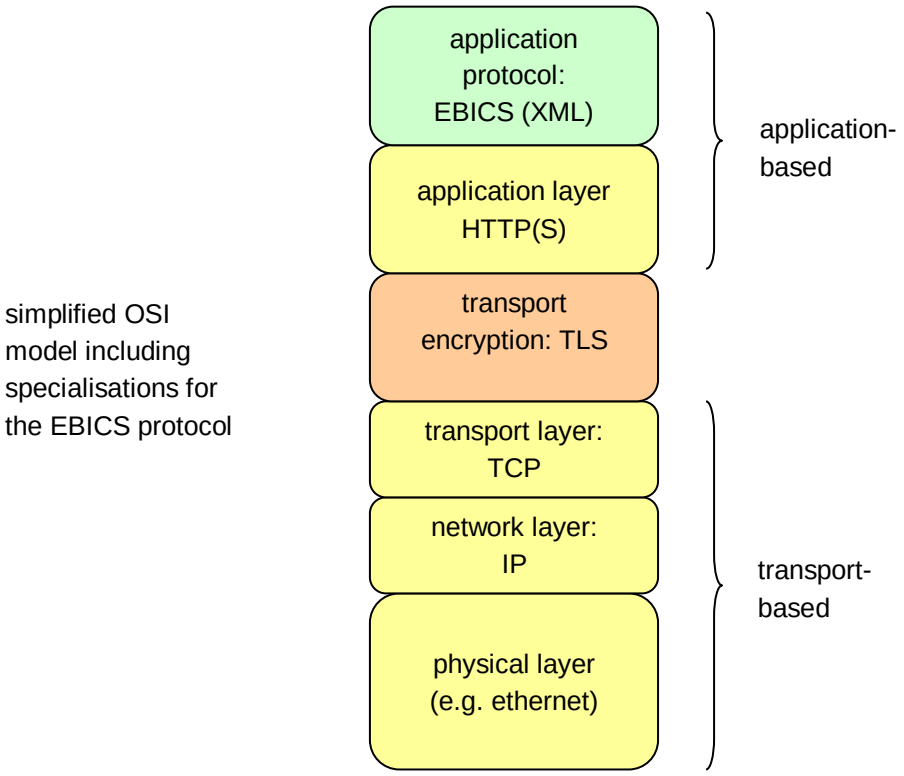
With the help of the aforementioned data types, new data types are defined in the EBICS schema:

- **simple data types** merely define restrictive or expanding characteristics with regard to the value range of an existing primitive or derived data type, i.e. they derive from an existing data type
- **complex data types** define new structures composed of fields and attributes of different (simple or complex) data types.

3 Design decisions

This chapter will point out decisions that had a decisive influence on the design of the EBICS protocol. It includes network-specific details as well as specifications of a professional and technical nature.

3.1 OSI model from EBICS perspective



3.1.1 TCP/IP as package-orientated transmission layer

TCP/IP is used as a transport protocol. The data that is to be exchanged is transmitted as packages via IP (Internet Protocol). This package transfer is monitored by TCP (Transmission Control Protocol) as a transmission monitoring protocol.

Communication is established using a URL (Uniform Resource Locator). Alternatively, an IP address belonging to the respective financial institution can also be used. The URL or IP address together with the EBICS host ID is required for establishing a connection to the bank computer and is given to the customer upon conclusion of the contract with the financial institution.

3.1.2 TLS as transport encryption

TLS was developed by the Transport Layer Security work group in IETF's Security Area. It is an open standard for secure transmission of package-orientated data, originally developed by Netscape (initially under the name SSL). TLS aims to guarantee data security on levels above TCP/IP. The protocol allows data encryption, authentication of servers and message integrity for TCP/IP communication.

It combines the following basic characteristics:

1. The TLS connection is confidential: With the TLS handshake, a common, secret key is agreed using asymmetric encryption (RSA, in the case of EBICS) that serves as a symmetric key (AES in the case of EBICS) in the rest of the TLS session.
2. The integrity of the TLS connection is assured: The message transport contains a message integrity verification via so-called Message Authentication Codes (MACs). Secure hash functions (SHA 256 in the case of EBICS) are used for the MAC evaluations.
3. The identity of the financial institution is attested by the use of server certificates and electronic signatures; the messages from the financial institution are authenticated by means of this TLS server authentication.
4. TLS contains mechanisms to protect against man-in-the-middle attacks on the TLS connection between customer and bank systems. To this end, it uses internal counters and "shared secrets", and additionally secures the handshake against such an attack with signed summaries of the data exchanged thus far.

A TLS connection is established between the customer system and the bank system for transmission of the EBICS messages between these two systems.

TLS (details see current EBICS Annex "Transport Layer Security") with X.509v3 server certificates is used, i.e. the server MUST authenticate itself via certificate. The type of certificate MUST be suitable for the key exchange algorithm of the selected key.

EBICS dispenses with TLS client authentication in Version H005 to promote better market acceptance. Later expansion to include TLS client authentication capability (and the associated issue of X.509v3 client certificates for TLS to customer systems) is not excluded.

Details regarding pre-distribution and verification of the trust anchors see EBICS Annex Transport Layer Security (Chapter 3, Validation of TLS server certificates).

3.1.3 HTTP(S) as a technical basic protocol

The Hypertext Transfer Protocol (HTTP) is a stateless data exchange protocol for the transmission of data. HTTP is predominantly used in the “World Wide Web” (WWW) for the transmission of websites.

The combination of HTTP and TLS as transport encryption is also referred to as “HTTPS” (HTTP Secure). Port 443 (SSL) is reserved for this purpose and can be used in an unrestricted manner by the majority of firewall configurations.

In the case of the EBICS protocol, the statelessness of HTTP forces the use of its own session parameters that logically combine several communication steps into one transaction.

Communication between the customer and the financial institution takes place in a classical manner via client/server roles. As before, the financial institution also takes on the (passive) server role and the customer takes on the (active) client role. With this communications schema, the client sends a request to the server via HTTP request; the server replies with an HTTP response. The request can generally be made as a GET request (additional data coded in the URL) or a POST request (additional data appended to the HTTP header); in the context of EBICS, POST is used exclusively.

With EBICS, HTTP 1.1 MUST be used by both the client and the server.

3.1.4 XML as an application protocol language

The EBICS application protocol uses the HTTP(S) technical base protocol. XML (Extensible Markup Language) has been selected as the protocol language on the application level. The following reasons are given for this decision:

1. XML uses readable tags. Tag names/attributes can be selected in such a way that their meaning is obvious even without documentation.
2. Freeware XML parsers are available for common operating systems and programming languages.
3. XML messages can easily be expanded with additional elements and attributes. It is not necessary to adapt the existing message sections to maintain the syntactic correctness (“well-formedness”) of the message as a whole.
4. XML schema is available as a standardised definition language for validation of XML messages.

UTF-8 MUST be used for character encoding within the EBICS XML message. UTF-8 is supported by all XML parsers and codes backwards compatible to ASCII.

The syntax of the XML messages is set with the help of so-called XSD files (XML Schema Definition). The following XSD files have been defined for EBICS and can be downloaded from https://www.ebics.org/en/ebics-schema

- “ebics_request_H005.xsd”
  contains the XML schema for requests from the customer system
- “ebics_response_H005.xsd”
  defines the XML schema for responses from the bank system
- “ebics_orders_H005.xsd” contains order-specific data structures
- “ebics_types_H005.xsd” lists simple EBICS type declarations.

In addition to these main schemas, the following specific variants for transactions that relate particularly to key management can also be found at the same place:

- “ebics_keymgmt_request_H005.xsd“ defines the XML schema for requests from the customer system within the framework of key management.
- “ebics_keymgmt_response_H005.xsd“ contains the XML schema for responses from the bank system within the framework of key management.
  The schema "ebics_signature_S002.xsd" has been defined for submitting the ES in structured form. It can also be downloaded from
  https://www.ebics.org/en/ebics-schema
- This schema has been defined as an independent one in order that it can be applied outside the EBICS domain. The import of the aforementioned name space is required for use of the ES in EBICS. It features the prefix "esig".
- The schema "ebics_signature_S002" references to structures of the XML signature standard of the W3C (see chapter 3.8). This schema is stored at the same place under the name "xmldsig-core-schema.xsd".

Each of the four XSD files with the extension "_request_H005" or "_response_H005" defines one or more types of EBICS XML messages each of which possesses a different XML root element with an unambiguous name.

For Standard EBICS messages "ebics_request_H005.xsd“ defines the root element `ebicsRequest` for customer system requests whereas "ebics_response_H005.xsd“ defines the root element `ebicsResponse` for responses of the bank system. For transactions of the key management "ebics_keymgmt_request_H005.xsd“ contains three additional XML messages for customer requests with the root elements `ebicsUnsecuredRequest`, `ebicsUnsignedRequest`, and `ebicsNoPubKeyDigestsRequest`. For key management "ebics_keymgmt response_H005.xsd“ defines the root element `ebicsKeyManagementResponse` for responses of the bank system.

„ebics_H005.xsd“ includes these four XML schema files and therefore contains the whole range of definitions of the EBICS schema version “H005”. It can also be downloaded from https://www.ebics.org/en/ebics-schema. Its target namespace is also “urn:org:ebics:H005”:
By means of this file can be verified that all global definitions in the EBICS namespace (elements and types) have unambiguous names. This feature of the EBICS XML protocol facilitates the processing of EBICS XML messages with the help of standard XML tools because the declaration of the XML root element and the EBICS namespace are already sufficient to determine the allowed format for the complete XML message. A standard XML parser, for example, is able to recognize by this XML fragment against which definition in the EBICS XSD files the whole document has to be vaildated:

```
<ebicsRequest xmlns="urn:org:ebics:H005" Version="H005">
```

The Schema Location consists of one pair of references, separated by a blank. The first member of the pair is the namespace name, and the second member is a hint describing where to find an appropriate schema document for that namespace, e.g. local file name ebics_request_H005.xsd:

```
 <ebicsRequest xmlns="urn:org:ebics:H005" Version="H005"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation=
"urn:org:ebics:H005 ebics_request_H005.xsd">
```

By means of the following example taken from the XML schema file "ebics_request_H005.xsd" the referencing of EBICS XML elements and attributes for the EBICS root structure regarding standard requests of the customer system (root element `ebicsRequest`and its sub-elements) is illustrated:

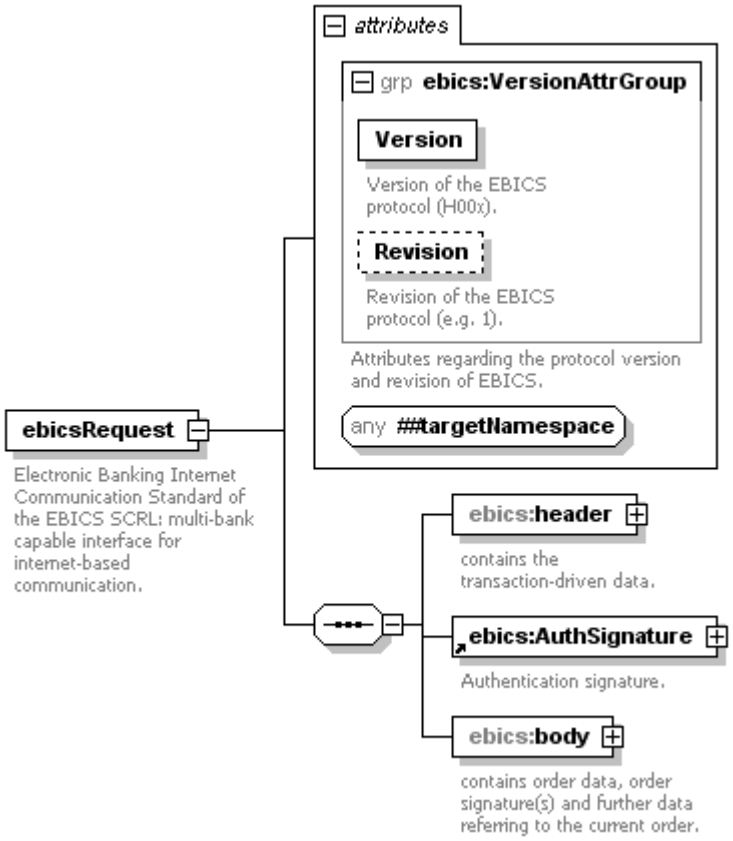


Diagram 3: Root structure of the EBICS protocol

The XML root element for standard EBICS messages containing requests of the customer system is called `ebicsRequest`. It contains some attributes with fundamental information that are required for parsing the message as a whole (attribute group `VersionAttrGroup`):

- `Version` for the EBICS protocol version (e.g. “H005”)
- `Revision` for the EBICS protocol revision: This attribute SHOULD also be sent to allow technical differentiation between several (compatible) revisions of the same protocol version.

The following elements form the direct sub-structure of `ebicsRequest`:

- `header`: The XML tag contains technical information (so-called “technical control data”) in the subtags:
  - `static` for the technical control data that remains constant throughout the entire transaction.
  - `HostID` for the EBICS host ID for the identification of the bank's EBICS computer system. The element `HostID` is contained in all EBICS request messages of the customer system (for standard transactions as well as system-related transactions). The EBICS host ID is communicated to the customer by the financial institution.
  - `mutable` for the mutable technical control data.

Both subtags of `header` MUST appear in the above sequence.

- `AuthSignature`: The identification and authentication signature according to the "XML Signature" standard is disposed in this element. The XML tag MUST appear in all messages with the exception of the administrative order types INI, HIA and HPB (these administrative order types use their own XML schemas; see Chapter 4.4).
- `body` contains the actual order data, signatures (ES's) and other data that is directly related to the order or that is required for its evaluation.

The XML requests from the subscribers to the financial institutions are designated as EBICS requests, the XML replies from the financial institutions are designated as EBICS responses. The HTTP binding of an EBICS request and the associated EBICS response is: the EBICS request is embedded in an HTTP POST request, the EBICS response is embedded in the corresponding HTTP response.

A typical HTTP request appears as follows in EBICS (extract):

```
POST /ebics HTTP/1.1
Host: www.die-bank.de
Content-Type: text/xml; charset=UTF-8
Content-Length: 800

<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest xmlns="http://urn:org:ebics:H005"
       xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="urn:org:ebics:H005 ebics_request_H005.xsd”
       Version="H005" Revision="1">
  <header authenticate="true">
    <static>
      <HostID>EBIXHOST</HostID>

      …
    </static>
    <mutable>
      …
    </mutable>
  </header>
  <AuthSignature>
    …
  </AuthSignature>
  <body>
    …
  </body>
</ebicsRequest>
```

A corresponding possible HTTP response is shown in the following extract:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=UTF-8
```

```
Content-Length: 1538

<?xml version="1.0" encoding="UTF-8"?>
<ebicsResponse xmlns="urn:org:ebics:H005"
       xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="urn:org:ebics:H005 ebics_response_H005.xsd"
       Version="H005" Revision="1">
  <header authenticate="true">
    <static>
      …
    </static>
    <mutable>
      …
    </mutable>
  </header>
  <AuthSignature>
    …
  </AuthSignature>
  <body>
    …
  </body>
</ebicsResponse>
```

Further details on the structure of EBICS protocol messages and transaction details can be found in Chapter 5. The formats of the XML messages for the standard responses of the bank system and the system-related messages of the key management use different root elements the structure of which is widely analogous to the standard request. The complete XML schemas can be found in the separate HTML schema documentation.

The schema "ebics_hev.xsd" (also not renamed/updated for EBICS 2.5) which is used for requests of EBICS versions supported by the bank is provided at https://www.ebics.org/en/ebics-schema (details see also chapter 9.5).

3.2 Compression, encryption and coding of the order data

EBICS handles bank-technical order data in a transparent manner. This means: independent of the specific data structure of different administrative order types, order data is handled as a binary block and is embedded in the XML structure. To this end, this order data MUST initially always be ZIP-compressed before transmission, then hybrid encrypted (in accordance with process E002) and the result finally base64-coded. Exceptions: In the case of the administrative key management order types INI, HIA and H3K transmission is unencrypted (see Chapter 4.4.1.2.5.1 for INI, HIA and H3K). The standards that define the ZIP algorithm and base64 format that are valid in EBICS are specified in the Appendix (Chapter 15).

The data representation generated in this way is then to be set, for example, in the XML element `ebicsRequest/body/DataTransfer/OrderData` without any further character conversion.

The ZIP compression serves to reduce the data volume that is to be transmitted.

The actual data is symmetrically encrypted in the case of hybrid encryption. The transaction key that is used for this purpose is again asymmetrically encrypted and is appended, for example, in the form of the XML element `ebicsRequest/body/DataTransfer/DataEncryptionInfo/TransactionKey` (see Chapter 6.2).

Encryption of the order data takes place in addition to TLS transport encryption. This ensures that the order data is protected from unauthorised read access both on its way via public networks (in addition to TLS) as well as on the other side of the TLS-protected connection path.

For coding the binary stream, base64 only uses printable ASCII characters and thus ensures that the order data reaches its destination in an unadulterated manner and can be evaluated there as authentic.

3.3 Segmentation of the order data

Segmentation means the separation of large data volumes into smaller, individual transmission segments.

With EBICS, segmentation of the order data takes place at the application protocol layer. Order data may only be transmitted in an individual EBICS message if it does not exceed the specified fixed size of 1 MB in compressed, encoded and base64-coded form. This applies equally to transmit and download orders. If the 1 MB limit is exceeded, the compressed, encrypted and base64-coded order data MUST be separated into segments, wherein the size of each of these does not exceed the fixed segment size of 1 MB. The segments are then transmitted in consecutive order in individual EBICS messages.

Further details on segmentation of order data can be found in Chapter 7.

3.4 Recovering the transmission of order data (recovery) [optional]

Recovery allows the transmission of an order to be continued after the occurrence of a transport or processing error without necessitating the re-transmission of all order data segments that have already successfully been transmitted.

EBICS defines a recovery process at the XML application protocol layer that is based on the sequence of transmission of order data in several fixed, pre-

determined steps. It is an optimistic recovery process that dispenses with a separate synchronisation step since the customer's system generally knows the step from which transmission of the order in question is to be continued.

Details relating to recovery can be found in Chapter 5.4.

3.5 Electronic signature (ES) of the order data

The "**E**lectronic **S**ignature" (ES) of the order data ensures the authenticity of the order data on the other side of the TLS transmission path, independent of the compression, encryption, coding and segmentation of the order data.

In the case of upload orders this is the deliberate signature of a subscriber that documents the content commitment of the subscriber, in the case of download orders it is the signature of the financial institution.

ES's are generated in accordance with the Appendix (see Chapter 14), in EBICS Version "H005" a minimum requirement is support of ES Version "A005" (see Appendix (Chapter 14)).

3.5.1 Subscriber's ES

The order data of upload orders MUST be signed before delivery, i.e. provided with at least one ES. Exceptions are the administrative key management order types INI and HIA, which are not signed in a bank-technical manner.

According to the signature process used and, regarding EBICS, supported by the bank, the bank system can extract the hash value of the signed order data from a subscriber's ES with the help of the signatory's public signature key.

The following **signature classes** are defined for the ES's of subscribers, listed here in order of reducing strength ("E" is the strongest and "T" is the weakest signature class):

- Single signature (type "E")
- First signature (type "A")
- Second signature (type "B")
- Transport signature (type "T")

An authorisation model for ES's is defined within the financial institution by the assignment of signature classes to subscribers. For example, subscribers with signature class A are entitled to provide first signatures for orders. Detailed authorisation models CAN be defined individually for institutions, wherein the

signature authorisation of a subscriber can be parameterised with regard to the BTF identifiers and/or the amount limit and/or the account used.

The signature class of a subscriber's given ES is the strongest class that can be assigned to this ES in the authorisation model of the corresponding financial institution.

Signature authorisations of type "T" are assigned globally to subscribers or (in detailed authorisation models) to subscribers in combination with certain BTF identifiers. However, they are not dependent on accounts or amount limits. Transport signatures are not used for bank-technical authorisation of orders, but rather merely for their (authorised) submission to the bank system.

Bank-technical ES's are deemed to be ES's of type "E", "A" or "B". Bank-technical ES's are used for the authorisation of orders. Orders can require several bank-technical ES's, which MUST then be supplied by different subscribers. Subscribers of different customers can also be the signatory of an order.

A **minimum number of required bank-technical ES's** will be agreed between the financial institution and the customer for each supported BTF identifiers.

Details on the format of the ES and its application for order authorisation are given in the Appendix (Chapter 11.2).

3.5.2 Financial institution's ES [planned]

The ES of the financial institution is a *planned* functionality of EBICS. The prerequisite for the use of this function is a definitive legal view relating to it.

Preparations have been made both in this detailed concept and also in the EBICS XML schema files that will facilitate the implementation of the following stipulations in future versions of EBICS:

- mandatory ES of the hash value and the display file of the order that is to be signed in the case of administrative order type HVD (see Chapter 8.3.2.2).
- Primed in the schema but not usable yet: ES of download data in the case of download orders.
- Download of the financial institution's public bank-technical key via order type HPB See Chapter 4.4.2.2
- Verification of the hash value of the financial institution's public bank-technical key within the framework of transaction initialisation.
  It is a component of each transaction initialisation to verify that the financial institution's public key that has been made available to the subscriber. Exceptions are the administrative key management order types INI, HIA, HPB as well as the administrative order type HEV for the request of EBICS versions supported by the

bank.
See Chapter 4.6.2 and Chapter 11.1.2.

- Binary format for the financial institution's ES is analogous to the subscriber's ES
  See Appendix (Chapter 11.2.2).

3.5.3 Representation of the ES's in EBICS messages

The ES's of an order are represented with the help of the XML element `UserSignatureData` (for the subscriber ES, which is defined in the schema file "ebics_signature_S002.xsd"). The structure `BankSignatureData` (for the bank ES) is not usable yet (planned feature).
Each of these substitutes the abstract element `EBICSSignatureData`. Diagram 4 contains the graphical representation of `EBICSSignatureData`: A005/A006 are contained in `OrderSignatureData`. ES's are configured in accordance with the Appendix (Chapter 14). In this structured format for signature processes from A005/A006 on, the customer ID is already contained in the element `PartnerID`. The declaration of a differing customer ID for the ES distributed among a number of customers is only possible with order type HVE by way of special order parameters. The financial institution's bank-technical ES is configured analogously to the known subscriber's bank-technical ES (see Appendix (Chapter 11.2.2) in comparison with Appendix (Chapter 14)), wherein the attribute `PartnerID` is dispensed with in this case.

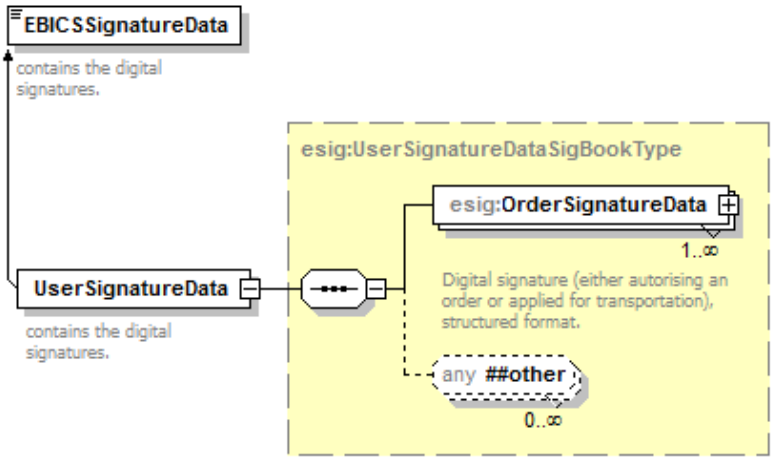


Diagram 4: XML structures UserSignatureData for the ES's of an order (in structured format)

The following steps are necessary to embed the ES's of an order in EBICS messages:

- Issue of an instance document to ebics_orders_H005.xsd or ebics_signature_S002.xsd that only comprises the element `BankSignatureData` (bank ID) or `UserSignatureData` (subscriber ID).

- ZIP compression, encryption, base64-coding of the instance document
  Encryption takes place with the transaction key `TransactionKey` from the XML branch `ebicsRequest/body/DataTransfer/DataEncryptionInfo` (see Chapter 6.2).
- The result is set in the element `SignatureData` in the branch `DataTransfer` of the EBICS body (see Chapter 3.1.4).

3.6 Preliminary verification [optional]

In the case of upload orders, the subscriber CAN send information in a first transaction step that the bank system CAN use for prevalidation of the order – insofar as it supports this functionality. Prevalidation can comprise one or more of the following checks: Account authorisation verification, limit verification, ES verification. If (technical) errors occur during prevalidation, it is pointless to continue transmission of the order – particularly since the order cannot be carried out.

Subscribers can discover whether a financial institution generally supports prevalidation via the bank parameter query (administrative order type HPD, returned XML structure `HPDResponseOrderData`, attribute `ProtocolParams/PreValidation@supported`). Supplied parameters for prevalidations that are not supported by a financial institution are ignored by the financial institution.

More details on the administrative order type HPD can be found in Chapter 9.2. See Chapter 5.3 for details on prevalidation.

3.7 Technical subscribers

EBICS customer systems can in turn be set up as client-server systems, so-called multi-user systems. In this case, the server takes on the part of the EBICS client within the communication with the bank system and as such is responsible for the transmission of orders in accordance with the EBICS specification.

Towards the bank system, this customer-sided server acts as a "technical subscriber" which essentially is administrated within the bank system like a (human) subscriber.

EBICS requests of a technical subscriber and a human subscriber differ from each other only in the point that for all EBICS requests, the technical subscriber allocates his subscriber identification to the field SystemID and generates the identification and authentication signature for the EBICS request.

EBICS responses for the technical subscriber are always encrypted with the technical subscriber's public encryption key.

The following applies to the technical subscriber:

- On principle, the technical subscriber's identification is assigned to the field SystemID (in addition to the fields PartnerID and UderID) in the EBICS request. By the presence of the field SystemID, the bank system detects that the request has been sent by a technical subscriber.
- The technical subscriber issues the identification and authentication signature for the EBICS request (except of the administrative order types which do not require an identification and authentication signature).
- The technical subscriber can execute all EBICS requests for the subscriber who is stated in the field UserID.
- The technical subscriber cannot issue a bank-technical signature.
- The technical subscriber can submit files with a particular transport signature (D file or submission to the EDS).
- The technical subscriber can submit files with bank-technical signatures of human subscribers. In this case, the technical subscriber does not have to issue a transport signature.

The following applies to the bank system's verification:

- The verification of the identification and authentication signature of the EBICS request issued by the technical subscriber is performed on the basis of the contents of the field SystemID.
- The order authorisation is verified by the contents of the fields PartnerID and UserID. The content of the field SystemID is not relevant.
  Only if the technical subscriber performs EBICS requests under his own name (the field UserID contains the technical subscriber's identification), the according order authorisation is required for the bank system.
- An account verification is not performed for technical subscribers.
- As usual, the electronic signature is verified independently of the contents of the fields SystemID and UserID.

3.8 Identification and authentication signature

Identification and authentication of the subscriber or the customer system and the financial institution is necessary in each transaction step to prevent the use of resources by unauthorised persons at the bank’s end and to prevent unauthorised state alteration of orders or data.

The identification and authentication signature represents an integral component of the EBICS protocol as a main XML branch between the EBICS header and body

data. It is generated in accordance with the XML signature standard and has a number of tasks to fulfil:

1. Identification and authentication of the (technical) subscriber: With the help of the identification and authentication signature, the bank system MUST convince itself of the correctness of the (technical) subscriber identification of known subscribers or customer systems.
2. Integrity of the control data/ES(s): Changes – even on the other side of the TLS transmission path - to the ES(s) as well as the technical and order-related data (with the exception of order data that is not acquired from the identification and authentication signature but rather from the bank-technical signature) are detected with the help of the identification and authentication signature as long as the XML structure of the signed data remains unchanged.

The identification and authentication signature (in contrast to the ES that signs the order data) is configured via the control data and via the ES(s) and MUST be supplied by both the customer system and the bank system in every transaction step of each administrative order type (with the exception of the system-related order types INI, HIA, H3K and HPB, see Chapter 4.4). Identification and authentication of the bank-technical ES(s) connects the order data that is signed with this/these ES(s) to the remaining protocol information and thus prevents the unauthorised exchange of orders together with their ES(s) within an EBICS transaction.

Details on the identification and authentication signature algorithms that are used can be found in Chapter 11.1. It is also stipulated here that a canonisation process (C14N) transmits the data in standardised format before generation and verification of the signature.
In addition to the XML signature's inherent structures, precisely those elements (and their substructures) that possess the attribute marker `@authenticate="true"` MUST go into the identification and authentication signature for signature configuration. The occurrence of these attribute markers are stipulated in the XML schema.

The identification and authentication signature of each EBICS message MUST be verified by the respective message recipient.

If the identification and authentication signature of an EBICS request cannot be successfully verified, the bank system cannot assume that the EBICS request actually originates from the corresponding (technical) subscriber.

In this event, the sender of the EBICS request will receive a corresponding error code (EBICS_AUTHENTICATION_FAILED). Further details can be found in Chapters 5.5.1.2.1 and 5.5.1.2.2, in each case under the sub-heading "Verifying the authenticity of EBICS requests".

If, on the other hand, the identification and authentication signature of the EBICS response cannot be successfully verified, the customer system cannot assume that the EBICS response originates from the expected bank system. In this event, the relevant EBICS transaction MUST be aborted by the customer system.

The settings of the customer's software that is used to establish the connection to the bank system, complying with the requirements of Chapter **Fehler! Verweisquelle konnte nicht gefunden werden.**, MUST then be verified at the customer's end. Furthermore, it MUST be verified whether the financial institution's public keys are up-to-date (see also Chapter 5.5.1.2.1, sub-heading "Verifying the hash values of the bank keys").

3.9 X.509 data

For cryptographic algorithms (i.e. for identification and authentication, encryption, signature), Version H005 of the EBICS protocol uses public keys in X509-standard that have been exchanged within the framework of subscriber initialisation between subscriber and financial institution (for key management see Chapter 4)
The structures of the W3C are referenced directly.
The element group for this is located in the EBICS XML schemas "ebics_request_H005.xsd" and "ebics_keymgmt_request_H005.xsd", for example in the path `ebicsRequest/body/X509Data`. The type definition uses the specification from the XML signature (see Diagram 5).

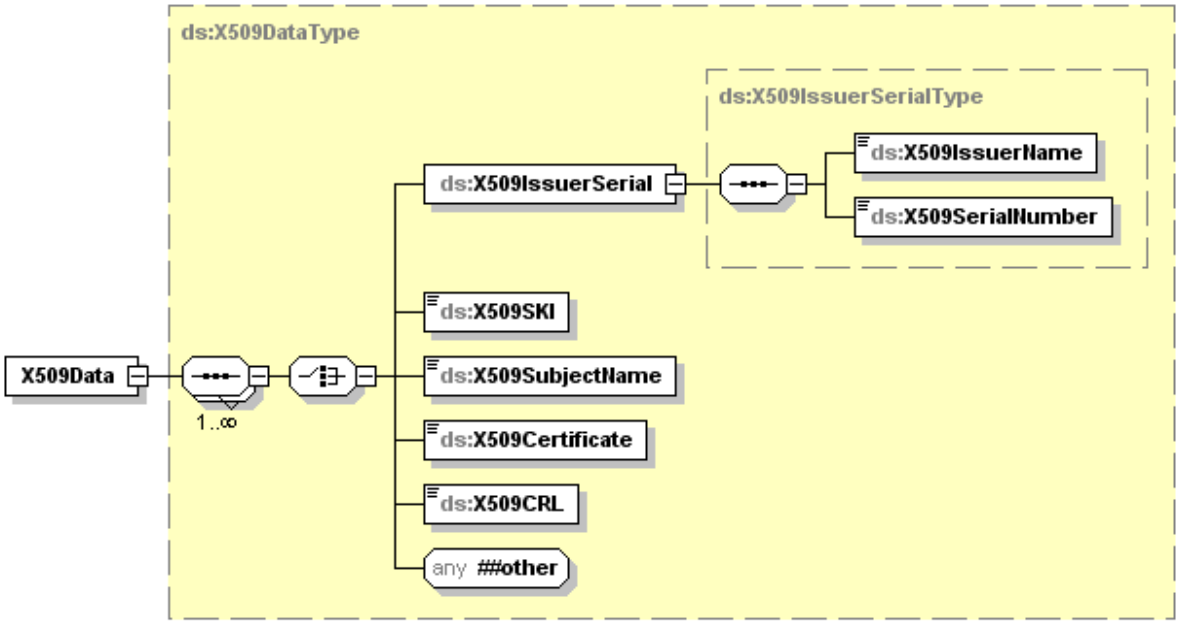


Diagram 5: X509DataType

In EBICS schema version H005, administrative order types of the key management require X.509 data to be transmitted together with public keys.

3.10 Supported administrative order types

All standardised, system-related and reserved administrative order types in accordance with the complete list (see Appendix Chapter 13) are supported by transparent embedding of the order data into the XML structure.

The **administrative order types**

Many administrative order types are described in detail in Chapters 8 and 9 (HIA/H3K: Chapter 4.4.1, HCA/HCS: Chapter 4.6.1). A complete list can be found in chapter 13.
Information on the support on the part of the bank (mandatory, optional, conditional) see also chapter 13.

3.11 Order parameters

The element `OrderParams` has been integrated into the fixed control data (under `ebicsRequest/header/static/OrderDetails`) for the transmission of order parameters that are not part of the order data. Depending on the administrative order type, this abstract element has a specific concrete characteristic:

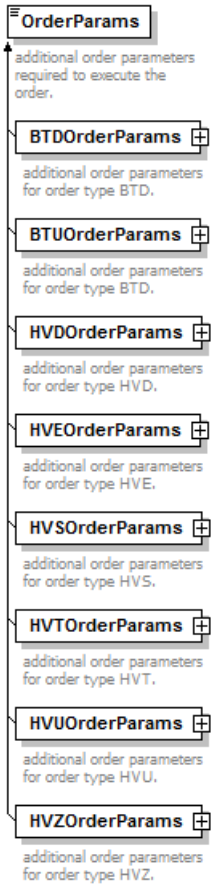


Diagram 6: OrderParams

- `BTDOrderParams` for the download of a business transaction format (BTF)
- `BTUOrderParams` for the upload of a business transaction format (BTF)
- `HVDOrderParams` in the case of order parameters for HVD
- `HVEOrderParams` in the case of order parameters for HVE
- `HVSOrderParams` in the case of order parameters for HVS
- `HVTOrderParams` in the case of order parameters for HVT
- `HVUOrderParams` in the case of order parameters for HVU
- `HVZOrderParams` in the case of order parameters for HVZ

The structures for the order parameters of BTD and BTU are explained in chapter 5. The structures of the order parameters for administrative order types HVD, HVE, HVS, HVT and HVU are explained in greater detail in Chapter 8.3.

3.12 Flow of the EBICS transactions

This chapter contains a simplified description of the protocol sequence for the transmission of a remote data transmission order via EBICS that allows for the stipulations in the previous chapter.

The transmission takes place in an EBICS transaction that can comprise several transaction steps. A transaction step is a pair, comprising an EBICS request and the corresponding EBICS response.

The first transaction step is the transaction initialisation step. Subscriber-related authorisation verifications are carried out in this step, such as e.g. the verification of authorisations for specific BTF identifiers. Successful authorisation verification is a prerequisite for continuation of the transaction. Furthermore, the ES's of the order are transmitted in this transaction step: in the case of upload orders, the ES's of the signatory are transmitted in the EBICS request; in the case of download orders, possibly the financial institution's bank-technical ES is transmitted in the EBICS response.

After transaction initialisation, a number of transaction steps usually follow in which the segments of order data are transmitted sequentially and in consecutive order.

Upload and download orders are always sent from the client to the server.

- In the customer to bank scenario the customer is always the client and the bank is always the server.
- In the communication between financial market infrastructures one party (which activates up- and downloads) is the client and the other party is the server.

Note: The specification is written in the style of a customer-bank usage. In the communication between financial market infrastructures it reads as follows:
The active party (submits upload and download requests), has the role of the customer and the other party that of the bank.

Upload orders that are sent to the bank system via EBICS can be authorised using two different methods:

Method 1: Authorisation by means of one or more bank-technical ES

The bank-technical ES's of an order file must be given by different subscribers. In case of the EDS, these subscribers may in special cases belong to different customers (customer-ID-spanning signature). The ES's can be submitted to the financial institution by different ways , while every ES submitted with a single EBICS transaction originates from the same customer.

1. Submission of the order data together with one or more ES's by way of an upload order with a present signature flag All ES's that are submitted in this manner originate from the customer of the party that submitted the order.
   If the transmitted ES's are not sufficient for the bank-technical approval
   a) the order is transferred to the EDS if the optional attribute @requestEDS is present. This means that the customer is able to add EDSs (electronic distributed signatures) via HVE .
   b) the order is rejected if the optional attribute @requestEDS is not present.

2. Submission of outstanding bank-technical ES's with the help of administrative order type **HVE**
   If an ES is submitted via an HVE transaction, this ES has to originate from the customer of the party that submitted the HVE transaction:
   HVE permits the special case of the customer-ID-spanning signature because the ES's submitted via HVE do not necessarily have to originate from the customer of the party that submits the orders.

   Method 2: Authorisation by means of an accompanying note signed by hand
   For the transmission of the order file the signature flag of the upload order is not present.
   Within the framework of the EBICS transaction, an ES of signature class "T" is transmitted together with the data of the order. The order is not passed on to the EDS but directly to the bank-specific post-processing.
   If the submitting subscriber possesses the authorisation for issuing a bank-technical ES in the bank system and signatures are submitted without the signature flag these signatures are strictly assessed only as transport signatures. The order must not be passed to the EDS either.

The meaning and admissible settings of the signature flag are described in chapter 3.14.

Transmission of an upload order with a (compressed, encoded and base64-coded) order data volume of 3 MB is represented by way of example with the help of the sequence diagram in Diagram 7. The EBICS transaction relating to an upload order is terminated as soon as the last order data segment has been successfully transmitted to the financial institution.

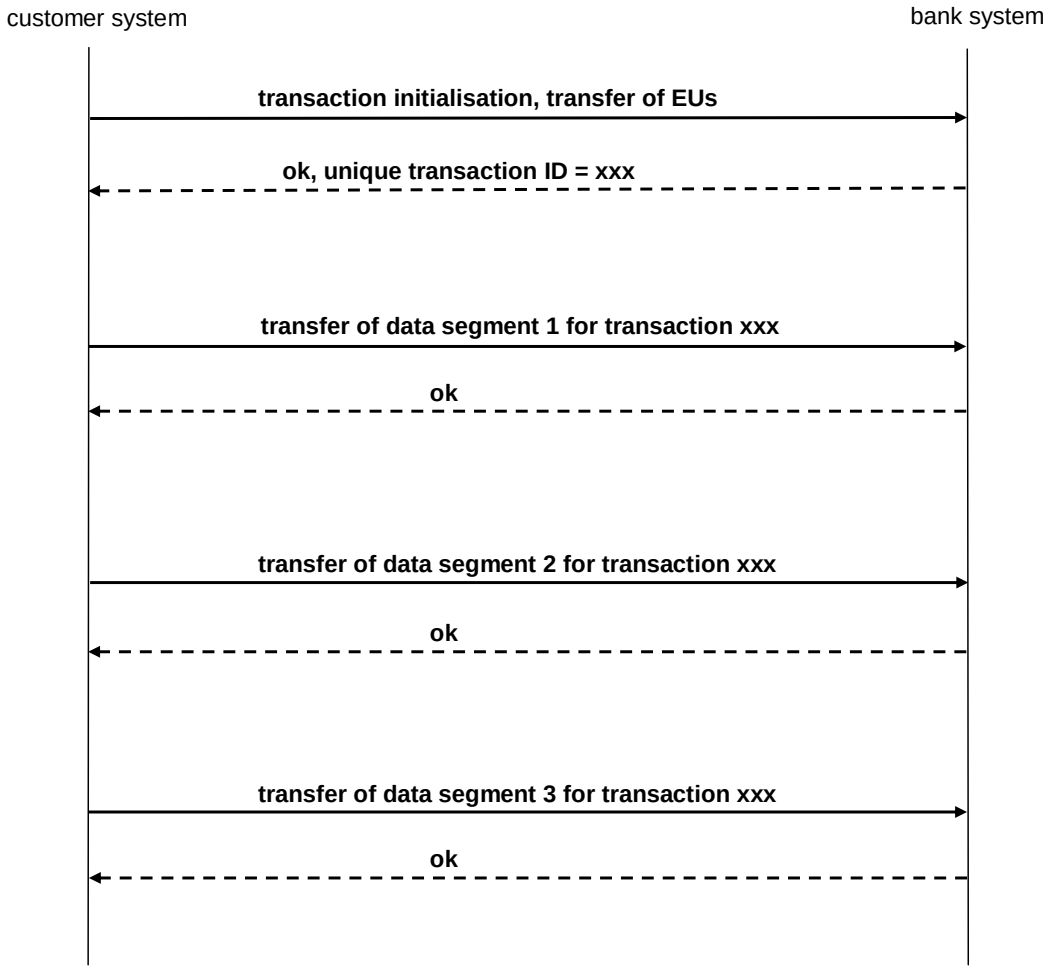


Diagram 7: Example of the sequence of an EBICS transaction for an upload order

Transmission of a download order with a (compressed, encoded and base64-coded) order data volume of 3 MB is represented by way of example with the help of the sequence diagram in Diagram 8. In the case of download orders, receipt of the download data is confirmed with an acknowledgement step. After this, the EBICS transaction for the download order is terminated.

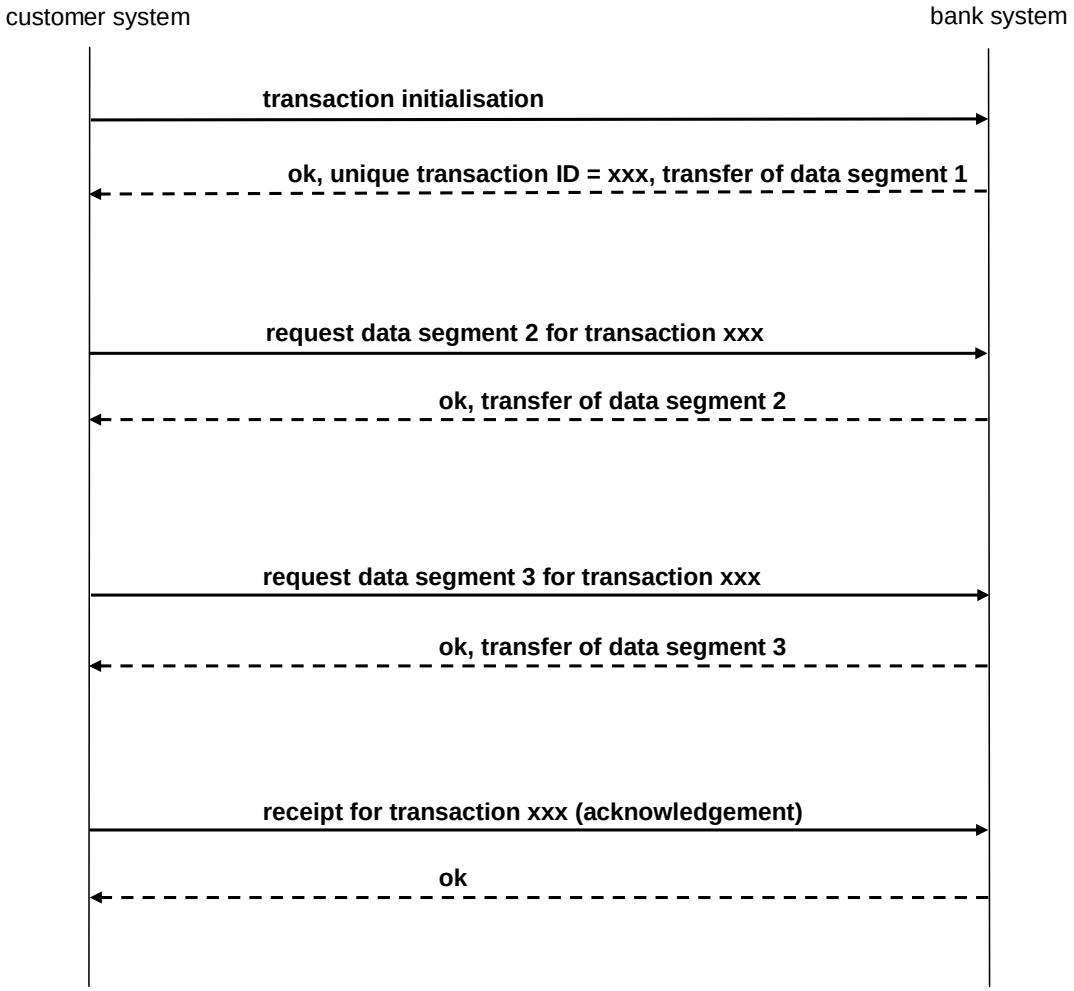


Diagram 8: Example of the sequence of an EBICS transaction for a download order

Further details on the sequence of EBICS transactions can be found in Chapter 5.

3.13 Interpretation of BTF element combinations by the bank server

This chapter describes how the BTF element groups service and service filter respectively which are delivered by EBICS requests are interpreted by the bank server:

Adminstrative EDS download orders:
Only delivered BTF elements in the filter are considered, missing (optional) elements are “wildcards”. The user will get all available matching orders, authorization of the user assumed.

Upload Requests:
The BTF elements must be complete and correct (corresponding to the uploaded case of business transaction). This means that also a missing (optional) element has a meaning!
But it depends on the bank (contract) how the authorizations are defined and stored:
It is independent of the EBICS communication, whether there is a model which stores all used/allowed single BTF combinations or hierarchical models where groups of business transactions are agreed (e.g. user is allowed to send SDD not dependent on service options and/or scopes)

Downloading files:
The BTF elements must be complete and correct (corresponding to the type of file which the user wants to download). This means that also a missing (optional) element has a meaning!

HKD / HTD:
Complete declaration of each BTF element combination the user/partner is authorized to in the HKD/HTD response. This means again that a missing elements has a meaning.

3.14 Interpretation of ES /EDS flag combinations by the bank server

This table describes the meaning of the flags for ES and electronic distributed signature (EDS):

ES flag	EDS Flag	Condition	Reaction	Matches with disestablished order attribute (for information only)
x	-	ES must be sufficient	Rejection, if ES not sufficient (“Authorization failed” 090003)	OZHNN
x	x	Customer has a contract for EDS (allowed to do HVE)	If sufficient number of valid ES, EDS Flag is ignored. Rejection, if no contract and number of ES not sufficient. (“EBICS Distributed Signature authorization failed” 091007)	OZHNN
-	-	File is not signed/authorized within EBICS	Depends on the contract: If signatures are needed and no signature outside EBICS is agreed the order is rejected (“Authorization failed” 090003)	DZHNN

4 Key management

4.1 Overview of the keys used

The EBICS protocol provides three RSA key pairs for each subscriber. These are used for the following purposes:

- bank-technical/technical ES of the order data that the subscriber/client system sends to the bank system
- identification and authentication of the subscriber by the bank system via identification and authentication signature
- decryption of the (symmetrical) transaction key used to encrypt the order data that the subscriber retrieves from the bank system.

Based on their use, one also talks of

- public / private bank-technical keys
- public / private identification and authentication keys
- public / private encryption keys

EBICS allows the use of three different key pairs per subscriber. In doing this, EBICS promotes the use of at least two different key pairs for each subscriber:

- One key pair is used exclusively for the bank-technical electronic signature.
- The use of one single key pair is allowed for identification and authentication of the subscriber by the bank system AND decryption of transaction keys.

Analogously to the subscriber keys, EBICS provides three different RSA key pairs for the bank system. These are used for the following purposes:

- bank-technical ES of the order data that is retrieved by a subscriber from the bank system
  In EBICS Version “H005” the financial institution’s bank-technical ES is only planned (see Chapter 3.5.2).
- identification and authentication of the financial institution by the subscriber via identification and authentication signature
- Decryption of the (symmetrical) transaction key to encrypt bank-technical order data sent by a subscriber to the financial institution.

The same restrictions as for the subscriber keys apply with regard to the use of an RSA key pair for different purposes.

The subscriber's keys are connected to processes that the subscriber would like to use for generation/verification of the ES, for generation/verification of the identification and authentication signature and for the encryption of order data. These processes are identified by unambiguous Versions so that different subscribers can use e.g. different processes for the ES. A prerequisite of EBICS is that the respective processes are administrated in the bank system for each subscriber.

Version "H005" of the EBICS protocol allows for the use of the following processes:

- "X002" for the identification and authentication signature
- "A005" or "A006" for the ES
- "E002" for the encryption.

Details of these processes can be found in the Appendix (Chapter 11.1, Chapter 11.2 and Chapter 11.3).

Subscribers of the same customer generally use the same processes for identification and authentication signature, encryption and ES.

A subscriber's orders can be delivered by a technical subscriber if both subscribers use the same processes for identification and authentication signature, encryption and bank-technical signature. In this case, administration of the public bank keys for all subscribers that wish to work with the same processes can be centralised at the customer's end.

4.2 Representation of the public keys

EBICS defines the administrative order types HIA, HCA, HCS and HPB, whose bank-technical order data constitutes public keys of the financial institution or the subscriber. For these order types, embedding of the public keys in EBICS messages takes place using newly-defined types based on the XML schema (see schema definition file ebics_types_H005.xsd). These types are contained in the following table:

Key type	XML type
Identification and authentication key	`AuthenticationPubKeyInfoType`
Bank-technical key	`SignaturePubKeyInfoType`
Encryption key	`EncryptionPubKeyInfoType`

The graphical representation of the XML types `AuthenticationPubKeyInfoType`, `SignaturePubKeyInfoType`, `EncryptionPubKeyInfoType` is contained in Diagram 9, Diagram 10, and finally Diagram 11.

The XML structures are composed in a similar manner to one another: They contain information relating to the usedX.509 certificate. Moreover, the version of the process for configuration/verification of the identification and authentication signature (see element `AuthenticationVersion`) is a component of the identification and authentication key. Analogously, the version of the encryption process is a component of the encryption key and the version of the bank-technical ES is a component of the bank-technical key.

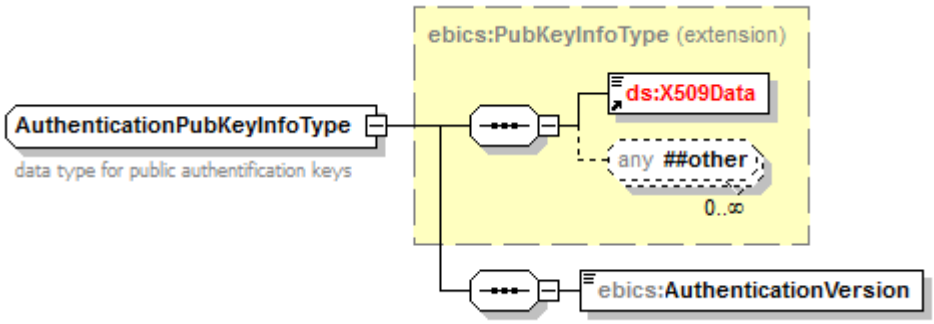


Diagram 9: Definition of the XML schema type AuthenticationPubKeyInfoType

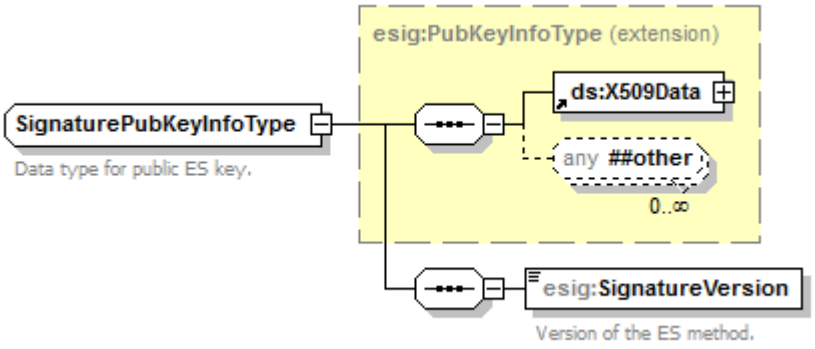


Diagram 10: Definition of the XML schema type SignaturePubKeyInfoType

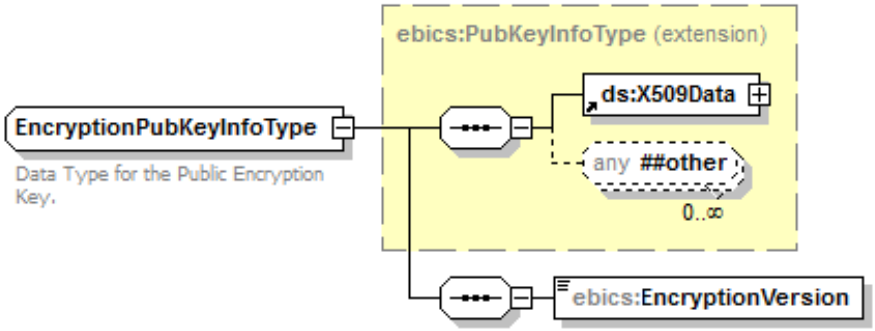


Diagram 11: Definition of the XML schema type EncryptionPubKeyInfoType

4.3 Actions within key management

Actual processing of the key management upload orders must take place synchronously to their transmission via EBICS. Hence processing must be completed before the final EBICS response of this transmission is sent to the subscriber.

This requirement applies in the case of INI and HIA as well as H3K (see Chapter 4.4.1) so that execution of subscriber initialisation is not delayed unnecessarily. It also applies equally in the case of SPR (see Chapter 4.5) so that the subscriber revocation is immediately activated. Subsequently-initialised EBICS transactions for the transmission of a bank-technical order are rejected at EBICS protocol level until the subscriber has again attained the state “Ready”. (Subscriber ES’s that have been successfully verified before the suspension also remain valid after the suspension. Such an ES can continue to be used for authorisation of an open order within the framework of the EDS).

Finally, this requirement also applies for all PUB, HCS, and HCA (see Chapter 4.6.1) to allow successful processing of immediately-following EBICS transactions from the relevant subscriber that already use the updated keys. (Subscriber ES’s that have been successfully verified before execution of PUB or HCS, respectively, also remain valid after the processing of PUB or HCS and the associated amendment of the bank-technical subscriber key. Such an ES can continue to be used for authorisation of an open order within the framework of the EDS).

4.4 Initialisation

A range of prerequisites must be fulfilled by the subscriber of a customer in order for them to be able to implement bank-technical EBICS transactions with a particular financial institution.

The basic prerequisite is the conclusion of a contract between customer and financial institution. In this contract it will be agreed as to which business transactions (BTF identifiers) the customer will conduct with the financial institution, which accounts are concerned, which of the customer’s subscribers work with the system and the authorisations that these subscribers will possess.

If the customer does not yet have access to a corresponding customer product, they will receive the client software and the financial institution’s access data (bank parameters) after conclusion of the contract. The financial institution will set up the customer and subscriber master data in the bank system in accordance with the contractual agreements. In doing this, the individual subscribers will receive the state “**New**”.

Details of the contractual agreements are not a subject of this standard, they are to be arranged individually between the customer and the financial institution.

Other prerequisites are successful subscriber initialisation and download of the financial institution's public keys by the subscriber. The necessary steps that must be taken by the financial institution, the customer and the subscriber and the chronological dependencies of these steps are contained in Diagram 12. Diagram 13 shows an example of a process by way of a sequence diagram. The state of the public bank keys at the subscriber's end is shown on the life-line of the subscriber system. Correspondingly, the state of the public subscriber keys at the bank's end and the state of the subscriber themselves are shown on the lifeline of the bank system. Details of these diagrams are explained in the following chapters.

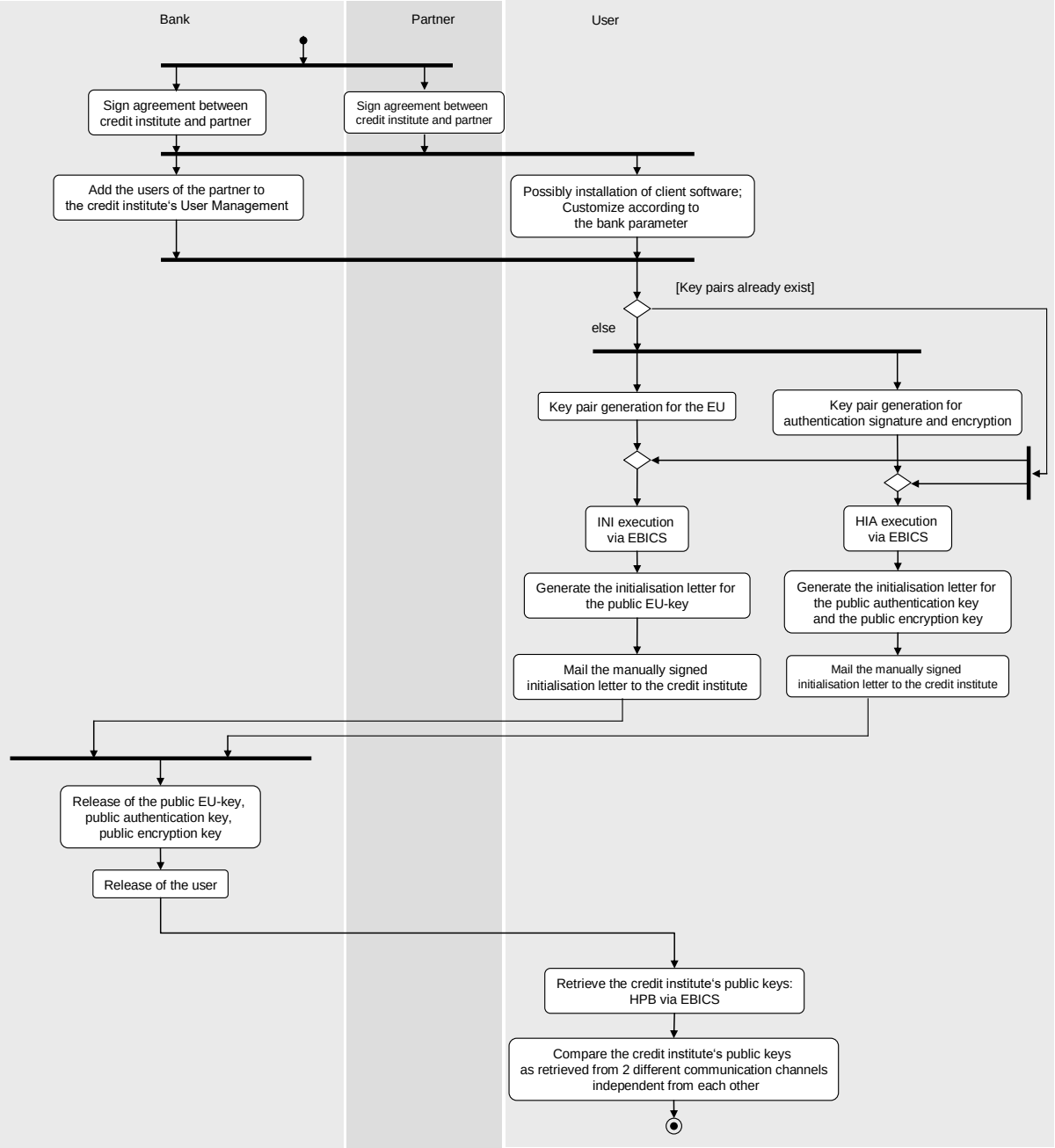


Diagram 12: Necessary steps prior to actual processing of business transactions via EBICS (using INI / HIA)

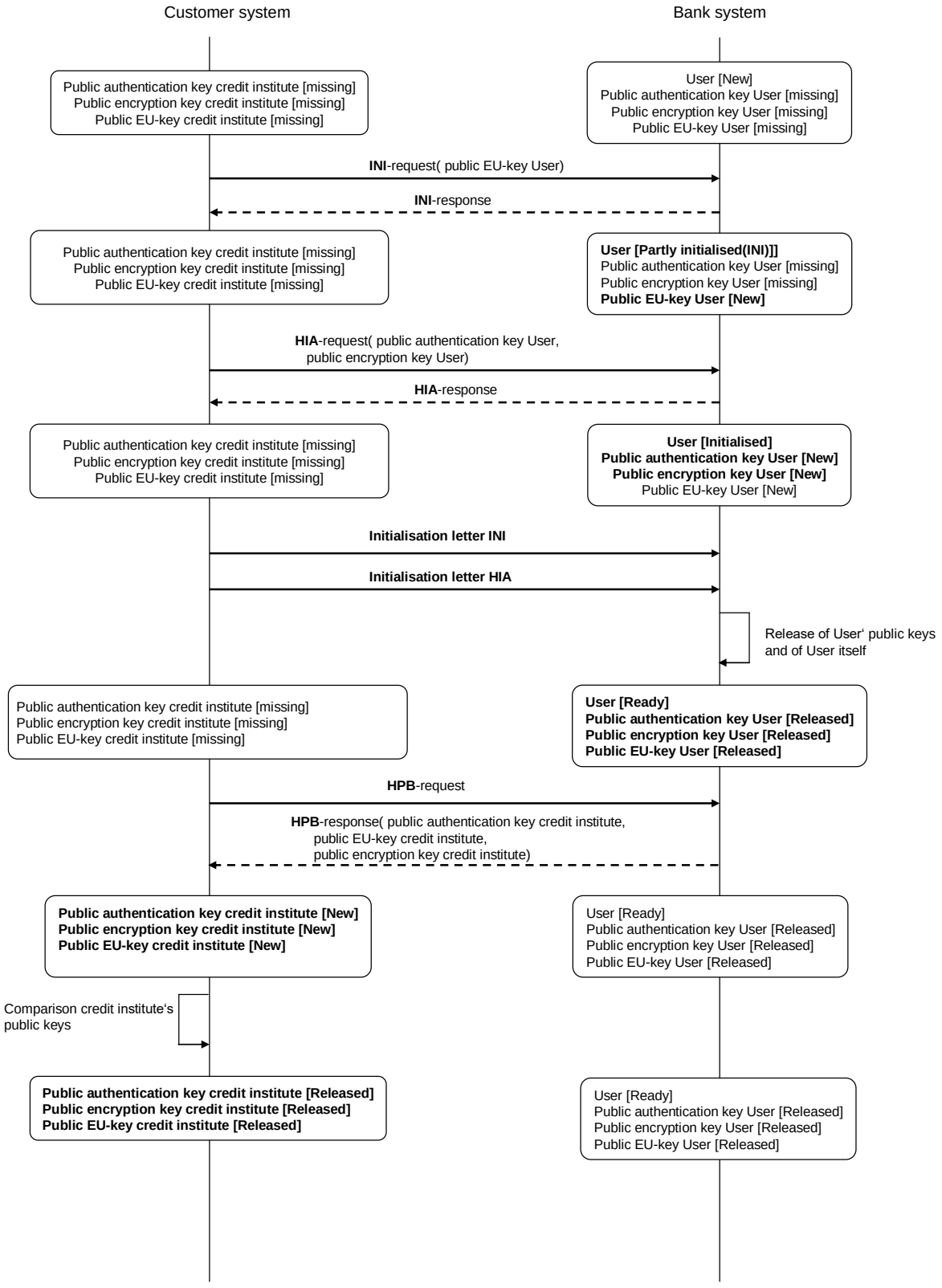


Diagram 13: Process example: Subscriber initialisation followed by download and verification of the bank keys (using INI / HIA)

4.4.1 Subscriber initialisation

4.4.1.1 General description

Transmission of the subscriber's public keys to the bank system is necessary for initialisation of the subscriber with the financial institution. The supported versions for the ES, the encryption and the identification and authentication signature are components of the bank parameters. The subscriber's bank-technical key must be newly generated if the subscriber does not have a suitable bank-technical key or does not wish to use an existing bank-technical key for the new bank connection. The same applies for the encryption key and the identification and authentication key.
The subscriber transmits his public keys to the financial institution as follows:

Alternative 1 (by two independent communication paths):

- via EBICS by means of the following administrative order types:
  - **INI**: send the public bank-technical key (key for the ES / authorisation key).
  - **HIA**: send the public identification and authentication key and the public encryption key.

  Transmission of the public subscriber keys to the financial institution via INI and HIA is referred to as **subscriber initialisation**

- by post with initialisation letters signed by the subscriber.

  The use of signed initialisation letters permits the financial institution to:

- verify the authenticity of the public subscriber's keys transmitted via EBICS as a prerequisite for the activation of subscribers

- guarantee the reproducibility of subscribers' key histories by storing the initialisation letters.

  The sequence for processing of INI and HIA is not fixed, but within the framework of subscriber initialisation precisely one INI order and precisely one HIA order will be implemented. Transmission of the public subscriber keys via two separate orders in any order requires definition of the subscriber states "**Partially initialised(INI)**" and "**Partially initialised(HIA)**". Within the framework of subscriber initialisation, the subscriber takes on the corresponding state depending on whether the first successful order is INI or HIA.

  This initialisation procedure (alternative 1) is always admissible: for RSA keys without as well as for keys with certificates issued by a CA. However, it is assumed that in each case initialisation letters are used for alternative 1. The public keys of the subscriber/user have still to be sent to the bank by the administrative order types INI (public bank-technical key) and HIA (public identification and authentication key as well as the public encryption key). However, in order to guarantee the authenticity of the subscriber's (user's) public keys, it must be ensured that the bank receives

the public bank keys via a second, independent communication path (initialisation letter for INI and for HIA, respectively). Having received the keys via different communication paths, the bank first compares the keys before approving them. Alternative 1 can also be used if alternative 2 fails.
For details and workflow concerning alternative 1, see chapter 4.4.1.2

Alternative 2 (in one step):
This alternative is only possible if the bank-technical (authorisation) key is based on a certificate issued by a CA, the authenticity of this particular public key is guaranteed by a CA as long as customer and bank have agreed to use this certain certificate and as long as it is valid. In this case, the initialisation letter can be resigned.

- via EBICS by means of the administrative order type:
  - **H3K**: send the public keys for the bank-technical signature (signature for authorisation; ES), and identification and authentication as well as encryption keys.

For the initialisation via CA-issued certificates, the administrative order type H3K simplifies the workflow:

1. All public keys can be sent in one step (H3K-request) using a certification issued by a CA for the bank-technical (ES) key.
2. INI and HIA letters are not necessary.

Details and workflow see chapter 4.4.1.3

4.4.1.2 Initialisation via INI and HIA

4.4.1.2.1 INI

Processing of INI is permissible if the state of the respective subscriber is “New”, “Suspended” or “Partially initialised(HIA)”. INI comprises a single EBICS request/response pair. The following applies for the EBICS request of INI:

- it does not require an identification and authentication signature since the subscriber’s public identification and authentication key has not yet been activated by the financial institution and hence cannot be used for verification.
- it does not contain a bank-technical signature, since the subscriber’s public bank-technical key is being transmitted for the first time in this request. This bank-technical key cannot be used by the financial institution to verification the bank-technical signature since its authenticity has not yet been ascertained.
- it contains the order data, i.e. the subscriber’s public bank-technical key in unencrypted form since the subscriber does not yet have the financial institution’s public encryption key (at least in the event of first initialisation).

The flow diagram in Diagram 14 represents the processing at the bank’s end that takes place on receipt of an INI request. Error situations that result from an invalid combination of customer/subscriber ID or an inadmissible subscriber state are not passed directly to the sender of the INI request. Instead, the sender receives the

technical error code
EBICS_INVALID_USER_OR_USER_STATE. INI does not give any errors of the type “Unknown subscriber” or “Inadmissible subscriber state” so that potential attackers are not given precise information about the validity of subscriber IDs or the state of subscribers. On the other hand, internal documentation must take place on the part of the financial institution to record the precise reason for the error.

The flow diagram provides verification of the subscriber state so that INI requests are rejected on the EBICS level if the subscriber state is inadmissible for INI. Admissible states for INI are: “New”, “Suspended” and “Partially initialised(HIA)”. Here, the state of the subscriber is verified from the header data of the request. The order data of the INI request (see Chapter 4.4.1.2.5.1) merely contains the subscriber whose bank-technical key is to be transmitted. For this reason, the subscriber from the header data should correspond with the subscriber from the order data. The EBICS protocol does not provide a verification for this correspondence. However, before actual processing of the order the state of the subscriber is verified (again) which is a part of the order data of INI.

Processing of an INI order can return the following error codes:

- EBICS_KEYMGMT_UNSUPPORTED_VERSION_SIGNATURE
  This business related error occurs when the order data contains an inadmissible version of the bank-technical signature process
- EBICS_KEYMGMT_KEYLENGTH_ERROR_SIGNATURE
  This business related error occurs when the order data contains a bank-technical key of inadmissible length
- EBICS_INVALID_ORDER_DATA_FORMAT
  This business related error occurs when the order data does not correspond with the designated format (see Chapter 4.4.1.2.5.1)
- EBICS_INVALID_USER_OR_USER_STATE
  This technical error occurs when the order data contains a subscriber that is either unknown or whose state is inadmissible for INI. The following subscriber states are admissible: New, Suspended, Partially initialised (HIA).
- For Return codes relating to CA-issued certificates, refer to Annex 1

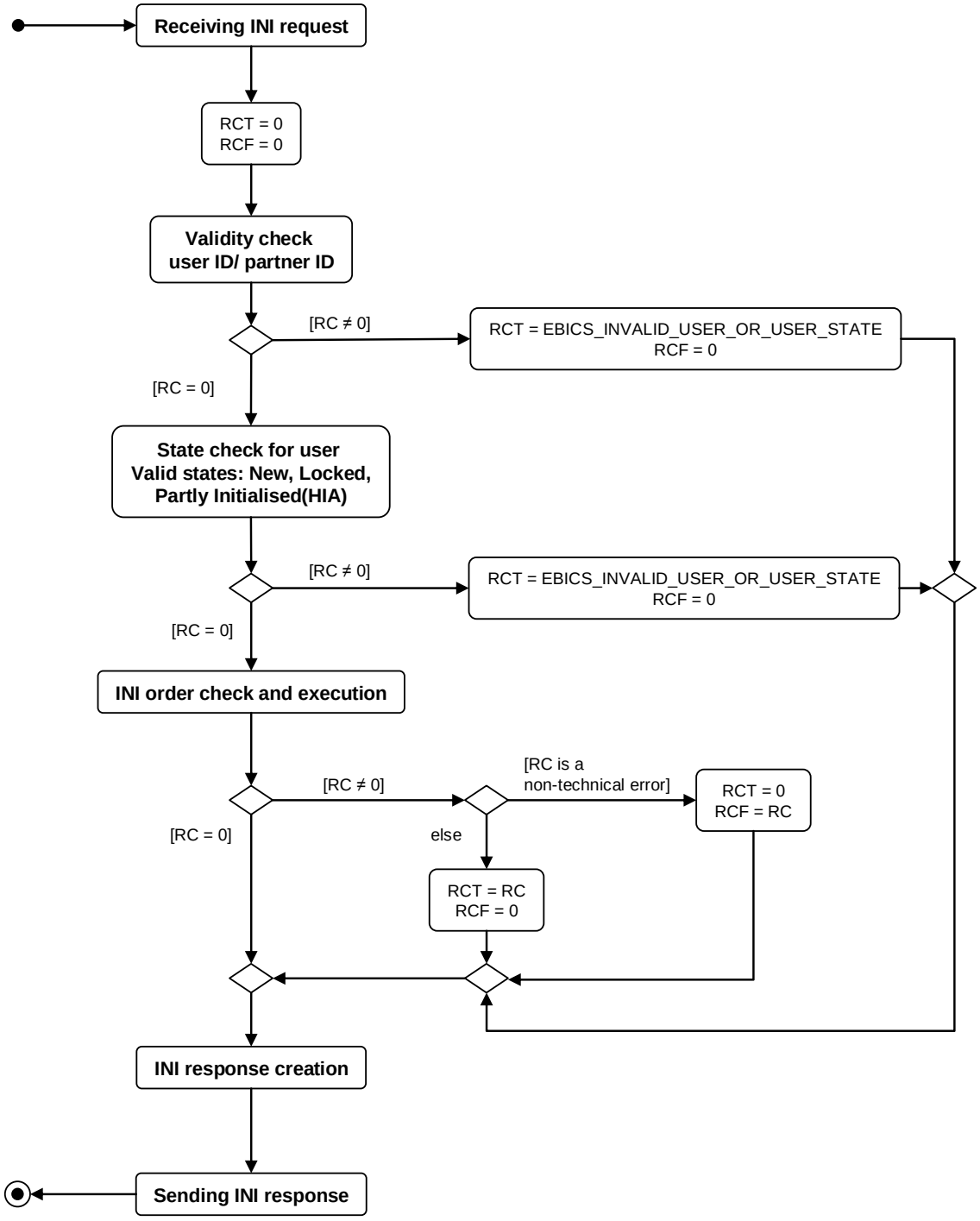


Diagram 14: Processing of an INI request at the bank's end

The EBICS response for INI does not contain an identification and authentication signature of the financial institution since the subscriber does not yet have the financial institution's public identification and authentication key with which they can carry out a verification.

In Diagram 13 INI is carried out before HIA, and correspondingly the subscriber changes from the state “New” to the state “Partially initialised(INI)”. The state “Partially initialised(INI)” means:

- The subscriber’s bank-technical key is available to the bank system, although it has not yet been activated
- The bank system does not (yet) have the subscriber’s public identification and authentication key or public encryption key.

In this state, the subscriber can only carry out one of the following two actions:

- Implement administrative order type HIA and then transfer into the state “Initialised”: Orders that are not equal to HIA that are submitted by the subscriber in this state are rejected by the bank system. Bank-technical signatures of the subscriber relating to existing orders are evaluated as invalid if the subscriber has the state “Partially initialised(INI)” at the time of verification.
- Have themselves suspended by the financial institution via telephone call: Following this, the only option is re-initialisation of the subscriber.

After the successful processing of INI, the subscriber sends a signed initialisation letter for INI. See Chapter 4.4.1.2.3 for details of the content of the initialisation letter.

4.4.1.2.2 HIA

Processing of HIA is permissible if the state of the subscriber is “New”, “Suspended” or “Partially initialised (INI)”. HIA comprises a single EBICS request/response pair. The following applies for the EBICS request of HIA:

- it does not contain an identification and authentication signature, since the subscriber’s public identification and authentication key is being sent for the first time in this request. This subscriber’s public identification and authentication key cannot be used by the financial institution to verify the identification and authentication signature since its authenticity has not yet been ascertained.
- it does not contain a bank-technical signature since the subscriber’s public bank-technical key has not yet been activated by the financial institution and hence cannot be used for verification.
- it contains the order data, i.e. the subscriber’s public encryption key and public identification and authentication key in unencrypted form since the subscriber does not yet have the financial institution’s public encryption key (at least on the event of first initialisation).

The flow diagram in Diagram 15 represents the processing at the bank’s end that takes place on receipt of an HIA request. In an analogous manner to INI, error situations that result from an invalid combination of customer/subscriber ID or an inadmissible subscriber state are also here not passed directly to the sender of the

HIA request. Instead, the sender receives the technical error code EBICS_INVALID_USER_OR_USER_STATE. HIA does not give any errors of the type “Unknown subscriber” or “Inadmissible subscriber state” so that potential attackers are not given precise information about the validity of subscriber IDs or the state of subscribers. Also analogously to INI, internal documentation must take place on the part of the financial institution to record the precise reason for the error. The flow diagram provides verification of the subscriber state so that HIA requests are rejected on the EBICS level if the subscriber state is inadmissible for HIA. Admissible states for HIA are: “New”, “Suspended” and “Partially initialised(INI)”. Here, the state of the subscriber is verified from the header data of the request. The order data of the HIA request (see Chapter 4.4.1.2.5.1) merely contains the subscriber whose identification and authentication key and encryption key are to be sent. For this reason, the subscriber from the header data should correspond with the subscriber from the order data. The EBICS protocol does not provide a verification for this correspondence. However, before actual processing of the order the state of the subscriber is verified (again) which is a part of the order data of HIA.

Processing of an HIA order can return the following error codes:

- EBICS_KEYMGMT_UNSUPPORTED_VERSION_ENCRYPTION
  This business related error occurs when the order data contains an inadmissible version of the encryption process
- EBICS_KEYMGMT_UNSUPPORTED_VERSION_AUTHENTICATION
  This business related error occurs when the order data contains an inadmissible version of the identification and authentication signature process
- EBICS_KEYMGMT_KEYLENGTH_ERROR_ENCRYPTION
  This business related error occurs when the order data contains an encryption key of inadmissible length
- EBICS_KEYMGMT_KEYLENGTH_ERROR_AUTHENTICATION
  This business related error occurs when the order data contains an identification and authentication key of inadmissible length
- EBICS_INVALID_ORDER_DATA_FORMAT
  This business related error occurs when the order data does not correspond with the designated format (see Chapter 4.4.1.2.5.1)
- EBICS_INVALID_USER_OR_USER_STATE
  This technical error occurs when the order data contains a subscriber that is either invalid or whose state is inadmissible for HIA. The following subscriber states are admissible: New, suspended, Partially initialised (INI).
- For Return codes relating to CA-issued certificates, refer to Annex 1

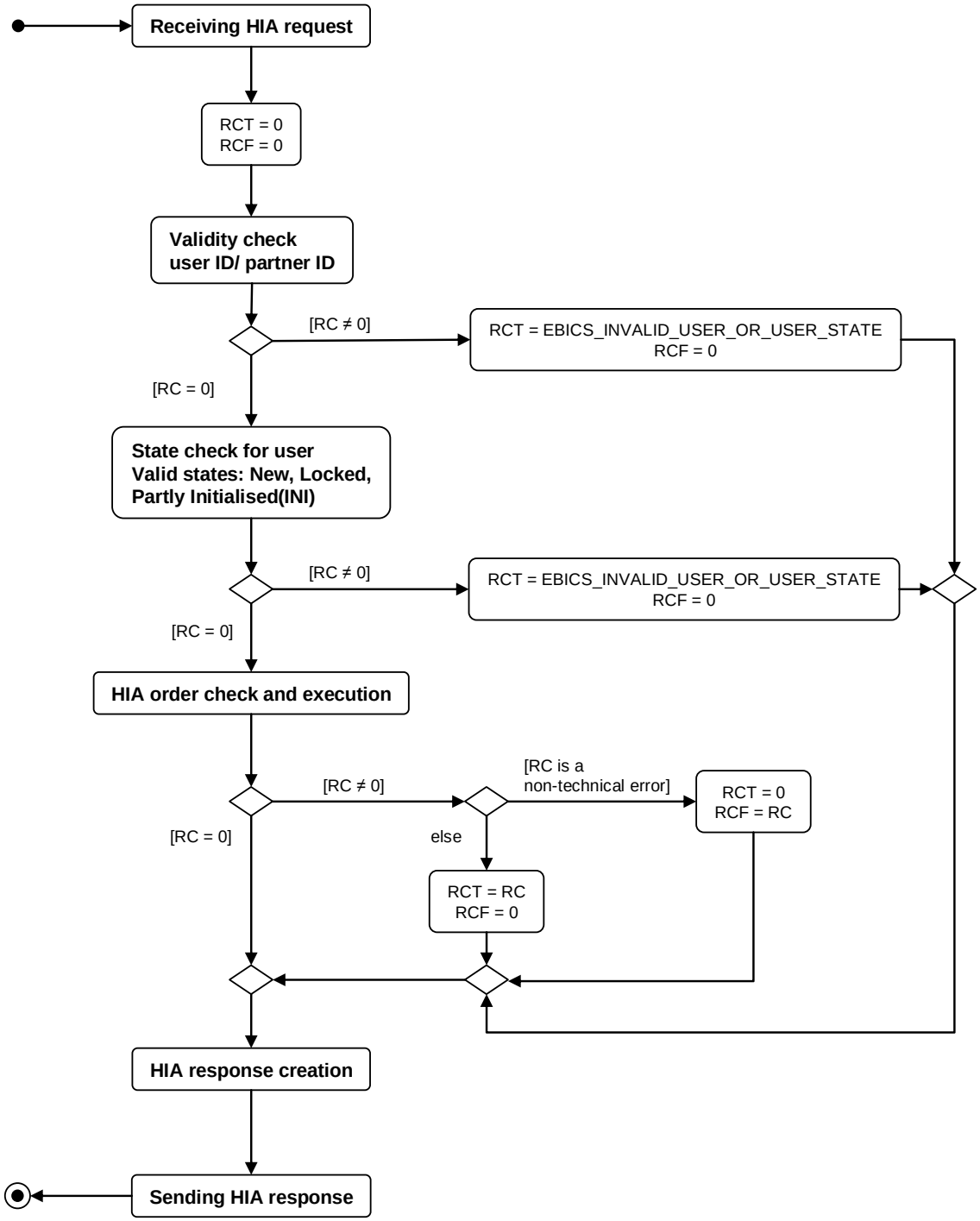


Diagram 15: Processing an HIA request at the bank's end

The EBICS response for HIA does not contain an identification and authentication signature of the financial institution since the subscriber does not yet have the financial institution's public identification and authentication key with which they can carry out a verification.

The meaning of the state “Partially initialised (HIA)“, that has not been taken into consideration in Diagram 13 is as follows:

- The bank system has the subscriber’s public identification and authentication key and public encryption key. Neither of these have been activated by the bank system
- The bank system does not (yet) have the subscriber’s public bank-technical key.

In this state, the subscriber can only carry out one of the following two actions:

- Carry out administrative order type INI:
  Orders that are not equal to INI that are submitted by the subscriber in this state are rejected by the bank system. Bank-technical signatures of the subscriber relating to existing orders are evaluated as invalid if the subscriber has the state “Partially initialised(HIA)” at the time of verification.
- Have themselves suspended by the financial institution via telephone call:
  Following this, the only option is re-initialisation of the subscriber.

After the successful processing of HIA, the subscriber sends a signed initialisation letter for HIA to the financial institution. See Chapter 4.4.1.2.3 for details of the content of the initialisation letter.

4.4.1.2.3 Initialisation letters

Initialisation letters for INI contain the public bank-technical subscriber certificate, initialisation letters for HIA contain the subscriber’s public identification and authentication certificate and the subscriber’s public encryption certificate. All certificates are presented in PEM format. In addition to the public subscriber certificates, the initialisation letters contain the following data:
:

- User name (optional): customer software-internal subscriber’s name
- Date: Date of processing of the corresponding EBICS order
- Time: Time of processing of the corresponding EBICS order
- Recipient bank
- Subscriber ID
- Customer ID.

In addition to the public subscriber certificate, the initialisation letter contains the following data:

- Purpose of the public subscriber certificate:

  - Bank-technical electronic signature

- Identification and authentication signature

- Encryption.

- Processes:

  - Bank-technical electronic signature process: A005 or A006

  - Identification and authentication signature process: X002

  - Encryption process: E002.

- Hash value of the public certificate in hexadecimal representation

  - The initialisation letter for INI contains the SHA-256 hash value of the certificate for the ES (in the case of A005 or A006, respectively). The composition of the hash value is described in chapter 14 for both processes.
  - The initialisation letter for HIA contains the SHA-256 hash value of the public identification and authentication certificate and the SHA-256 hash value of the public encryption certificate. The printed SHA-256 hash values of the certificate X002, E002 as well as of the A005 and A006 certificate are composed by calculating the SHA-256 hash value of the certificate in DER binary format, and presenting the resulting byte array (32 bytes) into hexadecimal representation (64 char) and in uppercase.

Initialisation letters for INI contain the public bank-technical subscriber certificate of the user, initialisation letters for HIA contain the subscriber's public identification and authentication certificate and the subscriber's public encryption certificate. Examples of initialisation letters can be found in the Appendix (Chapter 11.5.1).

4.4.1.2.4 Activation of the subscriber by the financial institution

After successful processing of INI and HIA, the subscriber is initially set to the state "**Initialised**": the bank system has all necessary public keys for the subscriber, but it will not have activated them yet. Subscribers that are set to the state "Initialised" cannot submit orders or signatures via EBICS: all attempts to do so will be rejected by the bank system.

After successful verification of the initialisation letters by the financial institution, the public subscriber keys are activated and the subscriber's state is set to "**Ready**" in the bank system. The state "Ready" means that the bank has all of the information necessary for the subscriber to successfully implement submission of orders or signatures. See also Diagram 13. The subscriber can also especially download the financial institution's so-called bank parameters via the administrative order type HPD (see Chapter 9.2).

Diagram 16 clarifies once again the state transitions of a subscriber as described above. Deletion of subscribers from bank systems' subscriber administration

databases is not covered in this standard. For this reason, a further state “Deleted” and an end state will not be displayed.

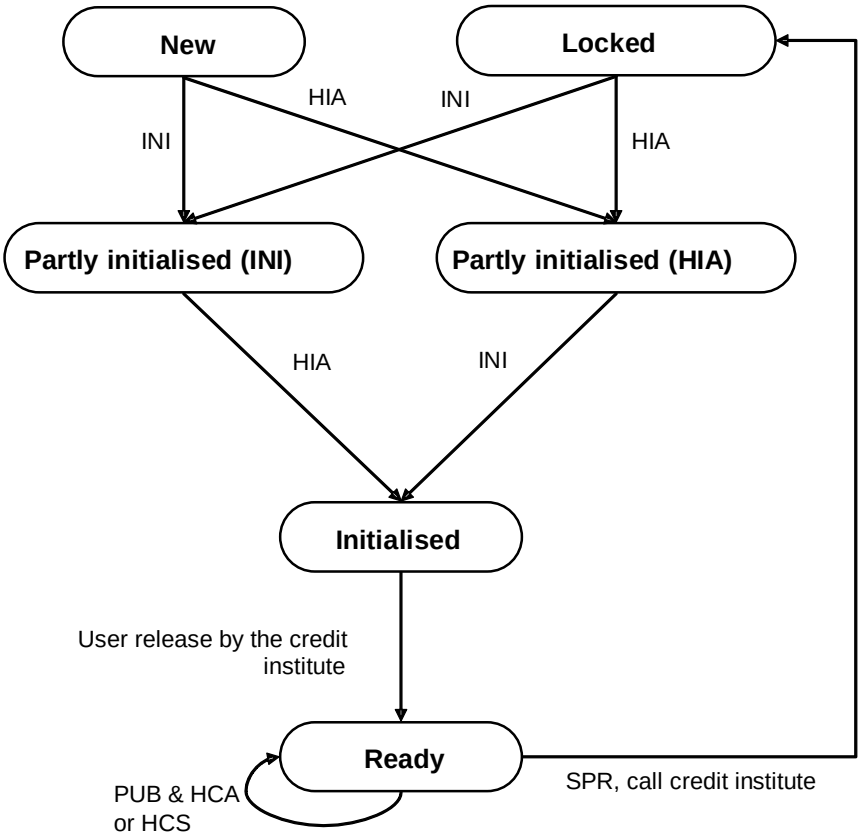


Diagram 16: State transition diagram for subscribers

The (renewed) processing of INI or HIA is not admissible in the subscriber state “Ready”. This is to prevent unintentional transfer of the subscriber from the state “Ready” to the state “Partially initialised(INI)” or “Partially initialised(HIA)”. The result of this would be that the affected subscriber would not be able to implement any further bank-technical orders for the time being.

Subscribers that are set to the state “Ready” must firstly suspend their remote access data transmission to the bank system before they can carry out renewed subscriber initialisation. Details on the suspension of subscribers can be found in Chapter 4.5.

4.4.1.2.5 Description of the EBICS messages

4.4.1.2.5.1 Format of the order data

The order data for INI is an instance document that conforms with ebics_signature_S002.xsd and comprises the top-level element `SignaturePubKeyOrderData`.

`SignaturePubKeyOrderData` is defined as follows via the XML schema:

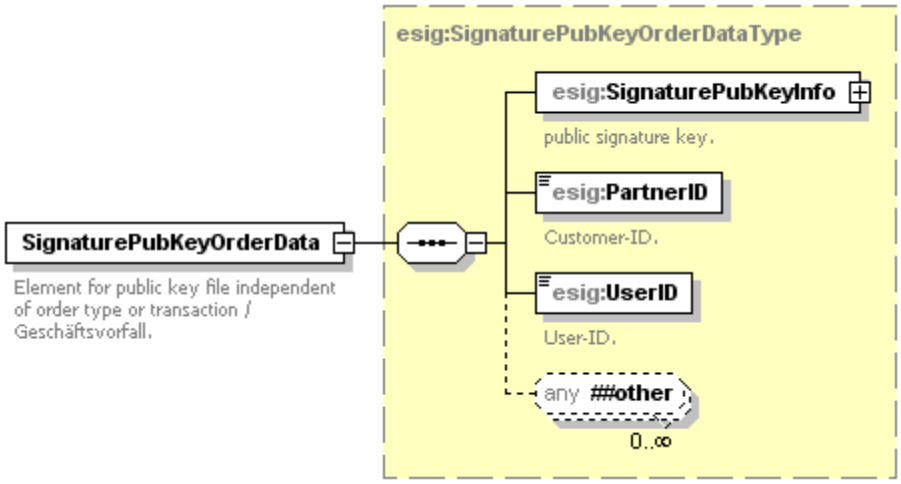


Diagram 17: Definition of the XML schema element SignaturePubKeyOrderData for INI order data (identical to PUB, see respective chapter)

The order data for HIA is an instance document that conforms with ebics_orders_H005.xsd and comprises the top-level element `HIARequestOrderData`. `HIARequestOrderData` is defined as follows via the XML schema:

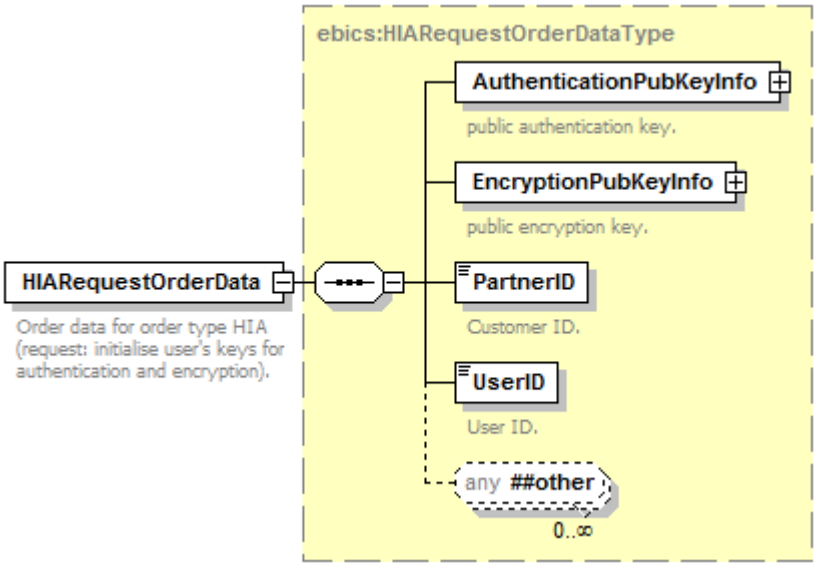


Diagram 18: Definition of the XML schema element HIARequestOrderData for HIA order data

The order data for INI and HIA are each compressed and base64-coded and embedded into the corresponding EBICS request.

4.4.1.2.5.2 Description and example messages

This chapter describes the EBICS messages for the administrative order types INI and HIA. INI and HIA requests are instance documents that conform with ebics_keymgmt_request_H005.xsd with the top-level element `ebicsUnsecuredRequest`. INI and HIA responses are instance documents that conform with ebics_keymgmt_response_H005.xsd with the top-level element `ebicsKeyManagementResponse`.

The data that is a component of these messages is listed here. The corresponding XML elements are given in brackets in XPath notation. The following conventions apply:

- Data that is fundamentally optional is marked “(optional)”.
- Data that may only be missing under certain conditions is instead marked “(conditional)”.
- Optional XML elements of the EBICS messages that are missing in the description may not appear in the EBICS message.
- Optional XML elements in the EBICS messages that appear in the description without the designation “(optional)” or “(conditional)” must always be placed in accordance with the description.

This description is supplemented by examples.

- Transmission of the following data in the INI request (see example in Diagram 19)
  Host ID of the EBCIS bank computer system (`ebicsUnsecuredRequest/header/static/HostId`)
  Subscribers (`ebicsUnsecuredRequest/header/static/PartnerID`, `ebicsUnsecuredRequest/header/static/UserID`) whose public bank-technical key is to be sent to the financial institution
  (Optional) technical subscribers (`ebicsUnsecuredRequest/header/static/PartnerID`, `ebicsUnsecuredRequest/header/static/SystemID`)
  `SystemID` can be contained in the message if the customer system is a multi-user system. Since INI requests do not contain an identification and authentication signature and the order data is unencrypted, declaration of the SystemID is optional.
  - (Optional) information on the customer product (`ebicsUnsecuredRequest/header/static/Product`)
  - Administrative Order type (ebics`UnsecuredRequest`/header/static/OrderDetails/AdminOrderType) set to “INI”
  - Security medium for the subscriber’s bank-technical key (`ebicsUnsecuredRequest/header/static/SecurityMedium`)

The admissible settings are listed in the Appendix (Chapter 12.3)
- Order data (`ebicsUnsecuredRequest/body/DataTransfer/OrderData`).

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsUnsecuredRequest
 xmlns="urn:org:ebics:H005" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_keymgmt_request_H005.xsd"
 Version="H005" Revision="1">
 <header authenticate="true">
   <static>
   <HostID>EBIXHOST</HostID>
     <PartnerID>CUSTM001</PartnerID>
     <UserID>USR100</UserID>
     <OrderDetails>
       <AdminOrderType>INI</AdminOrderType>
     </OrderDetails>
     <SecurityMedium>0200</SecurityMedium>
   </static>
   <mutable/>
 </header>
 <body>
   <DataTransfer>
     <!--INI file compressed and base64 encoded -->
     <OrderData>
       …
     </OrderData>
   </DataTransfer>
 </body>
</ebicsUnsecuredRequest>
```

Diagram 19: EBICS request for administrative order type INI

- **Transmission of the following data in the INI response** (see example in Diagram 20)

- Bank-technical return code
  (`ebicsKeyManagementResponse/body/ReturnCode`)
- Order number (`ebicsKeyManagementResponse`/header/mutable/OrderID)

This number is assigned by the bank server automatically.

- Technical return code
  (`ebicsKeyManagementResponse`/header/mutable/ReturnCode)
- Technical report text
  (`ebicsKeyManagementResponse`/header/mutable/ReportText)
- (Optional) time stamp for the last updating of the bank parameters
  (`ebicsKeyManagementResponse/body/TimestampBankParameter`).

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsKeyManagementResponse
 xmlns="urn:org:ebics:H005"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_keymgmt_response_H005.xsd"
 Version="H005" Revision="1">
 <header authenticate="true">
```

```
    <static/>
    <mutable>
      <OrderID>A101</OrderID>
      <ReturnCode>000000</ReturnCode>
      <ReportText>[EBICS_OK] OK</ReportText>
    </mutable>
  </header>
  <body>
    <ReturnCode authenticate="true">000000</ReturnCode>
  </body>
</ebicsKeyManagementResponse>
```

Diagram 20: EBICS response for administrative order type INI

- **Transmission of the following data in the HIA request** (analogous to INI, see example in Diagram 21)

  - Host ID of the EBICS bank computer system (`ebicsUnsecuredRequest/header/static/HostId`)
  - Subscribers (`ebicsUnsecuredRequest/header/static/PartnerID`, `ebicsUnsecuredRequest/header/static/UserID`) whose public identification and authentication key as well as public encryption key are to be sent to the financial institution
  - (Optional) technical subscribers (`ebicsUnsecuredRequest/header/static/PartnerID`, `ebicsUnsecuredRequest/header/static/SystemID`)
    `SystemID` can be contained in the message if the customer system is a multi-user system. Since HIA requests do not contain an identification and authentication signature and the order data is unencrypted, declaration of `SystemID` is optional.
  - (Optional) information on the customer product (`ebicsUnsecuredRequest/header/static/Product`)
  - Administrative Order type(`ebicsUnsecuredRequest/header/static/OrderDetails/AdminOrderType`) set to “HIA”
  - Security medium for the subscriber’s bank-technical key (`ebicsUnsecuredRequest/header/static/SecurityMedium`) set to “0000”.
    The security medium for the subscriber’s bank-technical key is set to “0000” since HIA orders neither transmit bank-technical keys nor contain ES’s.
  - Order data (`ebicsUnsecuredRequest/body/DataTransfer/OrderData`).

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsUnsecuredRequest
 xmlns=" urn:org:ebics:H005"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_keymgmt_request_H005.xsd"
 Version="H005" Revision="1">
 <header authenticate="true">
```

```
    <static>
      <HostID>EBIXHOST</HostID>
      <PartnerID>CUSTM001</PartnerID>
      <UserID>USR100</UserID>
      <OrderDetails>
        <AdminOrderType>HIA</AdminOrderType>
             </OrderDetails>
      <SecurityMedium>0000</SecurityMedium>
    </static>
    <mutable/>
  </header>
  <body>
    <DataTransfer>
      <!-- XML instance document using root element HIARequestOrderData in accordance
with ebics_orders_H005.xsd, compressed and base64 encoded -->
      <OrderData>
        …
      </OrderData>
    </DataTransfer>
  </body>
</ebicsUnsecuredRequest>
```

Diagram 21: EBICS request for administrative order type HIA

- **Transmission of the following data in the HIA response** (analogous to INI, see example Diagram 22):

- Bank-technical return code
  (`ebicsKeyManagementResponse/body/ReturnCode`)
- Order number (`ebicsKeyManagementResponse/header/mutable/OrderID`)

This number is assigned by the bank server automatically.

- Technical return code
  (`ebicsKeyManagementResponse/header/mutable/ReturnCode`)
- Technical report text
  (`ebicsKeyManagementResponse/header/mutable/ReportText`)
- (Optional) time stamp for the last updating of the bank parameters
  (`ebicsKeyManagementResponse/body/TimestampBankParameter`).

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsKeyManagementResponse
 xmlns="urn:org:ebics:H005"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_keymgmt_response_H005.xsd"
 Version="H005" Revision="1">
 <header authenticate="true">
   <static/>
   <mutable>
     <OrderID>A101</OrderID>
     <ReturnCode>000000</ReturnCode>
     <ReportText>[EBICS_OK] OK</ReportText>
   </mutable>
 </header>
 <body>
   <ReturnCode authenticate="true">000000</ReturnCode>
 </body>
```

```
</ebicsKeyManagementResponse>
```

Diagram 22: EBICS response for administrative order type HIA

4.4.1.3 Initialisation via H3K

If the bank-technical (authorisation) key is based on a self-signed certificate or an RSA key without certificate, the public keys of the subscriber/user are still sent to the bank by the administrative order types INI and HIA.
In order to guarantee the authenticity of the subscriber's (user's) public keys, it must be ensured that the bank receives the public bank keys via a second, independent communication path (initialisation letter for INI and for HIA, respectively). The bank first compares the keys having received via different communication paths before approving them.

When using CA-issued certificates the process for H3K is now the following:

1) The certificate for the bank-technical signature (ES) must be issued by a CA. In this case, a letter is not necessary for the initialisation of this key. However, the certificates for encryption as well as identification and authentication can be self-signed certificates as well as certificates issued by a CA.
2) As to the upload of the public keys for encryption and authentication, these can be signed by an ES. A letter for the initialisation of these keys is not necessary.
3) The necessary checks on the bank's side (before applying the keys for the first time) are:
   a. Does an agreement for the use of the CA-issued certificate exist?
   b. Are the administration steps for the customer/user finalised at the EBICS server and is the user known at the EBICS server?
   c. Is the certificate valid?
4) Taking into consideration 1) to 3), a new order type H3K can be defined:
   a. It combines INI and HIA (transport of three public keys, all keys base on certificates); for the ES key, the certificate must be issued by a CA.
   b. It already contains an ES (via certificate issued by a CA)

This concept requires a high level of authenticity for the certificates used in this H3K process (which serve as ES keys) and also for the Certification Authority (CA) which issues these certificates:

- Issuing of the certificate:
  - Strong identification requirements for the identification (regarding the person and the organisation requesting the certificate)
  - All data in the certificate have been thoroughly validated by a registration authorithy.
- Naming rules:
  - For the name in the certificate (SubjectName) there must be a fix schema, which allows a unique (automatic) assignment of the natural person to the company.
- Cryptographic requirements:

- E.g. length of the keys
- Validation requirements:
- Actuality and availability of revocation list

For the electronic initialisation (without INI letter) of new EBICS users with the administrative order type H3K the usage of CA-issued certificates is mandatory.
When the certificate is not issued by the bank itself, a necessary prerequisite for the electronic initialisation is that the new EBICS user has notified his certificate for the ES including his unique subject name/CA name and the relevant CA root certificate to his bank. The new EBICS user must use his certificate for the ES to notify the certificates for authentication and encryption to his bank.
The requirements that have to be fulfilled in the certificate policy are agreed bilaterally between customer and bank. The interoperability of different trust domains can be achieved only, if appropriate technical, organisational and legal requirements are defined. These requirements are not addressed in the EBICS specification as they are not relevant for the communication standard itself.

In the schema file ebics_orders_H005.xsd, the element group `H3KRequestOrderData` contains three certificates. In the schema file ebics_keymgmt_request_H005 the structure ebics:UnsignedRequest contains `H3KRequestOrderData` (compressed/base-64 encoded) and a signature.
In the schema file ebics_ types_H005.xsd the type (needed for sending a X509 Certificate) `SignatureCertificateInfoType` is defined. `AuthenticationCertificateInfoType` and `EncryptionInfoType` are extensions of this type and each of these types is used in `H3KOrderData`.

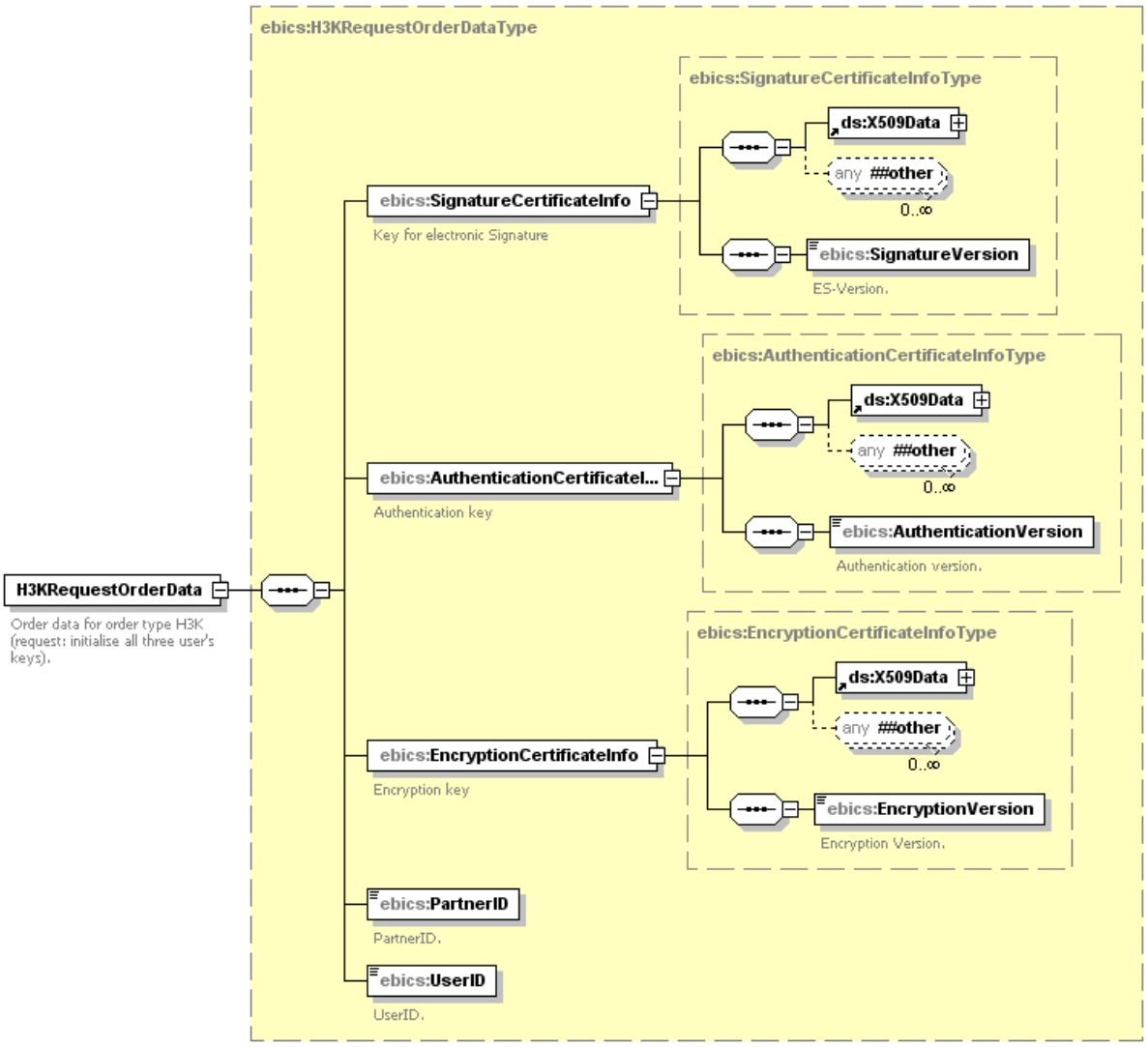


Diagram 23: Definition of the XML schema element H3KRequestOrderData for H3K order data

When initialising a user, the check process on the bank's side is as follows:

1- Check the structure of `H3KRequestOrderData` and extract the certificate for authorisation (ES).

2- Check if this certificate was delivered by a valid Certificate Authority (CA).

3- Check if the signature in SignatureData corresponds to the public ES key in `H3KRequestOrderData`.

4- Check if the information in the certificate (`ebicsUnsignedRequest` → `body` → `DataTransfer`) matches the previously declared information on the client.
Note: The Bank is free to choose the means and the kind of information necessary for the match. For example, it can be declared in a contract or by a process of uploading certificates published on the bank's web site.

a. If it matches, the server returns EBICS_OK (code 000000). The state of the user is automatically switched to '**Ready**'. The user doesn't need to send initialisation letters (or rather no other process of validation is necessary)

b. If the server is unable to match the certificate (ES key) with the previously declared information, the server sends a reject response and the H3KRequest is aborted. The server returns EBICS_CERTIFICATES_VALIDATION_ERROR (code 091219)
The state of the user remains the same ("**New**"). The user has two possibilities to go on:

   i. Process a H3K request again with a correct certificate (for the ES) issued by a CA

   ii. Or process INI and HIA for initialisation
(INI/HIA is especially usable as a backup process)

The signing certificate (ES) must be valid (and, above all, not expired). Otherwise the customer has to update/ declare the new certificate (issued by a CA).

Furthermore, all error codes related to the key management can be returned in the H3K process.

Note:
A bank can act as a "private" CA for its customers, i.e. it creates keys that meet all the conditions and properties of a CA certificate and is valid as a CA certificate at this bank.
The following rule applies to this type of certificate:
- At the bank issuing the "private" CA certificate, an EBICS initialisation can take place via H3K according to the process described above (i.e. INI/HIA and INI letter are not required)
- If the EBICS user wants to use these keys at another bank, however, these certificates are not considered CA certificates and therefore a standard initialisation must take place there via INI/HIA and INI letter.

4.4.2 Download of the financial institution's public keys

4.4.2.1 General description

The subscriber downloads all public keys from the financial institution by means of a specially-provided administrative order type (HPB). Download of the public bank keys necessitates the subscriber state "Ready", only then can the processes be established that the subscriber wishes to implement for the identification and authentication signature, bank-technical signature and encryption.

Processing of HPB merely requires a single EBICS request / response pair. The EBICS request of HPB contains the subscriber's identification and authentication signature itself, or that belonging to a technical subscriber of the same customer, via the control data.

Diagram 24 represents the processing at the bank's end that takes place on receipt of an HPB request. The replay test takes place in the same way as with the bank-technical order types (see Chapter 11.4). Thus HPB returns the technical error EBICS_TX_MESSAGE_REPLAY when the HPB request is a recovered message. In the same way, in the case of bank-technical order types, verification of the customer/subscriber ID, the subscriber state and the identification and authentication signature takes place within the process step "Verifying authenticity of the EBICS request" (see Chapter 5.5.1.2.1, Diagram 41).

This can produce the following error codes:

- EBICS_AUTHENTICATION_FAILED
  This technical error occurs when the subscriber's (or the technical subscriber's) identification and authentication signature could not be verified successfully
- EBICS_USER_UNKNOWN
  This technical error occurs when the technical user's identification and authentication signature could be successfully verified but the (non-technical) subscriber is unknown to the financial institution
- EBICS_INVALID_USER_STATE
  This technical error occurs when the technical user's identification and authentication signature could be successfully verified and the (non-technical) subscriber is known to the financial institution but does not have the state "Ready".

  After successful processing of the "Message authenticity verification", the actual processing of the HPB order does not produce any further specific technical or bank-technical errors.

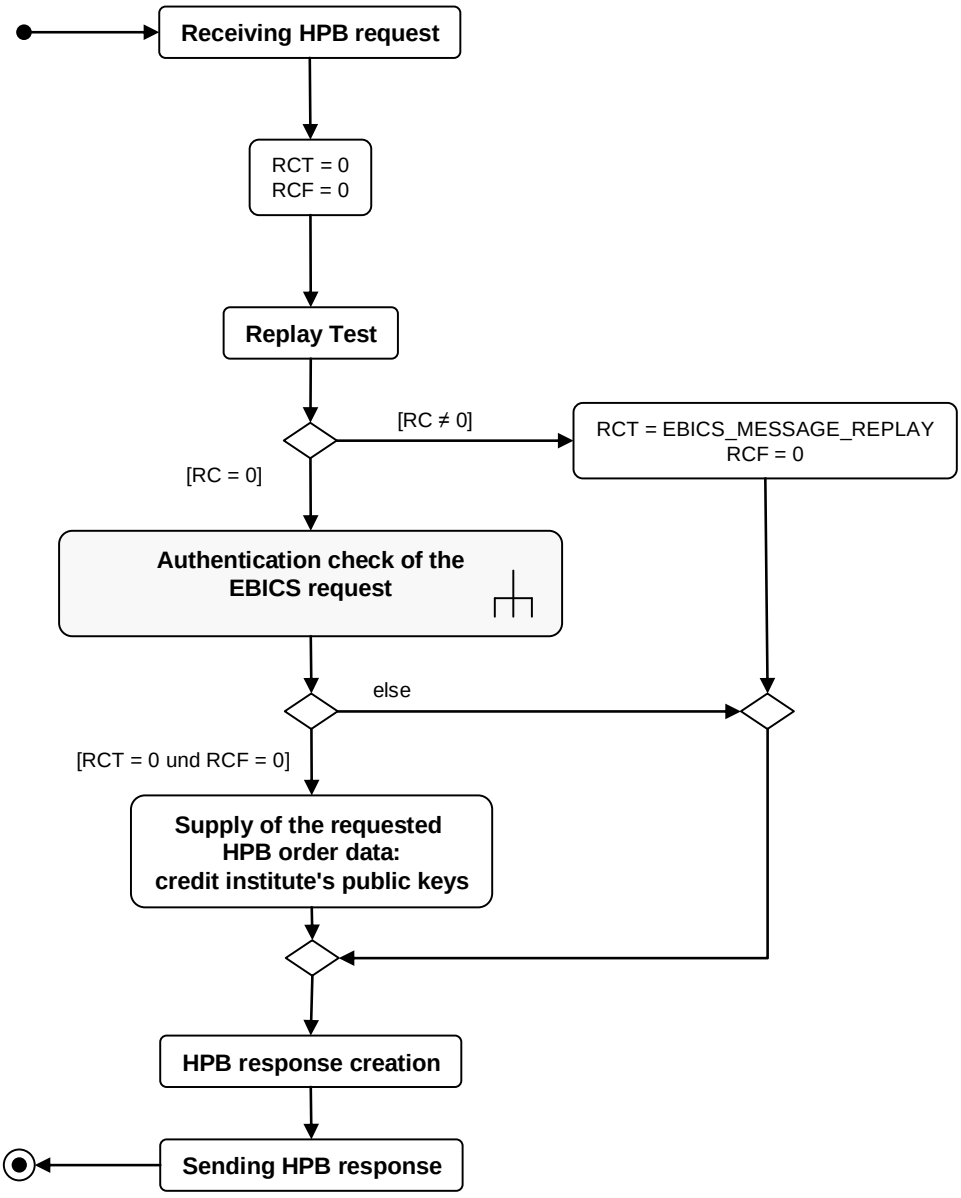


Diagram 24: Processing of an HPB request at the bank's end

The following applies in the case of the EBICS response:

- it does not contain an identification and authentication signature, since the financial institution's public identification and authentication key is being transmitted for the first time in this response. This financial institution's public identification and authentication key cannot be used by the subscriber to verify the identification and authentication signature since its authenticity has not yet been ascertained.
- it does not contain a bank-technical electronic order data signature, i.e. the public bank key, since the financial institution's bank-technical key is being transmitted for the first time in this response. This public bank-technical key cannot be used by the subscriber to verify the bank-technical ES since its authenticity has not yet been ascertained.
- it contains the order data, i.e. the public bank key, in encrypted format since the subscriber's public encryption key has already been activated by the financial institution.

The subscriber has all necessary public bank keys after successful processing of HPB, although they must verify them before they are used: as shown in Diagram 13 the state of these keys at the subscriber's end is set to "New". When they are set to the state "New", bank keys may not be used for communication via EBICS since their authenticity is not ensured.

In order to guarantee the authenticity of the bank keys, the financial institution must ensure that the subscriber receives the public bank keys and/or the hash-values via a second, independent, communication channel (e.g. via the financial institution's website). The subscriber is responsible for verification of the bank keys. The process for verification of the bank keys is not a part of this standard. If the bank provides certificates issued by a CA, the client has to check the validity of the certificates (for annotation, please refer to the Common Implementation Guide). It is dependent on the implementation of the EBICS client systems that ensure that subscribers only use the public keys after they have been successfully verified.

After successful verification, the state of the public bank keys at the subscriber's end changes from "New" to "Activated". This state change is also shown in Diagram 13 Only those bank keys that have the state "Activated" may be used for communication via EBICS.

In EBICS Version "H005" the ES of the financial institutions is only planned (see Chapter 3.5.2). Diagram 13 takes into account the state of the bank's public bank-technical key at the subscriber's end in preparation for future EBICS versions.

4.4.2.2 Description of the EBICS messages

4.4.2.2.1 Format of the order data

The order data for HPB is an instance document that conforms with ebics_orders_H005.xsd and comprises the top-level element `HPBResponseOrderData`. `HPBResponseOrderData` is defined as follows via the XML schema:

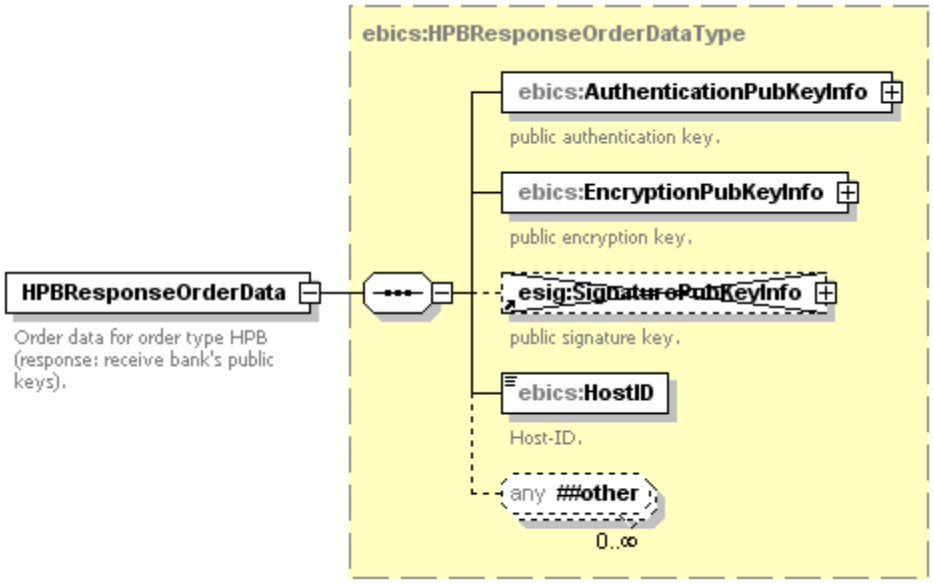


Diagram 25: Definition of the XML schema element HPBRequestOrderData for HPB order data

In Version “H005” of the EBICS protocol the ES of the financial institutions is only planned (see Chapter 3.5.2). The element `SignaturePubKeyInfo` is defined in preparation for future versions with maximum frequency (maxOccurs) being equal to 0.

The order data is compressed, encrypted and base64-coded, and embedded into the corresponding HPB response.

4.4.2.2.2 Description and example messages

This chapter describes the EBICS messages for the administrative order type HPB. HPB requests are instance documents that conform with ebics_keymgmt_request_H005.xsd with the top-level element `ebicsNoPubKeyDigestsRequest`. On the other hand, HPB responses are instance documents that conform with ebics_keymgmt_response_H005.xsd with the top-level element `ebicsKeyManagementResponse`.

The data that is a component of these messages is listed here. The corresponding XML elements are given in brackets in XPath notation. The following conventions apply:

- Data that is fundamentally optional is marked “(optional)”.
- Data that may only be missing under certain conditions is instead marked “(conditional)”
- Optional XML elements of the EBICS message that are missing in the description may not be present in the EBICS message
- Optional XML elements in the EBICS messages that appear in the description without the designation “(optional)” or “(conditional)” must always be placed in accordance with the description.

This description is supplemented with an example of an EBICS request / response pair for the administrative order type HPB.

- **Transmission of the following data in the HPB request** (see example in Diagram 26):
  - Host ID of the EBCIS bank computer system (`ebicsNoPubKeyDigestsRequest/header/static/HostId`)
  - Combination of Nonce and Timestamp, necessary to avoid replaying old EBICS messages (`ebicsNoPubKeyDigestsRequest/header/static/Nonce`, `ebics/header/static/Timestamp`)
  - Subscribers (`ebicsNoPubKeyDigestsRequest/header/static/PartnerID`, `ebics/header/static/UserID`) who initiates the HPB request
  - (Conditional) technical subscribers (`ebicsNoPubKeyDigestsRequest/header/static/PartnerID`, `ebics/header/static/SystemID`)
    `SystemID` must be present if the customer system is a multi-user system. The technical subscriber is responsible for the generation of the EBICS requests (including the identification and authentication signatures) that belong to orders that are submitted or bank-technically signed by the subscriber.
  - (Optional) information on the customer product (`ebicsNoPubKeyDigestsRequest/header/static/Product`)
  - Administrative Order type (`ebicsNoPubKeyDigestsRequest/header/static/OrderDetails/AdminOrderType`) set to “HPB”
  - Security medium for the subscriber’s bank-technical key (`ebicsNoPubKeyDigestsRequest/header/static/SecurityMedium`) set to “0000”.
    The security medium for the subscriber’s bank-technical key is set to “0000” since HPB orders neither require ES’s nor transmit bank-technical subscriber keys.
  - Identification and authentication signature of the technical subscriber, if such is available, otherwise the identification and authentication signature of the subscriber themselves (`ebicsNoPubKeyDigestsRequest/AuthSignature`)
    The identification and authentication signature includes all XML elements of the EBICS request whose attribute value for `@authenticate` is equal to “true”. The definition of the XML schema “ebics_keymgmt_request_H005.xsd“ guarantees that the value of the attribute `@authenticate` is equal to “true” for precisely those elements that must be signed

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsNoPubKeyDigestsRequest
 xmlns="urn:org:ebics:H005"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_keymgmt_request_H005.xsd"
 Version="H005" Revision="1">
 <header authenticate="true">
   <static>
     <HostID>EBIXHOST</HostID>
```

```
      <Nonce>234AB2340FD2C23035764578FF3091FA</Nonce>
      <Timestamp>2005-01-30T15:40:45.123Z</Timestamp>
      <PartnerID>CUSTM001</PartnerID>
      <UserID>USR100</UserID>
      <OrderDetails>
        <AdminOrderType>HPB</AdminOrderType>
             </OrderDetails>
      <SecurityMedium>0000</SecurityMedium>
    </static>
    <mutable/>
  </header>
  <AuthSignature>
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
      <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
      <ds:Reference URI="#xpointer(//*[@authenticate='true'])">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmldenc#sha256"/>
        <ds:DigestValue>…here hash value authentication...</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>…here signature value authentication...</ds:SignatureValue>
  </AuthSignature>
  <body/>
</ebicsNoPubKeyDigestsRequest>
```

Diagram 26: EBICS request for administrative order type HPB

- **Transmission of the following data in the EBICS response for HPB** (see example in Diagram 27):

  - Bank-technical return code (`ebicsKeyManagementResponse/body/ReturnCode`)
  - Technical return code (`ebicsKeyManagementResponse/header/mutable/ReturnCode`)
  - Technical report text (`ebicsKeyManagementResponse/header/mutable/ReportText`)
  - (Conditional) information for encryption of the order data and possibly the ES of the order data (`ebicsKeyManagementResponse/body/DataTransfer/DataEncryptionInfo`), if no errors of a technical or bank-technical nature have occurred.
    In particular, `DataEncryptionInfo` also contains the asymmetrically-encrypted transaction key (`ebicsKeyManagementResponse/body/DataTransfer/DataEncryptionInfo/TransactionKey`)
  - (Conditional) the order data (`ebicsKeyManagementResponse/body/DataTransfer/OrderData`), if no errors of a technical or bank-technical nature have occurred
  - (Optional) time stamp for the last updating of the bank parameters (`ebicsKeyManagementResponse/body/TimestampBankParameter`).

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsKeyManagementResponse
 xmlns="urn:org:ebics:H005"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_keymgmt_response_H005.xsd”
 Version="H005" Revision="1">
 <header authenticate="true">
   <static/>
   <mutable>
     <ReturnCode>000000</ReturnCode>
     <ReportText>[EBICS_OK] OK</ReportText>
   </mutable>
 </header>
 <body>
   <DataTransfer>
     <DataEncryptionInfo authenticate="true">
       <EncryptionPubKeyDigest Version="E002"
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256">..here hash value of public key for
encryption ..</EncryptionPubKeyDigest>
       <!-- asymmetricly encrypted transaction key -->
       <TransactionKey>…</TransactionKey>
       </DataEncryptionInfo>
     <!-- XML instance document using root element HPBResponseOrderData in accordance
with ebics_orders_H005.xsd, compressed and base64 encoded -->
     <OrderData>
       …
     </OrderData>
   </DataTransfer>
   <ReturnCode authenticate="true">000000</ReturnCode>
 </body>
</ebicsKeyManagementResponse>
```

Diagram 27: EBICS response for administrative order type HPB

4.5 Suspending a subscriber

4.5.1 Alternatives

If there is any suspicion that subscriber keys have been compromised, the subscriber MUST suspend their access authorisation to all bank systems that use the compromised key/s.

Subscribers that wish to suspend their remote access data transmission to a bank system can do this in two ways:

- The SPR transaction is a standard upload transaction transmitting the ES file exclusively containing the signature of the subscriber who is to be suspended with the help of a dummy file. The dummy file contains one blank character only and is not being transmitted. The corresponding EBICS request not only has to contain this signature but also an identification and authentication signature. The identification and authentication signature may also be provided by a technical subscriber.
  The SPR order does not comprise any additional order data and hence does not contain any order file either.

- In addition, the subscriber has the possibility of instigating the suspension via a second communication channel, e.g. by telephone via a specific contact of the financial institution. If a subscriber key gets lost or damaged, only this alternative is selectable.

  After successful execution of the suspension, the subscriber has the state “Suspended” and renewed initialisation of the subscriber is required.

4.5.2 Revoking a subscriber via SPR

SPR is a regular upload transaction. See Chapter 5.5 for a detailed description of the flow of the transaction (including its behaviour in cases of errors). Subsequently, only differences and supplements are given.

For SPR only an ES file is transmitted.. Processing is already being executed during the phase of initialisation, i.e. the bank system provides no transaction ID with the response.

The bank system has to ensure that the SPR request contains the identification and authentication signature of the subscriber who is to be revoked, or of the technical subscriber, respectively.
Verification of the customer/subscriber ID, the subscriber state and the identification and authentication signature takes place within the process step “Verifying authenticity of the EBICS request” (see Chapter 5.5.1.2.1, Diagram 41).

The ES file has to contain a valid electronic signature of the subscriber who is to be suspended by way of a file containing one blank character only.

The subsequent actual synchronous suspension of the subscriber does not return any further specific technical or bank-technical errors.

4.6 Key changes

In EBICS V 2.x, there were two options for the public key format of the ES key: a proprietary format or the xml element group x509, which contains special data related to the x509 standard. With EBICS V 3.0, the proprietary format has been dropped and now the element group x509 must always be used. However, for those who haven’t used certificates in EBICS V 2.x, this does not require a key change when migrating to EBICS V 3.0: Only if the EU key is not sufficiently long, i.e. less than 2048 bit, a key change must be carried out.
However, as of EBICS V 3.0, both new initializations as well as key changes must be carried out with certificates.

4.6.1 Changing the subscriber keys

With EBICS 2.3 and earlier versions, keys had to be changed by means of the administrative order types PUB (change of the bank-technical key) and HCA (change of the identification and authentication key as well as the encryption key). These changes could be executed independently. In order to simplify the key management at the customer's as well as the bank's end, with EBICS 2.4 the administrative order type HCS is introduced allowing all three keys of a single transaction to be modified. Therefore, order type HCS comprises PUB and HCA.
HCS – as well as PUB and HCA – require the bank-technical ES of the respective subscriber in the ES version supported in each case (e.g. A005, A006), but not the additional dispatch of initialisation letters. For reasons of compatibility, the administrative order types PUB and HCA can still be used as alternatives to HCS.
Depending on their state, the subscriber has two possibilities for updating their public subscriber keys on the bank system:

- With the state “Suspended”, subscriber initialisation MUST fundamentally be carried out so that bank-technical orders can be transmitted via EBICS. Hence suspension of access authorisation followed by subscriber initialisation is an alternative for activation of subscriber keys. Subscriber initialisation takes place using the administrative order types INI and HIA, and requires the additional sending of initialisation letters. The subscriber initialisation process is described in Chapter 4.4.1. Information on the subject of suspension of a subscriber’s access authorisation can be found in Chapter 4.5.
- When they have the state “Ready”, subscribers can update their public subscriber keys using the two administrative order types PUB, HCS and HCA without having to go the long way round with subscriber initialisation. In each case, PUB, HCS and HCA require the ES of the respective subscriber but not the additional dispatch of initialisation letters. On the one hand, this simplifies the key change process but on the other hand it removes the possibility of using initialisation letters to document a subscriber’s key history. It is the responsibility of the financial institution to document the key change via PUB, HCS and HCA so that it remains verifiable.

The subject of this chapter is the detailed description of key changing via PUB, HCS and HCA.

4.6.1.1 General description

Subscribers with the state “Ready” can update their public subscriber keys by using one of the following administrative order types:

- **PUB**: update the public bank-technical key
- **HCA:** update the public identification and authentication key and the public encryption key
- **HCS:** update the public bank-technical subscriber key, the public identification and authentication key and the public encryption key

PUB, HCS and HCA are regular upload transactions whose sequence (including behaviour in error situations) is described in detail in Chapter 5.5. Contained therein is Diagram 46 which describes the sequence of EBICS request handling by the bank during the data transfer phase of an upload request. A part of this procedure is the

process step “Verifying and processing of the order”. This step returns the following error codes for the administrative order type PUB:

- EBICS_KEYMGMT_UNSUPPORTED_VERSION_SIGNATURE
  This business related error occurs when the order data contains an inadmissible version of the bank-technical signature process
- EBICS_KEYMGMT_KEYLENGTH_ERROR_SIGNATURE
  This business related error occurs when the order data contains a bank-technical key of inadmissible length
- EBICS_INVALID_ORDER_DATA_FORMAT
  This business related error occurs when the order data does not correspond with the designated format (see Chapter 4.4.1.2.5.1)
- EBICS_USER_UNKNOWN
  This technical error occurs when the subscriber that is a component of the PUB order data is not a registered subscriber
- EBICS_UNKNOWN_USER_STATE
  This technical error occurs when the subscriber that is a component of the PUB order data does not have the state “Ready”
- EBICS_SIGNATURE_VERIFICATION_FAILED
  This business related error occurs when the ES of the subscriber in question could not be successfully verified.
- For Return codes relating to CA-issued certificates, refer to Annex 1

For HCA, the process step “Examination and processing of the order” returns the following error codes:

- EBICS_KEYMGMT_UNSUPPORTED_VERSION_ENCRYPTION
  This business related error occurs when the order data contains an inadmissible version of the encryption process
- EBICS_KEYMGMT_UNSUPPORTED_VERSION_AUTHENTICATION
  This business related error occurs when the order data contains an inadmissible version of the identification and authentication signature process
- EBICS_KEYMGMT_KEYLENGTH_ERROR_ENCRYPTION
  This business related error occurs when the order data contains an encryption key of inadmissible length
- EBICS_KEYMGMT_KEYLENGTH_ERROR_AUTHENTICATION
  This business related error occurs when the order data contains an identification and authentication key of inadmissible length
- EBICS_INVALID_ORDER_DATA_FORMAT
  This business related error occurs when the order data does not correspond with the designated format (see Chapter 4.4.1.2.5.1)

- EBICS_USER_UNKNOWN
  This technical error occurs when the subscriber that is a component of the HCA order data is not a registered subscriber
- EBICS_UNKNOWN_USER_STATE
  This technical error occurs when the subscriber that is a component of the HCA order data does not have the state “Ready”
- EBICS_SIGNATURE_VERIFICATION_FAILED
  This business related error occurs when the ES of the subscriber in question could not be successfully verified.
- For Return codes relating to CA-issued certificates, refer to Annex 1

HCS being a combination of PUB and HCA, all error codes in process step "Verifying and processing of the order" listed under PUB and HCA can be reported.

Either PUB and HCA or HCS must be submitted by the subscriber whose keys are to be updated. Each administrative order type PUB, HCS, and HCA require precisely one ES that must be supplied by the subscriber whose keys are to be updated. The signature class of this ES is irrelevant.

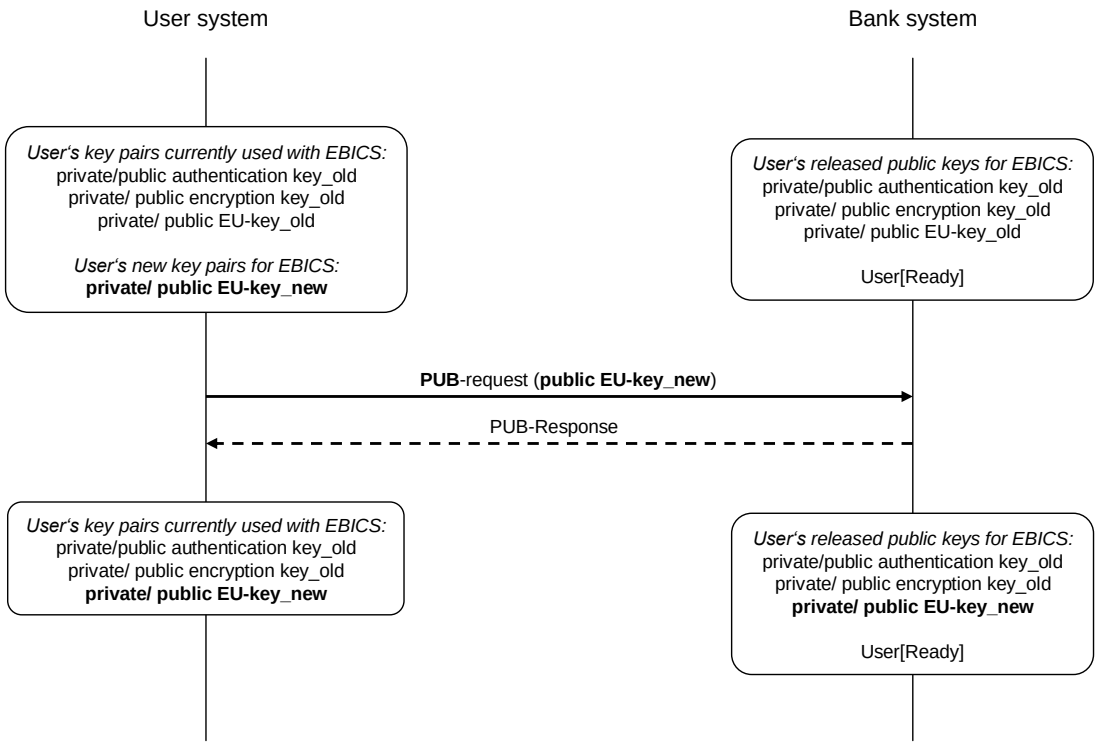


Diagram 28: Changing the bank-technical subscriber key via PUB

Diagram 28 represents the state of the public subscriber keys and the subscriber before and after processing of PUB. The following applies to the processing of PUB:

- The order data, i.e. the subscriber's new public bank-technical key, is compressed, encrypted and finally base64-coded, and is embedded into the EBICS messages.
- The order data is signed via ES by the subscriber whose public bank-technical key is to be updated. The subscriber's old bank-technical key (that is activated at this point) is used for this ES.

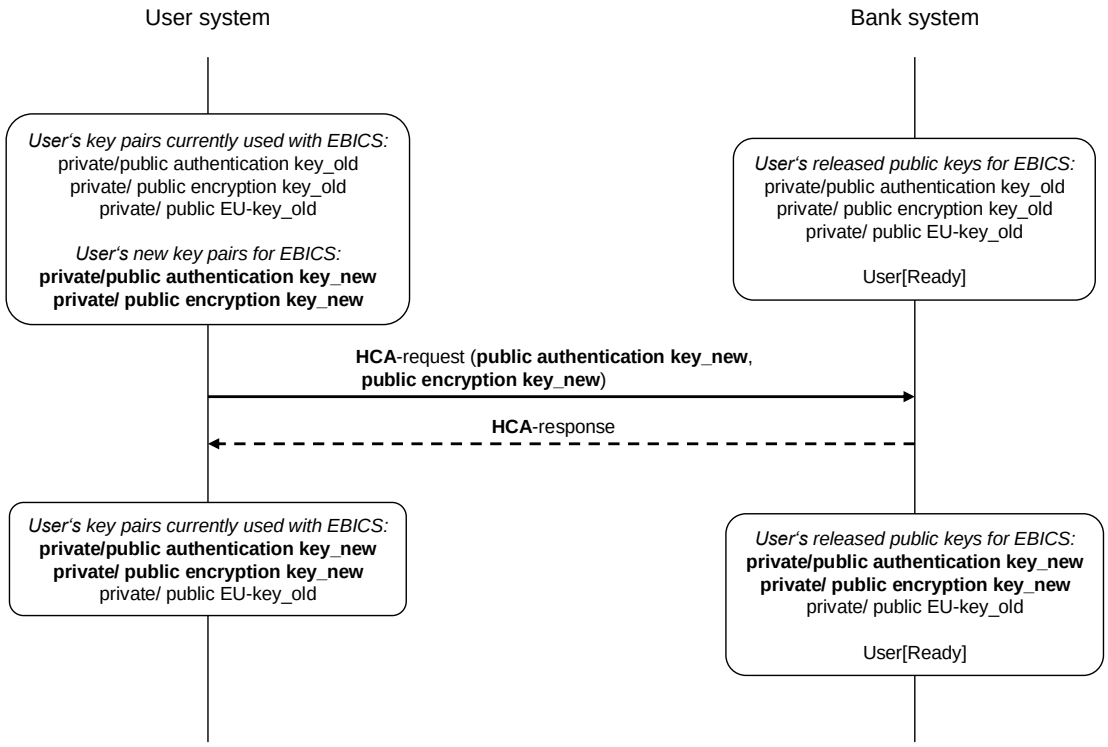


Diagram 29: Changing the authentication key and encryption key via HCA

Diagram 29 shows the state of the subscriber keys and the subscriber before and after the processing of HCA. In addition, the following applies to the processing of HCA:

- The order data, i.e. the subscriber's new public identification and authentication key and new public encryption key, is compressed, encrypted and finally base64-coded, and is embedded into the EBICS messages.
- HCA requests contain the identification and authentication signature of the affected subscriber or a technical subscriber. The identification and authentication signature of the affected subscriber is generated with the old identification and authentication signature (that is activated at this point). The financial institution's EBICS responses contain the financial institution's identification and authentication signature.

By using HCS all keys are changed. The administrative order type HCS can be regarded as an alternative to PUB and HCA which allow the keys for the bank-technical

electronic signature (PUB) and for the identification and authentication signature and encryption (HCA) only to be changed separately.
Therefore, the process looks like follows:

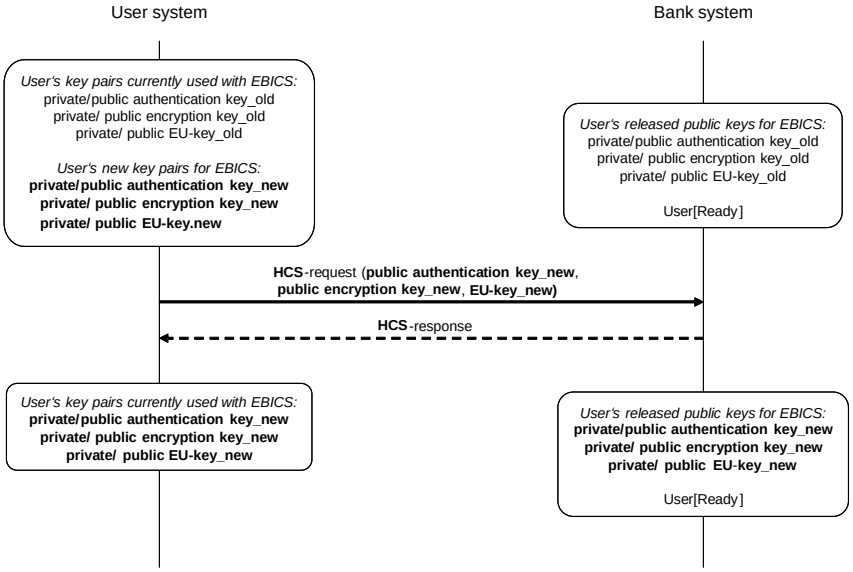


Diagram 30: Changing the bank-technical subscriber key, the authentication key, and encryption key via HCS

4.6.1.2 Format of the order data

When using the ES in structured form (from signature process A005/A006 on), the order data for PUB is an instance document that conforms with ebics_signature_S002.xsd and comprises the top-level element `SignaturePubKeyOrderData` (in compliance with INI, XML scheme representation see in chapter 4.4.1.2.5)

`SignaturePubKeyOrderData` is defined as follows via the XML schema:

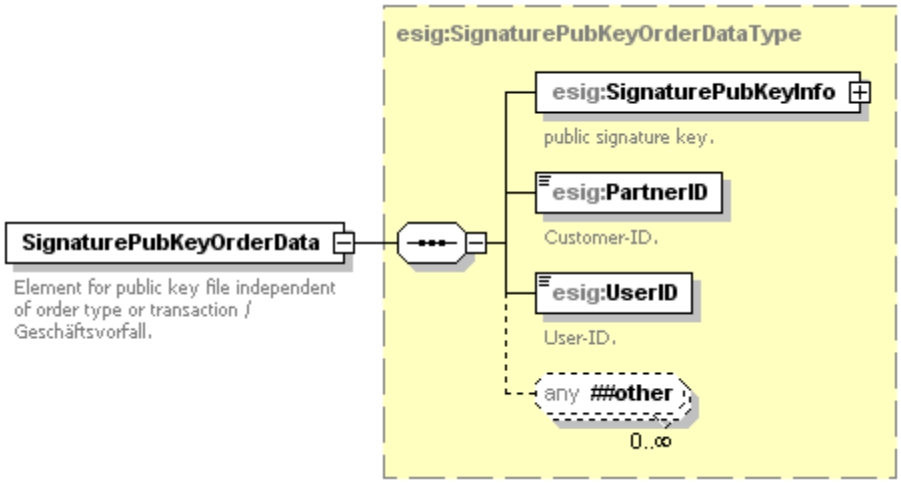


Diagram 31: Definition of the XML schema element SignaturePubKeyOrderData for PUB order data (identical to INI, see own chapter)

The order data for HCA is an instance document that conforms with ebics_orders_H005.xsd and comprises the top-level element `HCARequestOrderData`. `HCARequestOrderData` is defined as follows via the XML schema:

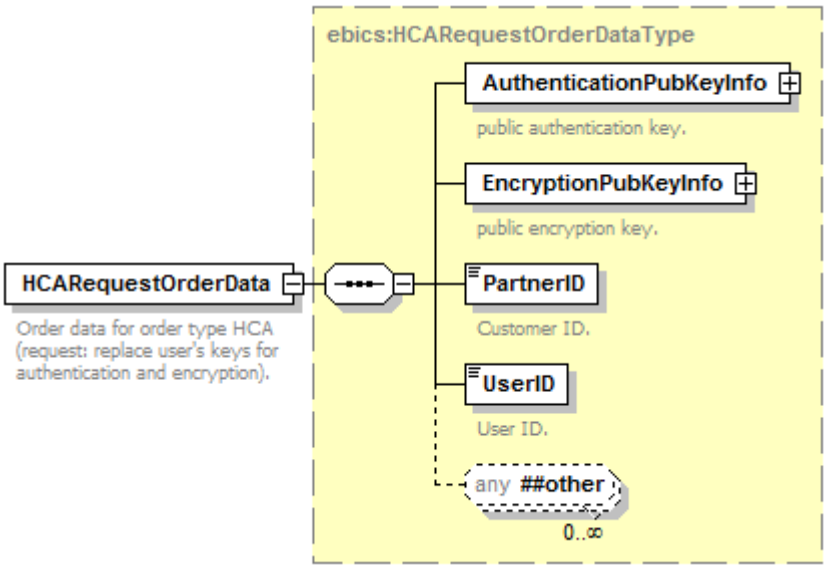


Diagram 32: Definition of the XML schema element HCARequestOrderData for HCA order data

The order data for HCS is an instance document that conforms with ebics_orders_H005.xsd and comprises the top-level element `HCSRequestOrderData`. `HCSRequestOrderData` is defined as follows via the XML schema:

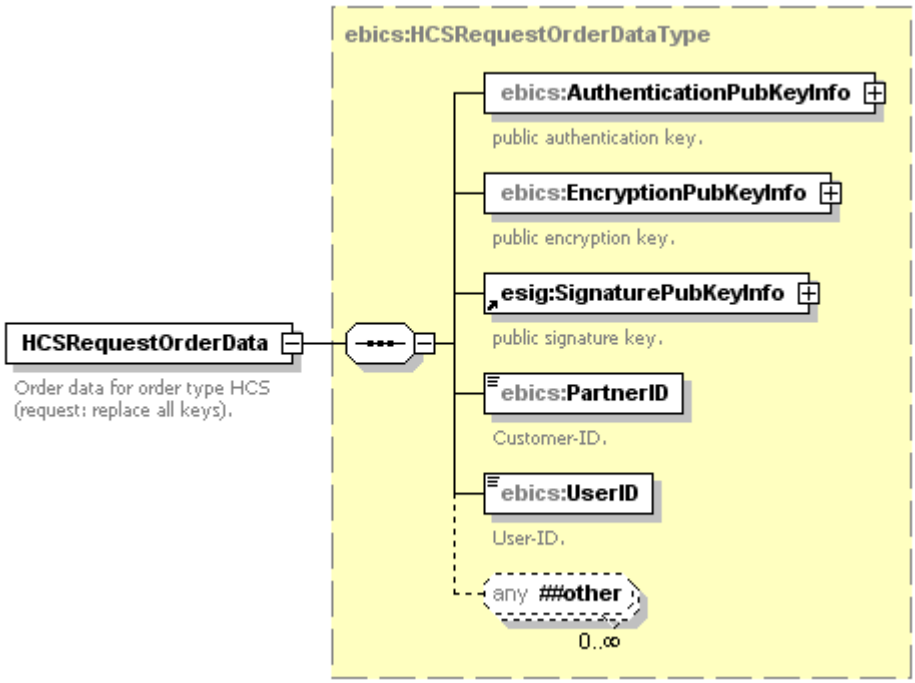


Diagram 33: Definition of the XML schema element HCSRequestOrderData for HCS order data

The order data for PUB, HCS, and HCA are compressed, encrypted and base64-coded, and embedded into the corresponding EBICS request.

4.6.2 Changing the bank keys

The process for updating bank keys is not a part of this standard, but at the end of this chapter there is a description for the automatic update of bank keys. The duration of validity of the bank keys is not part of the EBICS interface. From the point of view of the EBICS protocol, one set of currently-valid bank keys exist at any time and for any admissible combination of processes for the identification and authentication signature, encryption and the ES. In Version H005, this consists of precisely the following keys:

- Private/public encryption key for process E002
- Private/public identification and authentication key for process X002
- Private/public bank-technical key for process A005 or A006.

In EBICS there are no transition periods where more than one key is valid for the same process. Keys changed at the bank's end are immediately valid in EBICS.

If the bank provides new keys the subscriber is responsible for download of the respective current bank keys via HPB. HPB always delivers the current keys (key for

identification and authentication as well as key for encryption) and these are signed by the identification and authentication signature of the bank side. However for this signature the former key for identification and authentication is used. For this procedure “signature value” in the x.509 structure contains the identification and authentication signature (built up with the former current bank key) and the 1 to n occurences of “public key” contain the current public keys.

When processing of HPB for the first time (initialisation of the subscriber) the verification of this signature is not possible as the subscriber doesn’t hold former keys to verify the signature. The state of the bank keys at the subscriber’s end is equal to “New” in this moment. The bank keys may not (yet) be used for communication via EBICS while they have this state. The financial institution MUST make the new keys accessible by means of a second, independent, communication channel. As with initial download of the bank keys, the subscriber MUST carry out a comparison of the keys and/or its hash-values received via the different communication channels immediately after initialisation. After successful verification of the bank keys, their state is “activated” at the subscriber’s side. When they have the state “activated”, the bank keys can be used for communication via EBICS.”
When reprocessing HPB (further time for the subscriber due to the update of bank keys) the identification and authentication signature can be verified as the subscriber’s end holds the former keys.
Case 1:
When the verification is successful the state of the bank keys remains “activated”. The bank keys will be exchanged in the client system.
Case 2:
When the verification is not successful the subscriber is informed about this failure and the state of the bank keys is reset to “new”. As in the initialisation case he MUST carry out a comparison of the keys and/or its hash-values received via the different communication channels.

In order to ensure that the subscriber has the current bank keys, the sequence of an EBICS transaction (with the exception of INI, HIA, HPB) in the first EBICS request provides for the transmission of the hash value of the financial institution’s public keys (XML structure `ebicsRequest/header/static/BankPubKeyDigests`) with which the subscriber has been provided. The bank system verifies whether the these keys are up-to-date and returns the result of the verification to the subscriber. If one of these is no longer current, the transaction is terminated with the technical return code EBICS_BANK_PUBKEY_UPDATE_REQUIRED. The subscriber must then download the bank keys with HPB.

In Version “H005” of the EBICS protocol the ES of the financial institutions is only planned (see Chapter 3.5.2). In preparation for future versions of EBICS, the XML structure `BankPubKeyDigests` contains the hash value of the public bank-technical key with the maximum frequency being equal to 0. Further details on verifying the hash value can be found in Chapter 5.5.1.2.

Standard Approach for the automatic update of bank keys:

- The first bank key (for authentication and encryprion as well) is a self-signed certificate (Z1).
- All subsequent, new keys (Z2 – Zn) are normally generated as certificates which are signed (subscribed) by the preceeding certificate (Z2 → signed by Z1, Z3 by Z2, Zn by Zn-1).
- The data digest of HPB relates to the current certificate.
- The bank server delivers (via HPB) no certificate chains but only the current certificate, signed by the preceding one.
- It is a basic rule that the bank decides if the current certificate will be signed with the preceeding key. If the current certificate isn‘t signed by the bank, the (manual) release procedure for the key will be necessary (as before).
- For the clients there is **no change for the first download**: The data digest of the certificate has to be checked by the well-known manual procedure.
  There is no difference between self-signed and CA certificates.
- If the bank signs the certificate **in the 2nd ... nth download,** the client can try to validate the signature of the current certificate with the key of the preceeding certificate. **In the case of failure** he has to use the manual procedure. **In the case of success** he can use the key directly and the preceeding one expires.

4.7 Change-over to longer key lengths

The key lengths must continually be increased to guarantee the security of the RSA process. See the regular publications of the “Übersicht über geeignete Algorithmen” from the Regulierungsbehörde für Telekommunikation und Post.

The subject of this chapter is the transition to keys of greater length in EBICS.

In version “H005” Bank-technical keys of a minimum length of 2048 are to be used.

In Version “H005”, EBICS sets a minimum length of 2048 bits (= 2 Kbit) and a maximum length of 16 Kbit for identification and authentication keys and encryption keys. The minimum length must be changed when keys of this length are no longer to be used for security reasons. The maximum length must be changed when keys that are longer than this maximum length are allowed to be supported.

The order data formats of the administrative order types HIA, HPB, HCA, and HCS permit key lengths of any size. This means that these order data formats will not require adaptation after the key lengths have been increased.

New public identification and authentication keys or encryption keys of greater length will be transmitted to the bank systems via HIA, HCS, or HCA in exactly the same way as new identification and authentication keys of consistent length.

In the same way, the financial institution’s new public keys will be downloaded via HPB irrespective of whether the length of the financial institution’s identification and authentication key or encryption key has changed.

4.8 Summary

The following table summarises the most important features of the (administrative) key management order types:

Admin. Order type	Order data format	Identification and authentication signature subscriber / financial institution	Order data ES
INI	INI file (in accordance with Chapter 14)	no/no	no
HIA	ebics: HIARequestOrder» Data	no/no	no
HPB	ebics: HPBRequestOrder» Data	yes/no	no
PUB	see INI	yes/yes	yes
HCA	ebics: HCARequestOrder» Data	yes/yes	yes
HCS	ebics: HCSRequestOrder» Data	yes/yes	yes
H3K	ebics: H3KRequestOrder» Data	no/no	yes (certificate)
SPR	Sole key management order which only contains an ES	yes/yes	yes

5 EBICS transactions

The descriptions and stipulations in this chapter apply to all business transactions (s identified by BTF) and all administrative orders with the exception of the following administrative key management orders : INI, HIA, HPB, PUB, SPR, HCA, H3K and HCS.

5.1 General provisions

5.1.1 EBICS transactions

EBICS transactions serve for the transmission of orders to the bank-technical target system. Corresponding to the subdivision of orders into transmit and download orders, EBICS differentiates between upload and download transactions: Upload transactions transmit bank-technical order data and/or bank-technical signatures to the bank-technical target system; conversely, with a download transaction bank-technical order data and/or bank-technical signatures are downloaded from the bank-technical target system.

5.1.2 Transaction phases and transaction steps

Each EBICS transaction passes through different transaction phases. The phases of an upload transaction are initialisation and data transfer, the phases of a download transaction are initialisation, data transfer and finally acknowledgement. A transaction phase can comprise one or more connected transaction steps, wherein a transaction step is deemed to denote a pair comprising an EBICS request and an associated EBICS response. In this way, initialisation comprises the first initialisation step, but on the other hand data transfer can extend over several transaction steps, in each of which one order data segment is transmitted.

EBICS transactions can comprise one single transaction step, for example when they just transmit the bank-technical electronic signature for an order.

5.1.3 Processing of orders

5.1.3.1 Chronological dependencies between transmission and processing of upload orders

EBICS supports the chronological decoupling of the submission of bank-technical upload orders via EBICS from their actual processing on the back-end systems of the financial institution. The ES's and order data segments that are submitted within an EBICS transaction are firstly pre-processed. This **pre-processing** is not a component of EBICS, it is dependent on the implementation of the bank system, for example the intermediate storage of the order data segments is a part thereof. After transmission of the last order data segment the entire order data, order parameters and ES's are firstly

passed on to a component of the bank system that is responsible for the **management of pending orders**. Realisation is dependent on the implementation of the bank system, it is not a component of EBICS.

In contrast to the bank-technical upload orders, it is required that **processing** of the upload orders of administrative order types MUST be completed before transmission of the last EBICS response of the upload transaction. In addition to the (administrative) key management order types, this requirement also applies to download orders of EDS order types so that the distributed ES process can be handled as efficiently as possible and the involved subscribers can be given the must up-to-date state of the distributed ES's of an order.

5.1.3.2 Chronological dependencies between transmission and processing of download orders

The download data is a component of the financial institution's EBICS response. The bank system makes a further order data segment available with each EBICS transaction step. In order to accelerate the download process, the download data can be generated by the bank system in advance (such as e.g. in the case of account statements) or can not be generated until required.

5.1.4 Transaction administration

Control of the development of an EBICS transaction is normally incumbent on the customer system, the individual transaction steps of an EBICS transaction are each initiated by the customer system. In special cases, the bank system can also control the development of a transaction, e.g. in that it informs the customer system of a possible recovery point in the event of a recovery.

The EBICS transactions must also be administrated in the bank system to allow the following:

- Assignment of the individual transaction steps to a specific EBICS transaction.
- recording of the process of the EBICS transaction for administration of the transaction states with the objective of ensuring the progress of the EBICS transaction.
- Recovery of an EBICS transaction.

This produces the following responsibilities for the bank system's EBICS transaction administration:

- Generation of EBICS transactions during transaction initialisation. See Chapter 5.2 for details.
- Aborting EBICS transactions if continuation is not expedient or not possible due to the occurrence of error situations
- Termination of EBICS transactions if it has been possible to carry out all transaction steps successfully

- Verifying the process of EBICS transactions to ensure their sequence in accordance with Chapter 5.5.1.
- Supporting the process for recovering EBICS transactions in accordance with Chapter 5.5.2 and 5.6.2 if the bank system supports recovery.

5.2 Assignment of EBICS request to EBICS transaction

The first phase of every EBICS transaction is the initialisation phase. It is triggered by the first EBICS request of the transaction, and comprises:

- Verifications, wherein the successful execution of these verifications is a necessary prerequisite for acceptance of the order by the financial institution
- Further processing steps that are necessary for acceptance of EBICS transactions that comprise more than one transaction step into the transaction administration system

Examples of such verifications are checks on the state and the BTF identifiers authorisation of the subscriber that has submitted the order. The precise scope of these verifications/process steps is described in Chapter 5.5.1.2.1 for upload transactions and in Chapter 5.6.1.2.1 for download transactions .

If all necessary verifications have been successfully carried out and if the transaction comprises several transaction steps, the bank system's transaction administration generates an EBICS transaction with a transaction ID that is unambiguous within the bank system (details on generation of the transaction ID can be found in the Appendix (Chapter 11.6). The subscriber is notified of this via the financial institution's reply message. The bank system's transaction administration permanently assigns this transaction the following data, which is a component of the header data of the EBICS request:

- Customer ID, subscriber ID/technical subscriber ID
- Administrative Order type
- Order parameters (depend on kind of transaction which can be administrative order or a business-related order, see 3.11)
- Order number (only allowed for administrative order types HVE and HVS)

The order number is only present if a file is transmitted to the bank relating to an order with already existing order number (this is only valid for the transmission of an ES file with with the administrative order types HVE or HVS) for matching files with the same order number. Basically the order number is a component of the header data of the EBICS response (in uploads). It is assigned by the bank server automatically.

This data is permanently assigned to the transaction and cannot be changed in the course of the transaction.

Outside of the initialisation process, EBICS requests contain these transaction IDs for assignment to suitable EBICS transactions. As a whole, they contain the following elements that identify the transaction step:

- Transaction ID that is unambiguous throughout the bank system
- Transaction phase (initialisation, data transfer, acknowledgement) within the transaction
- Serial number of the data segment of bank-technical data, if in the data transfer transaction phase.

A detailed description of the structure of EBICS requests for upload and download transactions can be found in Chapters 5.5.1.1 and 5.6.1.1.

5.3 Preliminary verification of orders [optional]

The bank system CAN optionally support preliminary verification functionality to avoid the possibility of subscribers transmitting large quantities of data to the bank system, wherein it is only discovered by the bank after the transmission has taken place that the signatory of the upload order did not have the necessary authorisation. The information as to whether a bank system supports preliminary verification is contained in its retrievable bank parameters (see Chapter 12.2). If preliminary verification of upload orders is supported, determination of the scope of the preliminary verification is the responsibility of the individual financial institution. Support of one or more of the following verifications is possible:

- **Account authorisation verification**

The account authorisation verification ensures that the following condition is complied with for each signatory:

- The signatory is authorised to provide an ES of at least type “B” for orders of the specified BTF identifiers for each of the order party accounts in a given order.

- **Limit verification**

The limit verification ensures that the following condition is complied with for each signatory:

- The signatory is authorised to provide an ES of at least type “B” for orders of the specified BTF identifiers and to the respective amount for each of the order party accounts in a given order.

- **ES verification**

The ES verification verifies the ES of the signatory of the order and checks in each case as to whether the ES’s originate from different subscribers.
For a successful preliminary verification the subscriber requires one of the signature classes "E", "A", or "B". If orders are submitted only (i.e. signature class "T") the preliminary verification is not run through. The return code "EBICS SIGNATURE VERIFICATION FAILED" is returned if signature is not valid.

If the order at hand has already been signed the return code "EBICS DUPLICATE SIGNATURE" is returned.

Preliminary verification of an upload order is a part of the first transaction step within the framework of the corresponding upload transaction. The results of the preliminary

verification are given in the bank-technical return code in the corresponding EBICS response of the first transaction step. Preliminary verification takes place before transmission of the order's order data, based on information from the customer system about the order data that is still outstanding. It does not replace the corresponding verifications that are based on the actual order data after its transmission to the bank system.

The customer system CAN further limit the scope of the preliminary verifications. The account authorisation, limit or bank-technical ES preliminary verifications are only carried out by the bank if the data necessary for their execution is made available by the customer system. The preliminary verification data is transmitted in the first EBICS request of an upload transaction via the (optional) element `ebicsRequest/body/PreValidation` (see ebics_request_H005.xsd. The type is called `PreValidationRequestType` (see ebics_types_H005.xsd) and comprises a list of the following optional elements:

- `DataDigest`

  This element contains the hash value of the order data that has been signed by the order signatories via transport signature or bank-technical ES. During preliminary verification of an order, the ES's are verified solely on the basis of this hash value, the correctness of which cannot be verified at the time of verification.
  For the signature process used by the order signatories (and, relating to EBICS, supported by the bank) a hash value can be set which is to be calculated by the hash function of the respective signature process. The appropriate signature process is identified by means of the attribute `SignatureVersion`. `DataDigest` may occur multiple times if the signatories use different signature versions (this is the case if not every subscriber of a customer signs using the same signature process). This is the reason why the correct setting of the attribute `SignatureVersion` is so important for each `DataDigest`.

- `AccountAuthorisation`

  This element contains an order party account for the given order. For this account, the account number (`AccountAuthorisation/AccountNumber`) is given in German and/or international format and the bank code (`AccountAuthorisation/BankCode`) is given in German and/or international format. As an option, the account holder (`AccountAuthorisation/AccountHolder`) can also be provided. This account information is required by the account authorisation and limit verifications. In addition, the limit verification requires specification of the total for the individual orders relating to the given order party account. This amount is contained in the (optional) element `AccountAuthorisation/Amount`. The currency of the amount is the value of optional attribute `AccountAuthorisation/Amount@Currency`, if this is available. Otherwise the currency is the value of attribute `AccountAuthorisation@Currency`, which contains the currency of the account.

The individual preliminary verifications require the subscriber ID/ customer ID of each individual signatory. These are a component of the XML type OrderSignature that is used to represent individual ES's. See Chapter 3.5 for further details on embedding ES's into EBICS messages.

5.4 Recovery of transactions [optional]

This chapter describes the basic principles of the recovery procedure that apply to both upload and download transactions.

The EBICS recovery mechanism means that a transaction's order data that has already been received by the customer or bank system does not have to be re-transmitted if one of the following error situations occurs:

- Transport error
- Processing error in the EBICS message that contains the order data:
  In the case of upload transactions these are EBICS request processing errors that can occur at the bank's end, in the case of download transactions they are EBICS response processing errors that occur at the customer's / subscriber's end. For example, errors can occur during (intermediate) storage of order data.
  Recovery is an important aspect of the protocol, since the size of the order data can certainly reach a magnitude of several hundred megabytes.

The mechanism requires knowledge of the transaction ID of the EBICS transaction in question, and is based on the definition of transaction **recovery points**:

- In the case of upload transactions, the recovery point is the last transaction step in the transaction whose EBICS request has been successfully received by the bank system and whose EBICS response has been successfully transmitted. The recovery point is determined by the state of the transaction in the bank system.
- In the case of download transactions, there may be several recovery points. These are all of the previous transaction steps in the transaction in question whose EBICS request has been successfully received by the bank system and whose EBICS response has been successfully transmitted.

After transport or processing errors have occurred, a recovery point can be used to continue transactions from the transaction step that follows this recovery point in the transaction step sequence.

All EBICS requests relating to an open transaction that do not match the state of this transaction are evaluated by the bank system's EBICS transaction administration as recovery attempts.

In order to guarantee progress of the EBICS transactions, the number of possible recovery attempts per transaction MUST be limited by a maximum value. The bank system's transaction administration is responsible for administration of the corresponding counter for each transaction. Transactions whose counter exceeds the permitted limit will be terminated by the bank system's transaction administration. In addition, the bank system

CAN limit the number of open transactions with a positive recovery counter for each subscriber by setting a maximum value. The counter for recovery attempts that have already been initiated for each transaction and/or the counter for the pending transactions in recovery mode for each subscriber and also the permitted maximum numbers are not a part of the EBICS messages. Instead, they are a part of the processing of the EBICS transaction administration at the bank's end.

Analogously, the customer system's transaction control limits the number of attempts made to successfully carry out a particular transaction step in an EBICS transaction. In this case, counters and permitted maximum numbers are not part of the EBICS messages but are merely part of the processing of transaction control at the customer's end.

Details on recovery of upload and download transactions are given in Chapters 5.5.2 and 5.6.2.

5.5 Upload transactions

5.5.1 Sequence of upload transactions

The sequence of an upload transaction is shown in Diagram 34 by means of a flow diagram. Transmission of the order data segments takes place within a loop that is broken off when the last order data segment has been transmitted (note partial expression "[last data segment has been transmitted]" from the termination conditions). The sequence clarifies that order data does not necessarily also have to be transmitted within an upload transaction note partial expression "[AdminOrderType = HVE or HVS]" from the termination conditions). This is the case when only a bank-technical ES relating to an existing order is transmitted via HVE (add ES) or HVS (cancellation of order) .

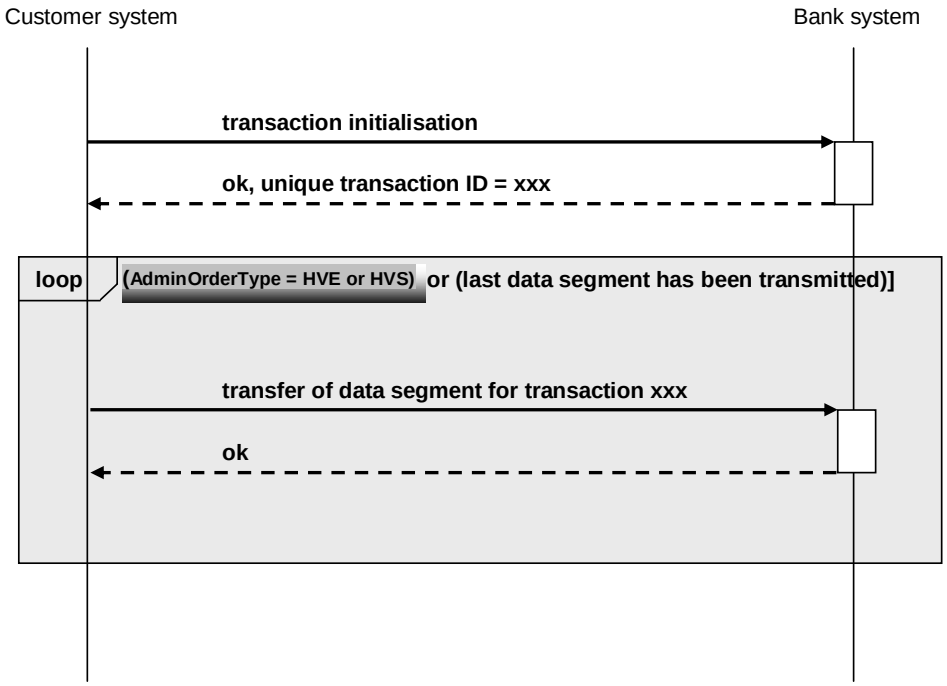


Diagram 34: Error-free sequence of an upload transaction

5.5.1.1 Description of the EBICS messages

For clarification purposes, the following description of the transaction steps in a business transaction format (BTF-)upload use example messages for the processing of the service “sct” (SEPA credit transfer) . It refers to elements of these example messages, using XPath notation.

The following chapters describe the messages in the individual transaction phases. The data that is a component of these messages is listed here. Data that is fundamentally optional is marked “(optional)”. Data that may only be missing under certain conditions is instead marked “(conditional)”. Optional XML elements that are missing in the description of an EBICS message relating to a specific transaction phase may not be present in this EBICS message. Optional XML elements that are present in the description of an EBICS message relating to a specific transaction phase MUST always be placed correspondingly in this EBICS message.

EBICS requests for upload transactions are (XML) instance documents that conform to ebics_request_H005.xsd and comprise the top-level element `ebicsRequest` which is declared in ebics_request_H005.xsd. EBICS responses for upload transactions are instance documents that conform to ebics_response_H005.xsd and comprise the top-level element `ebicsResponse` which is again declared in ebics_response_H005.xsd.

5.5.1.1.1 EBICS messages in transaction initialisation

- Transmission of the following data in the EBICS request (see Diagram 35):

- Host ID of the EBICS bank computer system (`ebicsRequest/header/static/HostID`)
- Transaction phase (`ebicsRequest/header/mutable/TransactionPhase`) with the setting “Initialisation”
  - Combination of Nonce and Timestamp, necessary to avoid replaying old EBICS messages (`ebicsRequest/header/static/Nonce`, `ebicsRequest/header/static/Timestamp`)
- Number of data segments to be transmitted (`ebicsRequest/header/static/NumSegments`)
- Subscriber (`ebicsRequest/header/static/PartnerID`, `ebicsRequest/header/static/UserID`) that is submitting a new order or that is providing bank-technical ES’s for an existing order.
- (Conditional) technical subscribers (`ebicsRequest/header/static/PartnerID`, `ebicsRequest/header/static/SystemID`)
  SystemID must be present if the customer system is a multi-user system. The technical subscriber is responsible for the generation of the EBICS request (including the identification and authentication signatures) that belong to orders that are submitted or bank-technically signed by the subscriber.
- (Optional) information on the customer product (`ebicsRequest/header/static/Product`)
- Administrative Order type (`ebicsRequest/header/static/OrderDetails/AdminOrderType`)
- (Conditional) Order number (`ebicsRequest/header/static/OrderDetails/OrderID`)
  OrderID is only present if a file is transmitted to the bank relating to an order with an already existing order number (only allowed for AdminOrderType = HVE or HVS)
- Order parameters (`ebicsRequest/header/static/OrderDetails/OrderParams`); the characteristics of the order parameters are dependent on the administrative order type (see also Chapter 3.11). For the upload of business transaction formats the order parameters and usage rules are specified in detail in chapter 5.5.1.1.3
- Hash values of the financial institution’s public keys that are available to the subscriber (ebicsRequest/header/static/BankPubKeyDigests/Authentication, ebicsRequest/header/static/BankPubKeyDigests/Encryption, ebicsRequest/header/static/BankPubKeyDigests/Signature).
  Both the utilised hash algorithm and the version of the corresponding identification and authentication, encryption and signature process will be specified for each of these hash values.
  The SHA-256 hash values of the financial institution's certificates X002 and E002 are composed by calculating the SHA-256 hash value of the certificate in DER binary format. In Version “H005” of the EBICS protocol the ES of the financial institutions is only planned (see Chapter 3.5.2). The element `BankPubKeyDigests/Signature` is already contained in this description in preparation for future versions of EBICS, but in Version “H005” its maximum frequency (maxOccurs) is set to 0.

- Security medium for the subscriber's bank-technical key(`ebicsRequest/header/static/SecurityMedium`)
- Identification and authentication signature of the technical subscriber, if such is available, otherwise the identification and authentication signature of the subscriber themselves (`ebicsRequest/AuthSignature`)
  The identification and authentication signature includes all XML elements of the EBICS request whose attribute value for `@authenticate` is equal to "true". The definition of the XML schema "ebics_request_H005.xsd" guarantees that the value of the attribute `@authenticate` is equal to "true" for precisely those elements that also need to be signed.
- (Optional) data for preliminary verification of the order (`ebicsRequest/body/PreValidation`)
- Information for encryption of the ES's and order data (`ebicsRequest/body/DataTransfer/DataEncryptionInfo`) which especially also contains the asymmetrically-encrypted transaction key (`ebicsRequest/body/DataTransfer/DataEncryptionInfo/TransactionKey`)
- ES's of the order's order data (`ebicsRequest/body/DataTransfer/SignatureData`)
  SignatureData contains an instance document that conforms to "ebics_orders_H005.xsd" and contains UserSignatureData as a top-level element. This instance document has been compressed with ZIP, encrypted for the financial institution and finally base64-coded before being embedded into the EBICS request (see Appendix (Chapter 11.2.2)). Diagram 36 contains an example of such an instance document that contains a single ES. The setting for the attribute `PartnerID` in the document `UserSignatureData` must be identical to the submitter's customer ID in the element `ebicsRequest/header/static/PartnerID`.
- The data digest of the order data (`ebicsRequest/body/DataTransfer/DataDigest`)
  The attribute `ebicsRequest/body/DataTransfer/DataDigest @SignatureVersion` specifies the Version of the signature process used for computation of the data digest.
- Additional order information (optional) (`ebicsRequest/body/DataTransfer/AdditionalOrderInfo`)

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
xmlns="urn:org:ebics:H005"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:org:ebics:H005 ebics_request_H005.xsd"
Version="H005" Revision="1">
  <header authenticate="true">
    <static>
      <HostID>EBIXHOST</HostID>
      <Nonce>BDA2312973890654FAC9879A89794E65</Nonce>
      <Timestamp>2005-01-30T15:30:45.123Z</Timestamp>
      <PartnerID>CUSTM001</PartnerID>
      <UserID>USR100</UserID>
```

```
      <Product Language="en" InstituteID="Institute ID">Product Identifier</Product>
      <OrderDetails>
        <AdminOrderType>BTU</AdminOrderType>
                <BTUOrderParams>
              <Service>
                 <ServiceName>SCT</ServiceName>
                 <MsgName>pain.001</MsgName>
              </Service>
      </BTUOrderParams/>     </OrderDetails>
      <BankPubKeyDigests>
        <Authentication Version="X002"
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256">1H/rQr2Axe9hYTV2n/tCp+3UIQQ=</Authentica
tion>
        <Encryption Version="E002"
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256">2lwiueWOIER823jSoiOkjl+woeI=</Encryption
>
      </BankPubKeyDigests>
      <SecurityMedium>0000</SecurityMedium>
      <NumSegments>2</NumSegments>
    </static>
    <mutable>
      <TransactionPhase>Initialisation</TransactionPhase>
    </mutable>
  </header>
  <AuthSignature>
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
      <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256">
      </ds:SignatureMethod>
      <ds:Reference URI="#xpointer(//*[@authenticate='true'])">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
        <ds:DigestValue>…here hash value authentication..</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>…here signature value authentication..</ds:SignatureValue>
  </AuthSignature>
  <body>
    <PreValidation authenticate="true">
      <DataDigest SignatureVersion="A006">
MTIzNDU2Nzg5MDEyMzQ1Njc4OTAxMjM0NTY3ODkwMTI=</DataDigest>
    </PreValidation>
    <DataTransfer>
      <DataEncryptionInfo authenticate="true">
        <EncryptionPubKeyDigest Version="E002"
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256">..here hash value of the public bank key
for encryption..</EncryptionPubKeyDigest>
        <TransactionKey>EIGI4En6KEB6ArEzw+iq4N1wm6EptcyxXxStA…</TransactionKey>
        <HostID>EBIXHOST</HostID>
      </DataEncryptionInfo>
      <SignatureData authenticate="true">n6KEB6ArEzw+iq4N1wm6EptcyxXxStAO…</SignatureData>
      <DataDigest SignatureVersion="A006">
MTIzNDU2Nzg5MDEyMzQ1Njc4OTAxMjM0NTY3ODkwMTI=</DataDigest>
    </DataTransfer>
  </body>
</ebicsRequest>
```

Diagram 35: EBICS request for transaction initialisation for a business transaction format upload

```
<?xml version="1.0" encoding="UTF-8"?>
<UserSignatureData
xmlns="http://www.ebics.org/S002"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ebics.org/S002
http://www.ebics.org/S002/ebics_signature_S002.xsd">
<UserSignatureData>
    <OrderSignatureData>
        <SignatureVersion>A005</SignatureVersion>
        <SignatureValue>EUXkQa……</SignatureValue>
        <PartnerID>PARTNER1</PartnerID>
        <UserID>User1</UserID>
    </OrderSignatureData>
</UserSignatureData>
```

Diagram 36: XML document that contains the ES's of the signatory of the upload order

- Transmission of the following data in the EBICS response (see also example in Diagram 37)

- Bank-technical return code (`ebicsResponse/body/ReturnCode`)
- Order number (`ebicsResponse/header/mutable/OrderId`)
- Technical return code (`ebicsResponse/header/mutable/ReturnCode`)
- Technical report text (`ebicsResponse/header/mutable/ReportText`)
- (Conditional) Transaction ID that is unambiguous throughout the bank system (`ebicsResponse/header/static/TransactionID`), if the following conditions are met:
  - No errors of a technical or bank-technical nature have occurred during transaction initialisation
  - Within the current transaction, order data segments are transmitted in further subsequent transaction steps, i.e. the AdminOrderType is not HVE or HVS.
- Transaction phase (`ebicsResponse/header/mutable/TransactionPhase`) with the setting "Initialisation"
- Identification and authentication signature of the financial institution (`ebicsResponse/AuthSignature`)
  The identification and authentication signature includes all XML elements of the EBICS response whose attribute value for `@authenticate` is equal to "true". The definition of the XML schema "ebics_response_H005.xsd" guarantees that the value of the attribute `@authenticate` is equal to "true" for precisely those elements that must be signed
- (Optional) time stamp for the last updating of the bank parameters (`ebicsResponse/body/TimestampBankParameter`).

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsResponse
xmlns="urn:org:ebics:H005"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:org:ebics:H005 ebics_response_H005.xsd"
Version="H005" Revision="1">
  <header authenticate="true">
    <static>
      <TransactionID>ABCDEF413946443634453132 43ABCDEF</TransactionID>
    </static>
```

```
    <mutable>
      <TransactionPhase>Initialisation</TransactionPhase>
      <OrderId>OR01</OrderId>
      <ReturnCode>000000</ReturnCode>
      <ReportText>[EBICS_OK] OK</ReportText>
    </mutable>
  </header>
  <AuthSignature>
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
      <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
      <ds:Reference URI="#xpointer(//*[@authenticate='true'])">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
        <ds:DigestValue>…here hash value authentication..</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>…here signature value authentication..</ds:SignatureValue>
  </AuthSignature>
  <body>
    <ReturnCode authenticate="true">000000</ReturnCode>
  </body>
</ebicsResponse>
```

Diagram 37: EBICS response for transaction initialisation for the upload order

5.5.1.1.2 EBICS messages in the phase data transfer of a order data segment

- Transmission of the following data in the EBICS request (see example in Diagram 38):

- Host ID of the EBICS bank computer system (`ebicsRequest/header/static/HostID`)

Data for identification of the current transaction step:

- Transaction ID (ebicsRequest/header/static/TransactionID)
- Transaction phase (`ebicsRequest/header/mutable/TransactionPhase`) with the setting “Transfer”
- Serial number of the order data segment (`ebicsRequest/header/mutable/SegmentNumber`)
  The attribute `ebicsRequest/header/mutable/SegmentNumber@lastSegment` specifies whether this is the last data segment.

- Identification and authentication signature of the technical subscriber, if such has been defined for the current transaction, otherwise the identification and authentication signature of the submitting subscriber themselves (`ebicsRequest/AuthSignature`)
  The identification and authentication signature includes all XML elements of the EBICS request whose attribute value for `@authenticate` is equal to “true”. The definition of the XML schema “ebics_request_H005.xsd“ guarantees that the value of the attribute `@authenticate` is equal to “true” for precisely those elements that also need to be signed

- The actual order data segment (`ebicsRequest/body/DataTransfer/OrderData`)

(see Chapter 3.3 and Chapter 7 for details on the segmentation of order data).

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
 xmlns="urn:org:ebics:H005"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_request_H005.xsd"
 Version="H005" Revision="1">
 <header authenticate="true">
   <static>
     <HostID>EBIXHOST</HostID>
     <TransactionID>ABCDEF41394644363445313243ABCDEF</TransactionID>
   </static>
   <mutable>
     <TransactionPhase>Transfer</TransactionPhase>
     <SegmentNumber lastSegment="true">4</SegmentNumber>
   </mutable>
 </header>
 <AuthSignature>
   <ds:SignedInfo>
     <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
     <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256">
     </ds:SignatureMethod>
     <ds:Reference URI="#xpointer(//*[@authenticate='true'])">
       <ds:Transforms>
         <ds:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
       </ds:Transforms>
       <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
       <ds:DigestValue>…here hash value authentication..</ds:DigestValue>
     </ds:Reference>
   </ds:SignedInfo>
   <ds:SignatureValue>…here signature value authentication..</ds:SignatureValue>
 </AuthSignature>
 <body>
   <DataTransfer>
     <OrderData>RUJJQ1MtUmVxdWVzdCBm/HIgZGllINxiZXJ0…</OrderData>
       </DataTransfer>
 </body>
</ebicsRequest>
```

Diagram 38: EBICS request for transmission of the last order data segment of a business transaction format order

- Transmission of the following data in the EBICS response (see also example in Diagram 39)

Bank-technical return code (`ebicsResponse/body/ReturnCode`)

Order number (`ebicsResponse/header/mutable/OrderId`)

Technical return code (`ebicsResponse/header/mutable/ReturnCode`)

Technical report text (`ebicsResponse/header/mutable/ReportText`)

Data for identification of a transaction step:
If the technical return code has the value EBICS_TX_RECOVERY_SYNC, this transaction step identifies the recovery point of the upload transaction. However, if neither technical nor specialist errors have occurred in this example, this transaction step reflects the current transaction step.

- Transaction ID (`ebicsResponse/header/static/TransactionID`)

- Transaction phase (`ebicsResponse/header/mutable/TransactionPhase`)
- (Conditional) Serial number of the order data segment
  (`ebicsResponse/header/mutable/SegmentNumber`), if the value of `TransactionPhase` is not equal to “Initialisation”.
  The attribute `ebicsResponse/header/mutable/SegmentNumber@lastSegment` specifies whether this is the last data segment.
- Identification and authentication signature of the financial institution
  (`ebicsResponse/AuthSignature`)

The identification and authentication signature includes all XML elements of the EBICS response whose attribute value for `@authenticate` is equal to “true”. The definition of the XML schema “ebics_response_H005.xsd“ guarantees that the value of the attribute `@authenticate` is equal to “true” for precisely those elements that also need to be signed.

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsResponse
 xmlns="urn:org:ebics:H005"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_response_H005.xsd"
 Version="H005" Revision="1">
 <header authenticate="true">
   <static>
     <TransactionID>ABCDEF4139464436445313243ABCDEF</TransactionID>
   </static>
   <mutable>
     <TransactionPhase>Transfer</TransactionPhase>
     <SegmentNumber lastSegment="true">4</SegmentNumber>
     <OrderId>OR01</OrderId>
     <ReturnCode>000000</ReturnCode>
     <ReportText>[EBICS_OK] OK</ReportText>
   </mutable>
 </header>
 <AuthSignature>
   <ds:SignedInfo>
     <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
     <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
     <ds:Reference URI="#xpointer(//*[@authenticate='true'])">
       <ds:Transforms>
         <ds:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
       </ds:Transforms>
       <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
       <ds:DigestValue>…here hash value authentication..</ds:DigestValue>
     </ds:Reference>
   </ds:SignedInfo>
   <ds:SignatureValue>…here signature value authentication..</ds:SignatureValue>
 </AuthSignature>
 <body>
   <ReturnCode authenticate="true">000000</ReturnCode>
 </body>
</ebicsResponse>
```

Diagram 39: EBICS response for transmission of the last order data segment for a business transaction form order

5.5.1.1.3 Upload Request Structure for Business Transaction Formats (BTF)

The following general rules define the interdependency of the BTF elements:

1) For the element <ServiceName> all usable codes are defined by the EBICS SC (external code list). They have by default a global meaning. An example for globally accepted services are e.g. SEPA services.
2) Services without a globally accepted implementation guide need to use <Scope>.
3) A global service with some additional specific (market or bilateral) modifications (e.g. namespace, specific assignments) need to apply an appropriate scope to identify this fact.
4) Usable scope codes are also specified in an external code list.
5) For the element <ServiceOption> the following structure is binding:

a) 3 characters: Global codes (External code list)
b) 4 characters: Codes defined by a market (External code lists).
The market can be identified by <Scope>-element.
c) 5 to 10 characters: Codes defined bilaterally between a single bank and its customer (publication not mandatory)
d) All ServiceOption code lists (global, market, (bilateral)) must include the applicable ServiceNames for each of the defined codes
e) The use of 4-10 character service options requires a scope element.
This means that only <ServiceName> without a <ServiceOption> or together with 3-character (i.e. globally agreed) ServiceOption code can be globally agreed business transactions.

To put it in a nutshell: <Scope> refers to the mandatory elements <ServiceName> and <MsgName> and – if present – to <ServiceOption>. This description is also valid for download orders, dateils see chapter 5.6.1.1.4).

The values of the order parameters are positioned in
`ebicsRequest/header/static/OrderDetails`
of the type `BTUOrderParams` which is usable in the upload direction and with order type “BTU”:

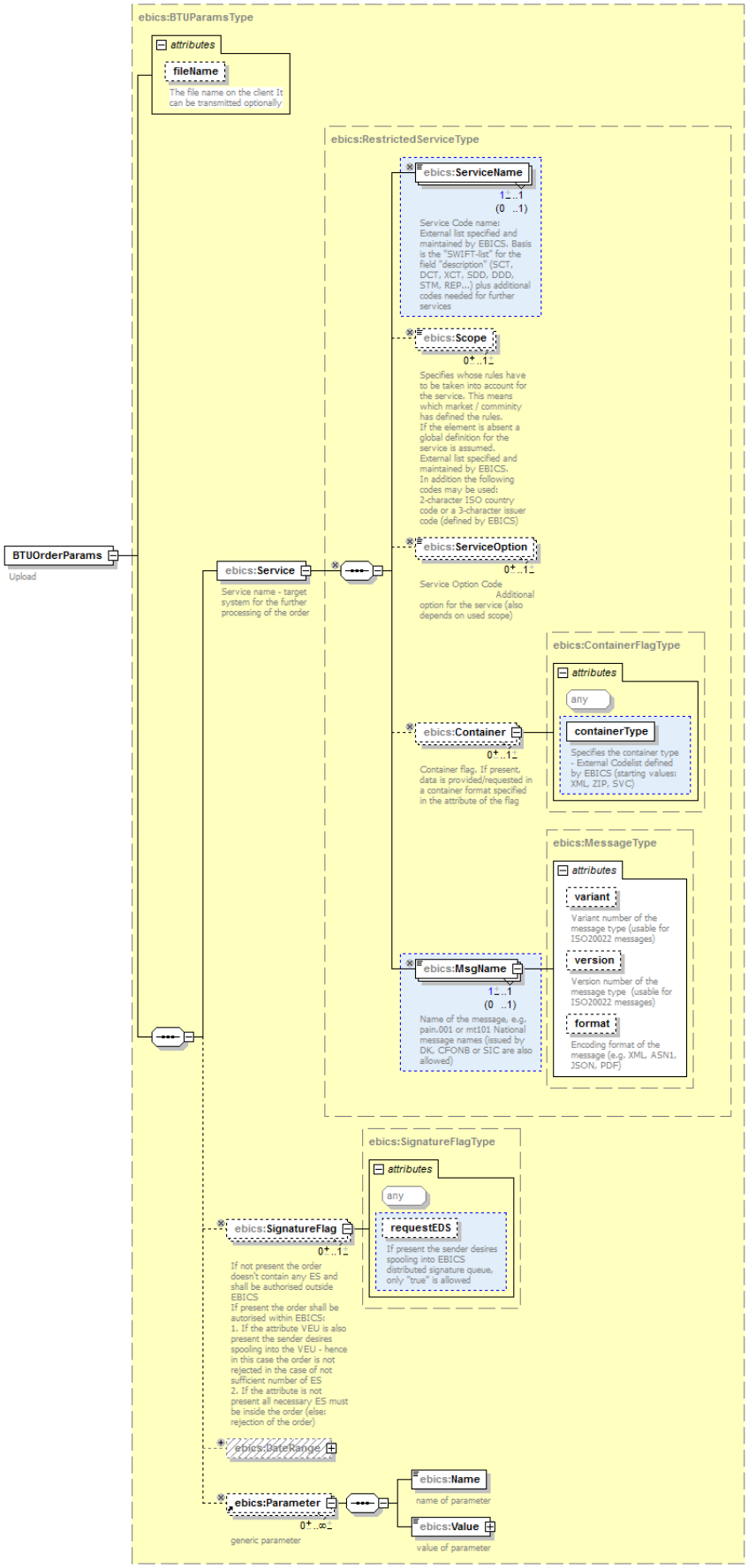


Diagram 40: BTF structure for upload (using restricted service type)

XML element/ attribute	Data type	#	Meaning	Example
BTUOrderParams	ebics:BTUParamsType (complex)	1	Required to qualify an upload business transaction and the used/corresponding format	- (complex)
BTUOrderParams @FileNm	ebics:FileNameStringType (simple) - restriction base: minLength value="1" maxLength value="256"	0..1	Original file name of the file on the client system	
Service	RestrictedServiceType (complex)	1	Indicates the target system (nature)/process to handle the Transaction/File	- (complex)
ServiceName	ebics: ServiceNameStringType (simple) - restriction base: minLength value="3" maxLength value="3" pattern = [A-Z0-9]	1	Name of the service	ServiceName is subject to an external code list (maintained by EBICS) - Example: “SCT” = SEPA credit transfer
Scope	ebics: ScopeStringType (simple) restriction base: minLength value="2" maxLength value="3"	0..1	Specifies whose rules have to be taken into account for the service. This means which market / community defined the rules. External scope name list specified and maintained by EBICS. A missing scope element means globally accepted rules.	Scope is subject to an external code list (maintained by EBICS). 2-char country codes 3-char codes for other scopes “BIL” means bilaterally agreed The meaning of a missing Scope element is global. Instead of a missing scope element it can

				also be provided as code “GLB”.
ServiceOption	ebics:CodeStringType (simple) restriction base: minLength value="3" maxLength value="10" pattern = [A-Z0-9]	0..1	Optional characteristic(s) of a service	ServiceOption is subject to external code lists (global, market, bilateral) Example: “URG” = urgent
Container	ebics:ContainerFlagType	0..1	Flag to indicate the use of a container	Only value true allowed. No presence means false
Container@containerType	ebics:ContainerstringType (simple) restriction base: minLength value="3" maxLength value="3" pattern = [A-Z0-9]	1..1	Indicates what type of container is used.	If the container Flag is present, one of the internal values has to be used (internal code list) “XML” “ZIP” “SVC”
MsgName	ebics: MessageNameStringType (simple) restriction base: minLength value="1" maxLength value="10" pattern = [a-z\.0-9]	1	message names starting with a BA code (ISO) or MT (FIN) or string to be evaluated	“pain.001”, “mt103” Message names (issued by markets, specified in “scope”) are also allowed
MsgName@version	ebics:NumString (simple) restriction base: minLength value="2" maxLength value="2" pattern = [0-9]	0..1	Used ISO version of message, ignored if no ISO message name	“03”
MsgName@variant	ebics:NumString (simple) restriction base: minLength value="3"	0..1	Evaluated together with <MsgName>, ignored if no ISO message name	“001”

	maxLength value="3" pattern = [0-9]			
MsgName@format	ebics:CodeString (simple) restriction base: minLength value="1" maxLength value="4" pattern = [A-Z0-9]	0..1	Evaluated together with <MsgName>, admissible for each kind of message name, but only to be used if it is not the standard format for the used message standard (especially non-XML for ISO 20022).	„XML“, „ASN1“, „JSON“, „PDF“
SignatureFlag	SignatureFlagType	0..1	Flag to indicate the presence of ES' (see rules in combination with requestEDS Flag)	Only value `true` allowed. No presence means `false`
SignatureFlag@requestEDS	boolean	0..1	If present the order shall be authorized within EBICS (Details regarding bank server reactions refer to chapter 3.14)	Only value `true` allowed. No presence means `false`
Parameter		0..1	Generic oder params: Any number of Name-Value pairs can be specified	
Name	boolean	1..1	Name of parameter	
Value	anySimpleType	1..1	Value of parameter	
Value@type	NCName	1..1	Type of value	Recommen-dation for a default is `string`

5.5.1.2 Processing of EBICS messages

Chapter 5.5.1.1 describes the contents of the EBICS messages that are exchanged within the framework of an upload transaction. The subject of this chapter is the processing of these EBICS messages at the bank's end. Action sequences are pointed out in the flow diagram in Diagram 34, the course of which is described here in greater detail.

In order to simplify the description of the processes, it is assumed that every processing step produces a return code (RC) whose value is equal to 0 ("000000", EBICS_OK) if it has been possible to successfully complete this step. The technical return code (RCT) and the bank-technical return code (RCF) are set depending on the RC, and their values then flow into the EBICS messages.

The validity of the EBICS request is verified on the basis of the XML schema definition file "ebics_request_H005.xsd", and with due regard to the restrictions that have been specified for the individual requests in Chapter 5.5.1.1. The validity verification usually takes place in parallel and/or interlocked with the other process steps in processing the EBICS request. The following processes dispense with representation of a process step of type "EBICS

request validity verification" in favour of the simplest possible representation. In consequence, these processes can be terminated by the following additional technical errors:

- EBICS_INVALID_XML
  The received EBICS XML message does not conform to the specifications of the XML schema in view of syntax. The XML code is not well-formed or, according to the schema, not valid. For example, if the upload request does not contain the element `HostId` (that the schema requires).

- EBICS_INVALID_REQUEST
  The received EBICS XML message does not conform to the EBICS specifications in view of syntax, for example, if the upload request does not contain the element `NumSegments` (which is optional according to the XML schema, but required according to chapter 5.5.1.1.1).

- EBICS_INVALID_REQUEST_CONTENT
  The received EBICS XML message does not conform to the EBICS specifications in view of semantics although being correct according to the schema.

5.5.1.2.1 Processing in the initialisation phase

Diagram 44 shows processing at the bank's end of the EBICS request which is transferred from the customer system to the bank system in the initialisation stage of an upload transaction. The individual processing steps are explained in greater detail in the following text:

I. Generation of an EBICS transaction (see Diagram 43)

This processing step is relevant for both upload and download transactions. The following description takes both transaction types into consideration so that the following chapters on the subject of download transactions will be able to refer to this description.

I.a. Verifying the identifier for the business transaction (and administrative order as well)

Verification of the order type returns the technical return code EBICS_INVALID_ORDER_IDENTIFIER in the case of an invalid administrative order type or rather an invalid combination of BTF identifiers, or the technical return code EBICS_UNSUPPORTED_ORDER_IDENTIFIER in the case of a valid but optional order that is not supported by the bank system.

I.b. Replay test

The replay test returns the following return code EBICS_TX_MESSAGE_REPLAY if the EBICS request is a replayed request. Details on replay avoidance can be found in the Appendix (Chapter 11.4).

I.c. Verifying the authenticity of the EBICS request (see Diagram 41):

The identification and authentication signature is provided by a technical subscriber, if such is a component of the control data. Otherwise the identification and authentication signature is generated by a (non-technical) subscriber of the EBICS transaction who submits the order or, subsequently, bank-technical ES's that relate to an existing order. In order to be able to verify the identification and authentication signature of a subscriber (technical or non-technical), the corresponding combination of customer and subscriber ID must be registered in the bank system and the state of the subscriber must be set to "Ready". In error situations that result from an invalid combination of customer ID / subscriber ID or an inadmissible subscriber state, the sender receives the technical return code EBICS_AUTHENTICATION_FAILED.

Verification of the identification and authentication signature contains:

- A verification as to whether all required elements of the EBICS message have been signed with the identification and authentication signature: These are all XML elements of the EBICS request whose attribute value for `@authenticate` is equal to "true".
- Verification of the identification and authentication signature itself.

This processing step terminates with the technical error EBICS_AUTHENTICATION_FAILED if the identification and authentication signature cannot be successfully verified.

If the successfully-verified signature originates from a technical subscriber, the validity and the state of the (non-technical) subscriber is also verified. Errors that result from an invalid combination of customer ID/ subscriber ID or an inadmissible subscriber state are communicated to the sender of the EBICS request with the help of the technical error codes EBICS_USER_UNKNOWN and EBICS_INVALID_USER_STATE.
Reason: If the identification and authentication signature cannot be successfully verified, the EBICS request potentially originates from an attacker. In this event, errors such as "Unknown subscriber" or "Inadmissible subscriber state" are not forwarded to the sender of the EBICS request so that potential attackers are not given precise information on the validity of subscriber IDs or the state of subscribers. However, after the identification and authentication signature of the technical subscriber has been successfully verified, the errors EBICS_USER_UNKNOWN and EBICS_INVALID_USER_STATE, which relate to the non-technical subscriber of the EBICS transaction, are forwarded to the authenticated sender.

I.d. Verifying the hash value of the bank keys

This verification is intended to prevent a subscriber from submitting orders when they are not in possession of the financial institution's current public keys. In Version "H005" of EBICS the ES of the financial institutions is only planned (see Chapter 3.5.2). For this reason, only the hash values of the public identification and authentication key and the public encryption key are verified in Version "H005". In this processing step, subsequent EBICS versions that support the financial institution's ES must also verify the hash value of

the financial institution's public bank-technical key. For this reason, the subscriber transfers the hash values of the financial institution's public key with which they have been provided. The bank system verifies these hash values. If they do not match the hash values of the current public keys, transaction initialisation is terminated with the technical return code EBICS_BANK_PUBKEY_UPDATE_REQUIRED.

If the subscriber does not have the financial institution's current identification and authentication they cannot successfully verify the identification and authentication signature of the financial institution's EBICS response. Nevertheless, when the error EBICS_BANK_PUBKEY_UPDATE_REQUIRED occurs it should be verified as to whether the bank keys are up-to-date, and if necessary the latest keys should be downloaded with the help of the administrative order type HPB.

I.e. Subscriber-related order verifications (see Diagram 42)

I.e.a. Verifying authorisation for the business transaction (and administrative order as well)

This verifies as to whether the subscriber is entitled to submit the order (administrative order type or business transaction format) in question. If this verification fails, transaction initialisation is terminated with the business related error EBICS_AUTHORISATION_ORDER_FAILED.

In the case of upload orders, order type authorisation is successful if the subscriber has at least ES authorisation of class "T" for the order in question.

Note: The ES authorisation of the actual signatory of the order is not verified here. This verification is a part of the (optional) preliminary verification of an order.

In the case of download orders, the order authorisation is not coupled to an ES authorisation. It is verified as to whether the subscriber is authorised for the order type in question.

I.e.b. Bank-technical preliminary verification

This verification only affects upload orders, details of the preliminary verification are given in Chapter 5.3.

If the optional preliminary verification of orders is principally not supported by the financial institution, but the EBICS request contains data for preliminary verification of the order, the information EBICS_NO_ONLINE_CHECKS is returned.

This technical information has no influence on the ongoing transaction. The order is continued.

The bank-technical preliminary verification of an upload order returns the following business related return codes in the event of an error:

- EBICS_SIGNATURE_VERIFICATION_FAILED
  This business related error occurs when the ES of the order signatory could not be successfully verified

- EBICS_INVALID_SIGNATURE_FILE_FORMAT
  The submitted ES data do not conform to the specified format.

- EBICS_ PARTNER_ID_MISMATCH
  The partner ID (=customer ID) of the ES file differs from the partner ID (=customer ID) of the submitter.

- EBICS_ACCOUNT_AUTHORISATION_FAILED
  This business related error code is returned when the account authorisation verification fails for one of the signatories

- EBICS_AMOUNT_CHECK_FAILED
  This business related error occurs when the limit verification fails for one of the signatories

- EBICS_SIGNER_UNKNOWN
  This business related error occurs when one of the signatories is not a valid subscriber

- EBICS_INVALID_SIGNER_STATE
  This business related error occurs when the state of one of the signatories is not equal to "Ready".

- For Return codes relating to CA-issued certificates, refer to Annex 1.

I.e.c. Order number verification

The following verifications only relate to business related upload orders:

1) It is only permitted to send business related orders without an order number.
2) Files which only contain an ES but no further data are not allowed except for the administrative order types HVE and HVS (and SPR as well).

3) For HVE and HVS please note that the aforesaid refers to the order number of HVE itself (and HVS, respectively) and not to the related order in the EDS. It is self-evident that this (related) order number has to be always transmitted (via OrderParams).

"

Hence the possible error codes for upload requests are:

- For an SPR (and HVE and HVS, respectively) upload request which is submitted with an order number, the code EBICS_INVALID_REQUEST_CONTENT is returned

- For the administrative order types HVE or HVS submit an unknown order number, the code EBICS_ORDERID_UNKNOWN is returned

- For an HVE- or HVS upload request which is submitted with an already assigned order which has, however, an invalid processing state (because the order has already been fully authorized or rejected) the code EBICS_ORDERID_ALREADY_FINAL is returned.

Rule for an upload response: Every upload response contains TransactionID and OrderId assigned by the server (also in the case of an error)

I.f. Generation of a new EBICS transaction with unambiguous transaction ID

When all of the previous verifications have been successfully carried out and more transaction steps follow, the EBICS transaction administration generates a new EBICS transaction at the bank's end with a transaction ID that is unambiguous throughout the bank

system. Details on generation of the transaction ID are given in the Appendix (Chapter 11.6).

II. Pre-processing

Here, pre-processing relates to the transmitted ES's and the order parameters of all orders considered in this chapter except the administrative order types HVE and HVS. Pre-processing is not a component of the EBICS specification and is thus dependent on bank system implementation. For example, intermediate storage of ES's and order parameters is a part of this pre-processing.

III. Forwarding to managment of pending orders

For the administrative order types HVE and HVS the the EBICS transaction comprises a single request/response pair. In this case the transmitted order parameters and ES's are forwarded directly to the management of pending orders and the transaction is terminated. The component 'management of pending orders' is not a part of the EBICS standard.

IV. Generation of the EBICS response

This processing step generates the EBICS response that is afterwards sent to the customer system. In the event of an error, this EBICS message contains the corresponding technical or business related error code of preceding process steps. The contents of this EBICS message are described in greater detail in Chapter 5.5.1.1.1.

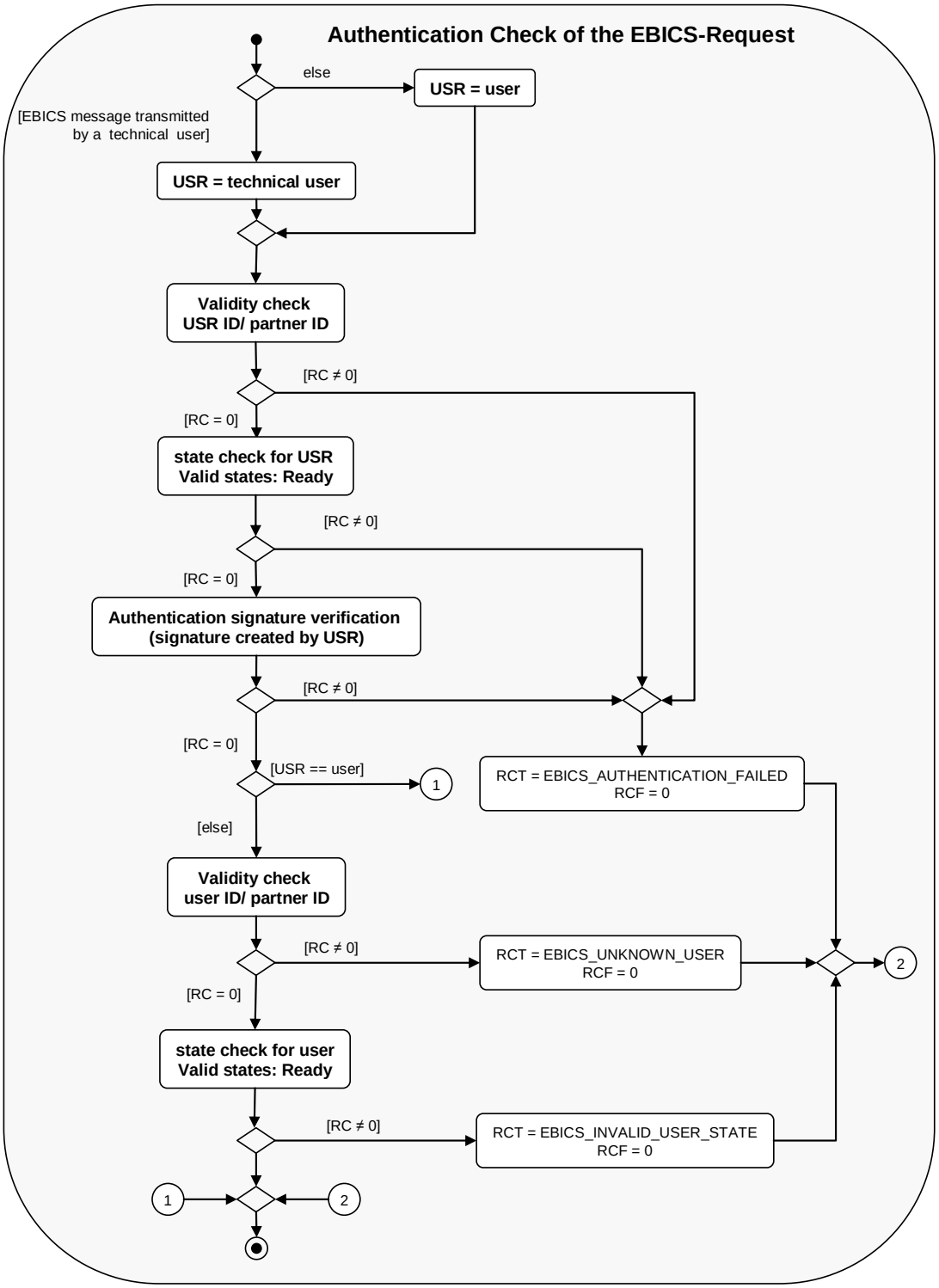


Diagram 41: Detailed description of the process step "Authentication check of the EBICS request"

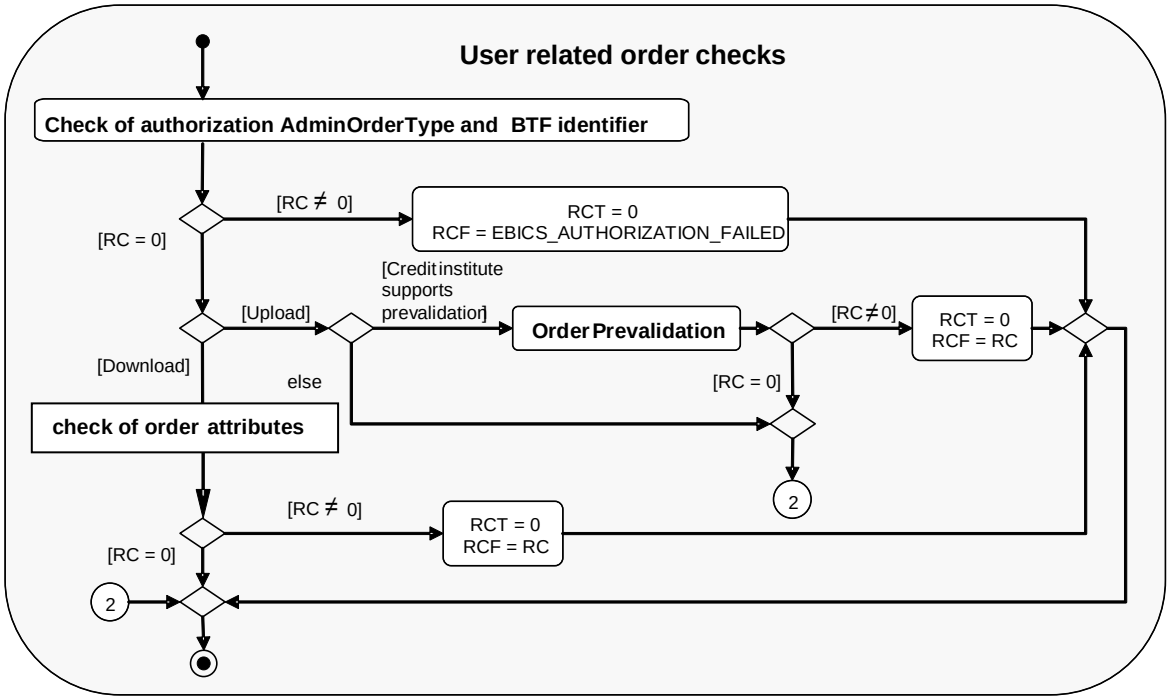


Diagram 42: Detailed description of the process step "User related order checks"

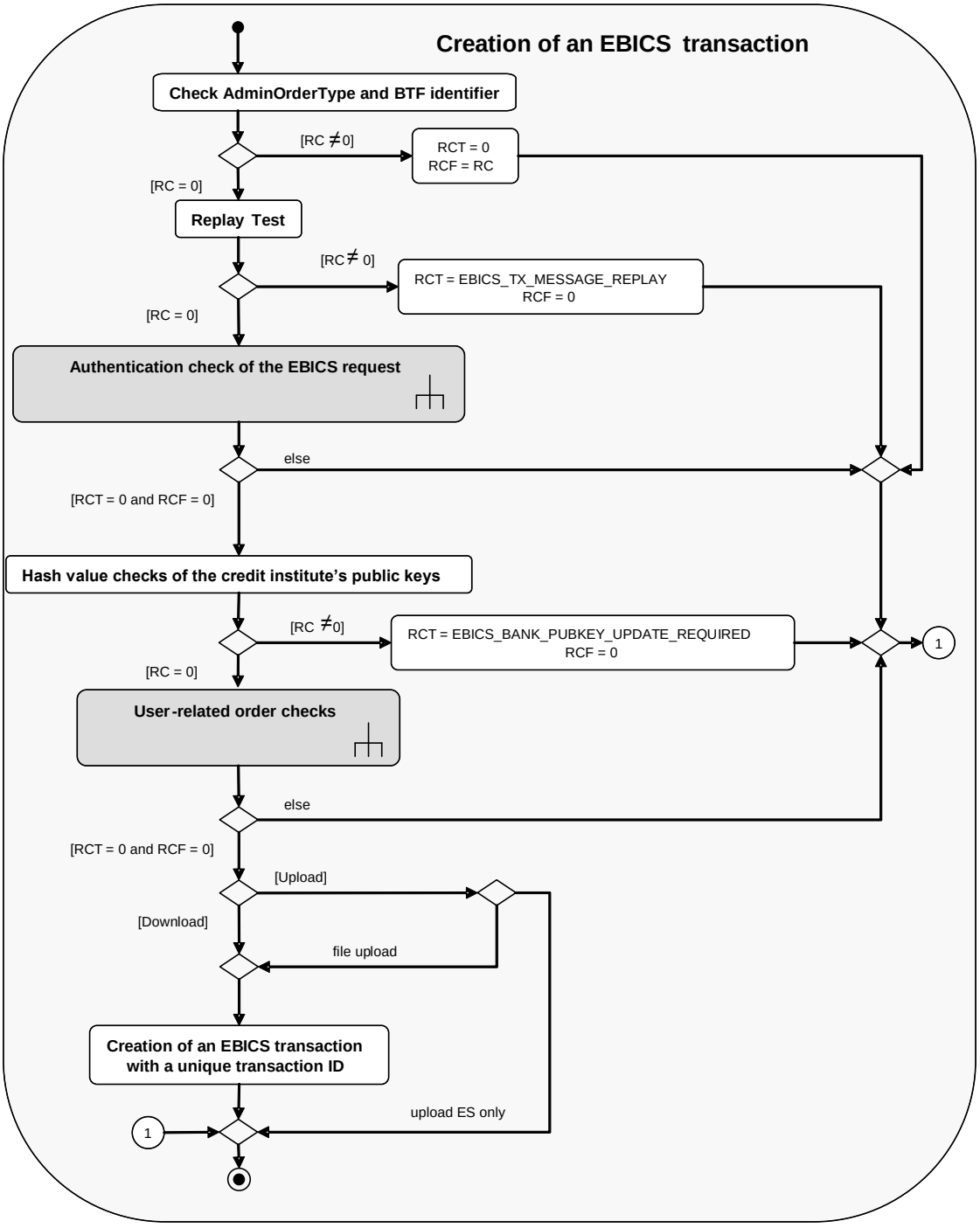


Diagram 43: Detailed description of the process step "Creation of an EBICS transaction"

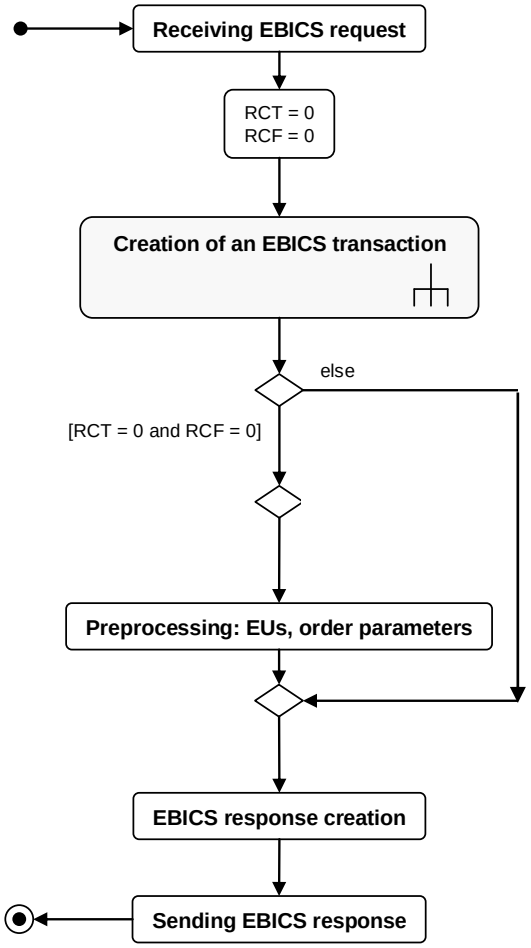


Diagram 44: Processing the EBICS request from transaction initialisation

5.5.1.2.2 Processing in the data transfer phase

The processing at the bank's end of the EBICS request that is transmitted in the data transfer phase from the customer's system to the bank's system is represented in Diagram 46 and Diagram 47. The individual processing steps are explained in greater detail in the following text:

I. Verifying the EBICS transaction (see Diagram 45)
This processing step is relevant for both upload and download transactions. The following description takes both transaction types into consideration so that the following chapters on the subject of download transactions will be able to refer to this description.

I.a. Verifying the transaction ID

A verification is carried out as to whether the EBICS transaction with the corresponding ID exists as an open, not yet completed, transaction in the bank system's EBICS transaction administration system. If this is not the case, the technical error code EBICS_TX_UNKNOWN_TXID is returned to the sender of the EBICS request.

I b. Verifying the authenticity of the EBICS request (see Diagram 41)
This EBICS request authenticity verification takes place in exactly the same way as in the

initialisation phase of the transaction (see Chapter 5.5.1.2.1, I.c) – apart from the fact that the required data (e.g. customer/subscriber ID) is not part of the header data of the EBICS request but is stored in the financial institution's transaction administration with the transaction ID in question. If the verification cannot be carried out successfully the EBICS response contains a corresponding error code in accordance with the sequence shown in Diagram 41. This is one of the errors EBICS_AUTHENTICATION_FAILED, EBICS_USER_UNKNOWN or EBICS_INVALID_USER_STATE. Unauthenticated requests do not have any effect on the state of the transaction in the bank system's transaction administration. Data that has an effect on the state of a transaction such as e.g. the next expected transaction step or the current recovery counter, is not changed. This prevents attackers from being able to have any effect on a transaction with the help of unauthenticated EBICS requests. The transaction can be continued by the subscriber as if the EBICS request with the invalid identification and authentication signature had not been received.

I c. Verifying TxPhase/ TxStep from the EBICS request

At this point, a verification is carried out as to whether the transaction step from the EBICS request matches the current state of the EBICS transaction in the bank system if one assumes a specific sequential order for the transaction steps.

In the case of an upload transaction, the sequential order according to Diagram 34 is assumed. Verificationing of the transaction phase / transaction step is successful when:

- The last transaction step initialised by the subscriber has been successfully completed, i.e. initialisation and transmission of the n^{th} data segment was successful.
- The transaction step from the EBICS request is the next transaction step in the sequential order of transaction steps, i.e. it is the transmission of the 1^{st} or the $(n+1)^{th}$ data segment.

The normal sequential order of transaction steps of a download transaction is shown in Diagram 51. The transaction phase / transaction step is deemed to have been successfully verified when the following two conditions are met:

- The last transaction step initiated by the subscriber has been successfully implemented, i.e. the initialisation (and hence transmission of the first data segment, or the request of the n^{th} data segment within the framework of the data transfer were successful.
- The transaction step from the EBICS request is the next transaction step in the sequential order of the transaction steps, i.e. it is the request for the $(n+1)^{th}$ data segment or acknowledgement of the downloaded data where n represents the last data segment.

II. Evaluation of the EBICS transaction verification results

If the transaction step verification was unsuccessful, then:

- A verification is carried out as to whether the upload transaction can be recovered, if the bank system supports the recovery of transactions. This verification is carried out in accordance with the description in Chapter 5.5.2. The technical error code EBICS_TX_RECOVERY_SYNC is returned if the transaction can be recovered, otherwise the transaction is terminated with the technical error code EBICS_TX_ABORT.
- The upload transaction is terminated with the business related error code EBICS_RECOVERY_NOT_SUPPORTED if the bank system does not support transaction recovery. If MAX is set to 0, the flow diagram also considers the case where recovery is not supported.

III. Verifying segment number and segment size

The serial number of the transmitted order data segment (`ebicsRequest/header/mutable/SegmentNumber`) must be less than or equal to the total number of data segments that are to be transmitted. If the number of transmitted order data segments matches the total number, the value of attribute `ebicsRequest/header/mutable/SegmentNumber@lastSegment` must also be equal to “true”. If one of these two conditions is not fulfilled, the transaction is terminated with the technical error code EBICS_TX_SEGMENT_NUMBER_EXCEEDED.

If the serial number of the transmitted order data segment is less than the total number of the order data segments that are to be transmitted and the value of attribute `ebicsRequest/header/mutable/SegmentNumber@lastSegment` is nevertheless “true”, then technical return code EBICS_TX_SEGMENT_NUMBER_UNDERRUN of error class “Note” is returned.

The size of the transmitted order data segment may not exceed the segment size of 1 MB that has been firmly specified for EBICS “H005”. Otherwise the transaction is terminated with the technical error code EBICS_SEGMENT_SIZE_EXCEEDED.

IV. Pre-processing

Here, pre-processing relates to the transmitted order data segment. Pre-processing of order data segments is not part of the EBICS specification. It is dependent on the bank system implementation, intermediate storage of the order data segment may be a part of pre-processing.

V. Forwarding to management of pending orders

If the transmitted order data segment was the last one, and the matter at hand is a bank-technical upload order, all of the order parameters, ES’s and order data transmitted within the framework of the EBICS transaction are forwarded to the management of pending orders. Following this, the EBICS transaction can be terminated.
The component ‘management of pending orders’ is not a part of the EBICS standard.

(Optional) check for double upload : As the data digest of the order data is transmitted within the transfer phase the bank server has the option to control by means of the data digest if the order has already been transmitted (if necessary also by taking other order information into consideration). If the order already exists on the server (same data digest was already transmitted recently), the request will be rejected by the return code EBICS_ORDER_ALREADY_EXISTS.

Note: Checks basing on this data digest are optional as double uploads can also be supervised by taking other action within the implementation.

VI. Verifying and implementing the order

If the transmitted order data segment was the last one, and if the order is a system-related upload order, it is synchronously verified and implemented on the basis of the transmitted order data. The returned technical or business related error codes are dependent on the order type and are defined in the chapters in which these order types are described.

VII. Generation of the EBICS response

This processing step generates the EBICS response that is afterwards sent to the customer system. In the event of an error, this EBICS message contains the corresponding technical / business related error code of the preceding process steps. The contents of this EBICS message are described in greater detail in Chapter 5.5.1.1.2.

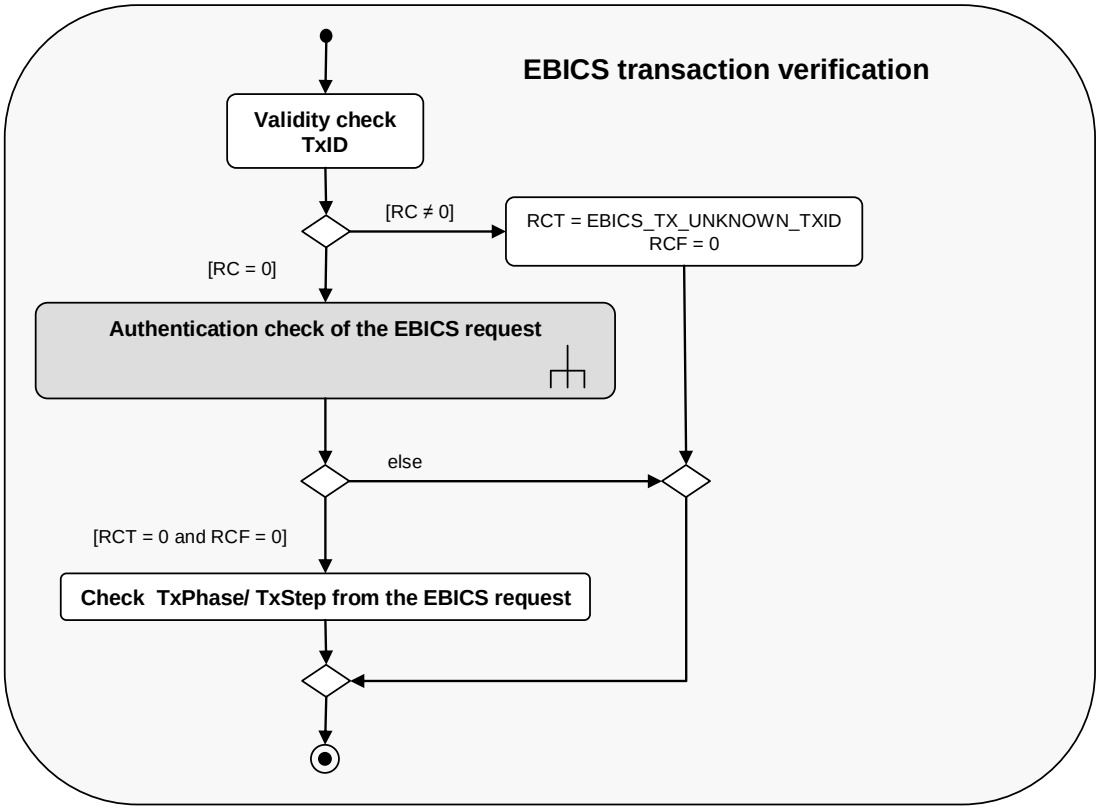


Diagram 45: Detailed description of the process step "EBICS transaction verification"

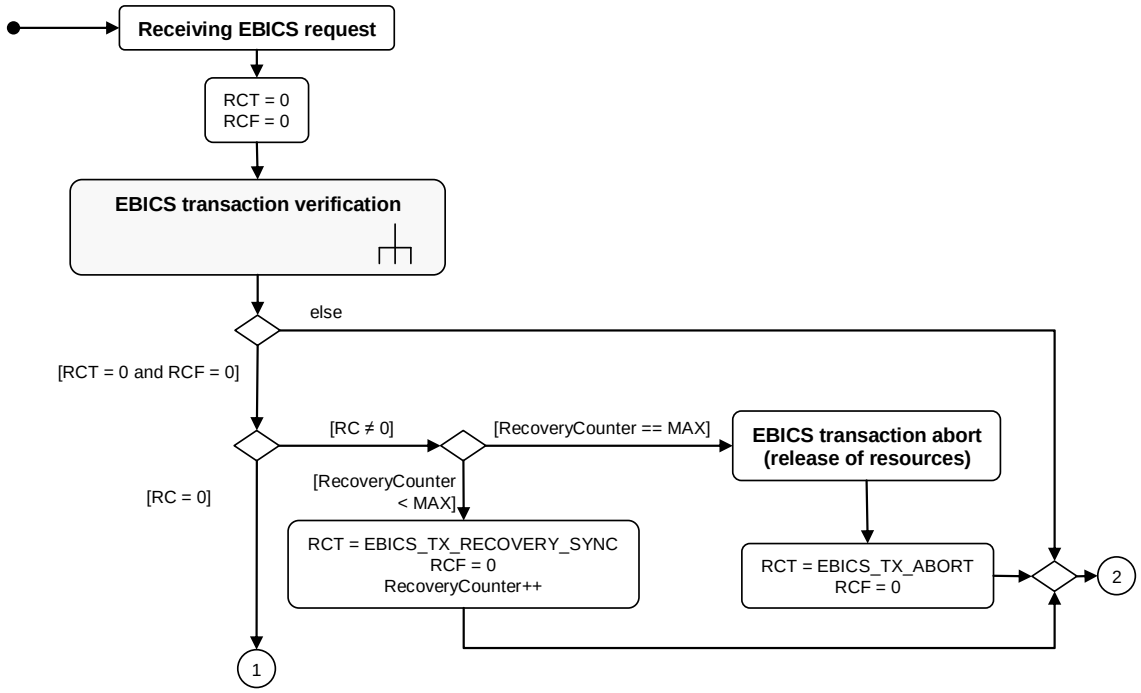


Diagram 46: Processing an EBICS request for transmission of an order data segment (part 1)

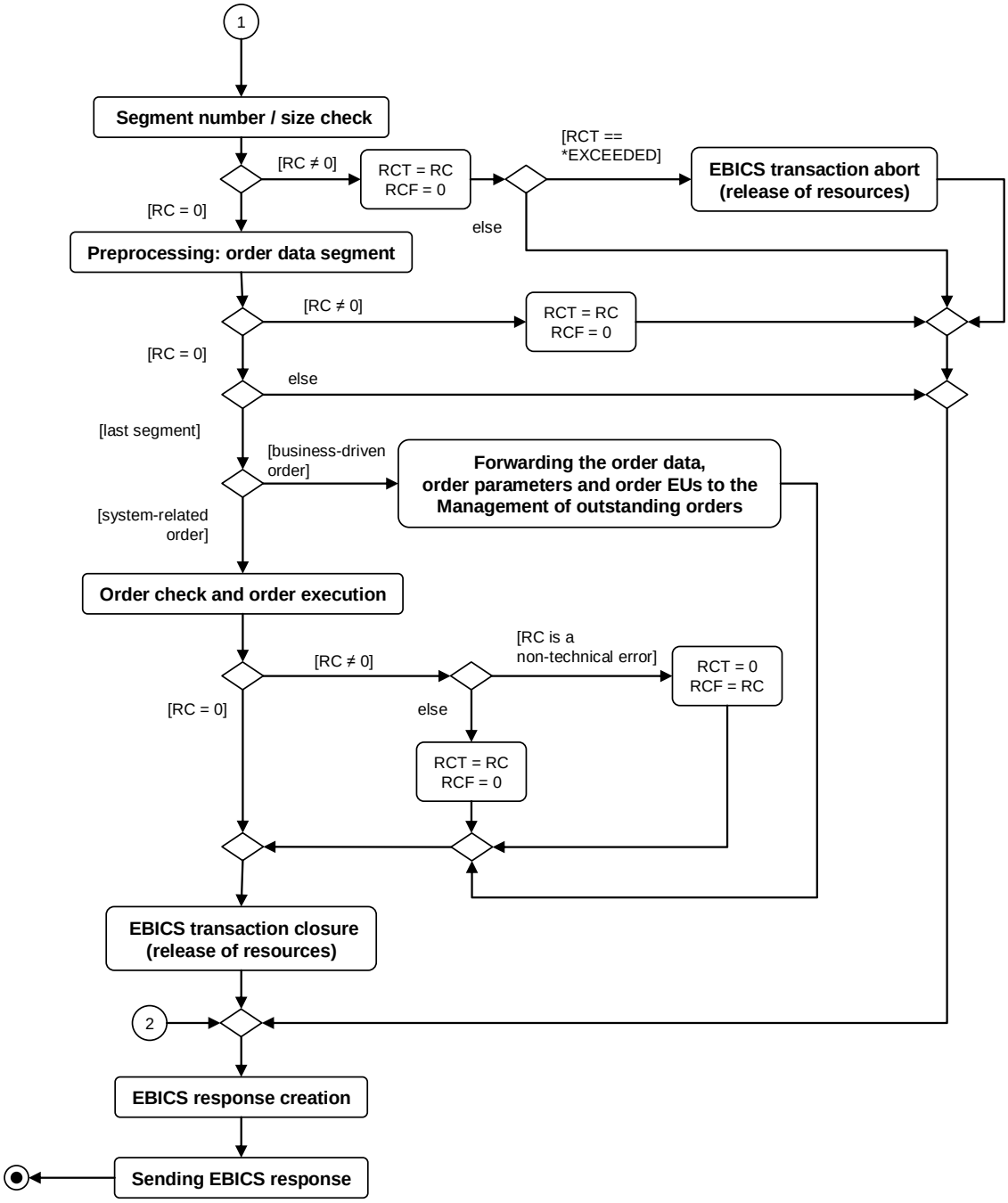


Diagram 47: Processing an EBICS request for transmission of an order data segment (part 2)

5.5.2 Recovery of upload transactions

The customer system can initiate the recovery mechanism when one of the following error situations occurs:

- Transport error during transmission of an EBICS request in the data transfer phase of the transaction
- Timeout or transport error when receiving an EBICS transaction in the data transfer phase of the transaction
- Loss of the transaction state at the subscriber's end.

Incorrect processing of an EBICS request at the bank's end during the data transfer phase, caused by e.g. errors in the pre-processing of a transmitted order data segment, may require renewed transmission of this request. This is a special recovery case, since the customer system does not recognise the necessity for repetition of the transmission without further action. This special case can be dealt with by the EBICS recovery mechanism.

EBICS uses an optimistic approach when recovering an upload transaction and dispenses with a separate synchronisation step with the bank system. If one of the above error situations occurs, the customer system initially assumes knowledge of the transaction's recovery point due to the transaction data stored (possibly in a sustained manner) in the customer system.

If the customer system assumes that the recovery point is the transmission of the n^{th} order data segment, then the next initiated transaction step is transmission of the $(n+1)^{th}$ order data segment. EBICS requests within the framework of the recovery of upload transactions do not differ from the EBICS request of a normal, error-free flow of an upload transaction.

By way of example, the flow of a transaction that repeatedly necessitates recovery is shown in Diagram 48. In each case, the recovery takes place without explicit synchronisation between the customer system and the bank system. The 2^{nd} order data segment is transmitted three times since the customer system could not receive the corresponding EBICS response due to a timeout or a transport error. On the second and third transmission of the 2^{nd} order data segment, the customer system assumes that the recovery point is transmission of the 1^{st} order data segment. The value of the recovery counter is equal to 2 after the third and successful transmission of the 2^{nd} order data segment, since the last two transmissions of the 2^{nd} order data segment were evaluated as recovery attempts by the bank system. The transaction finally fails due to the number of recovery attempts being too high.

If the assumption regarding the recovery point is false, the EBICS response for transmission of the $(n+1)^{th}$ data segment receives the actual recovery point of the transaction in addition to the technical return code EBICS_TX_RECOVERY_SYNC. For example, if this recovery point is the transmission of order data segment k, the transaction can easily be resumed after this synchronisation with transmission of segments k+1, k+2, etc.

Diagram 49 shows the successful flow of a transaction that contains a recovery of the transaction after an explicit synchronisation between the customer system and the bank system. Here, the customer system transmits order data segment 1 in a state in which the bank system actually expects segment 3. The financial institution's EBICS response (see Diagram 50) thus contains the recovery point of the transaction, which in this case is transmission of the 2nd order data segment. Following this, the customer system continues with transmission of order data segment 3 and ends the transaction with the transmission of the last segment 4.

Independent of whether a customer system detects errors in the flow of a transaction, the bank system can force renewed transmission of an EBICS request. Analogously to the above recovery situations, this is achieved by the associated EBICS response containing the technical return code EBICS_TX_RECOVERY_SYNC as well as the recovery point of the transaction.

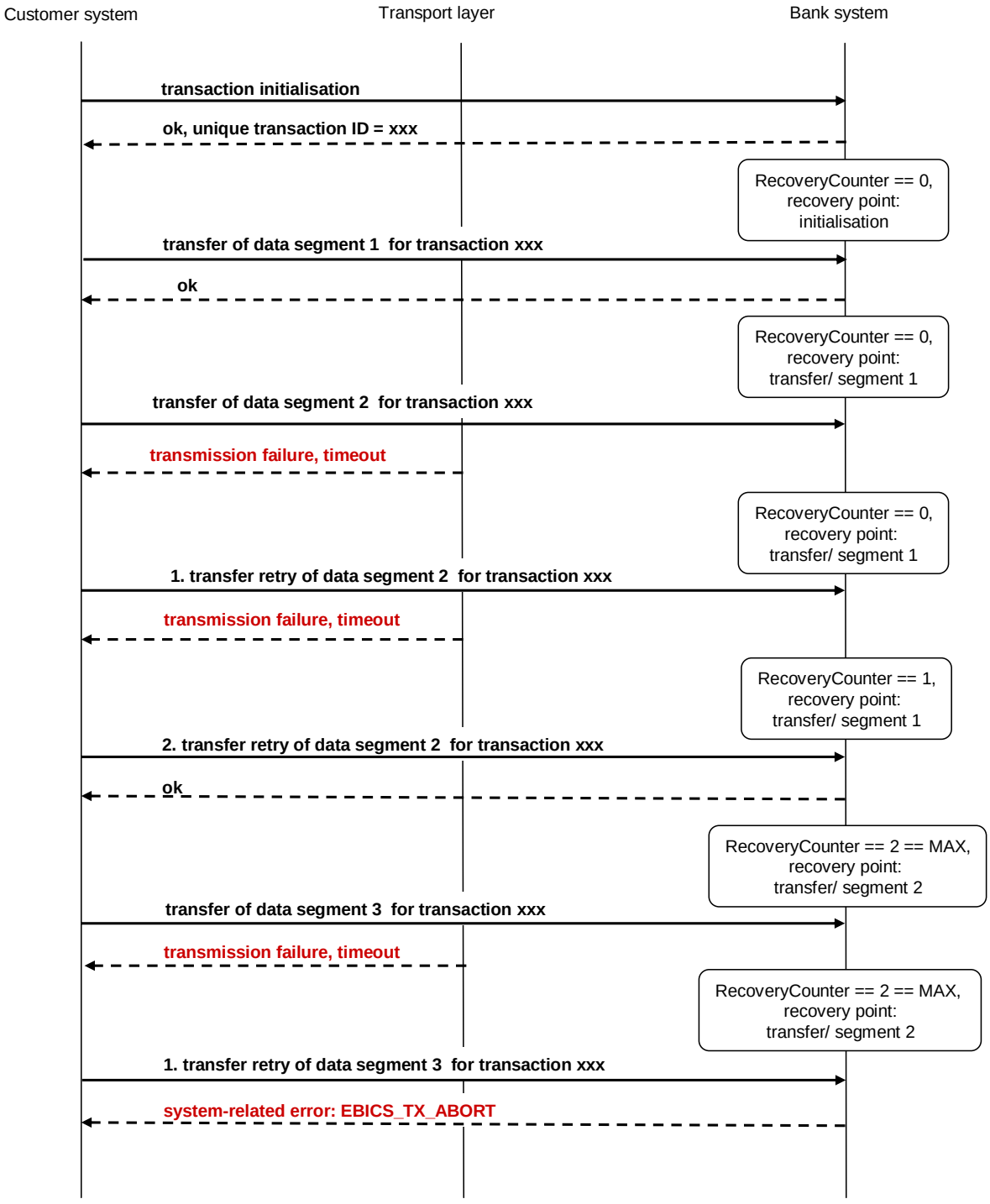


Diagram 48: Termination of the recovery of an upload transaction due to the maximum number of recovery attempts being exceeded

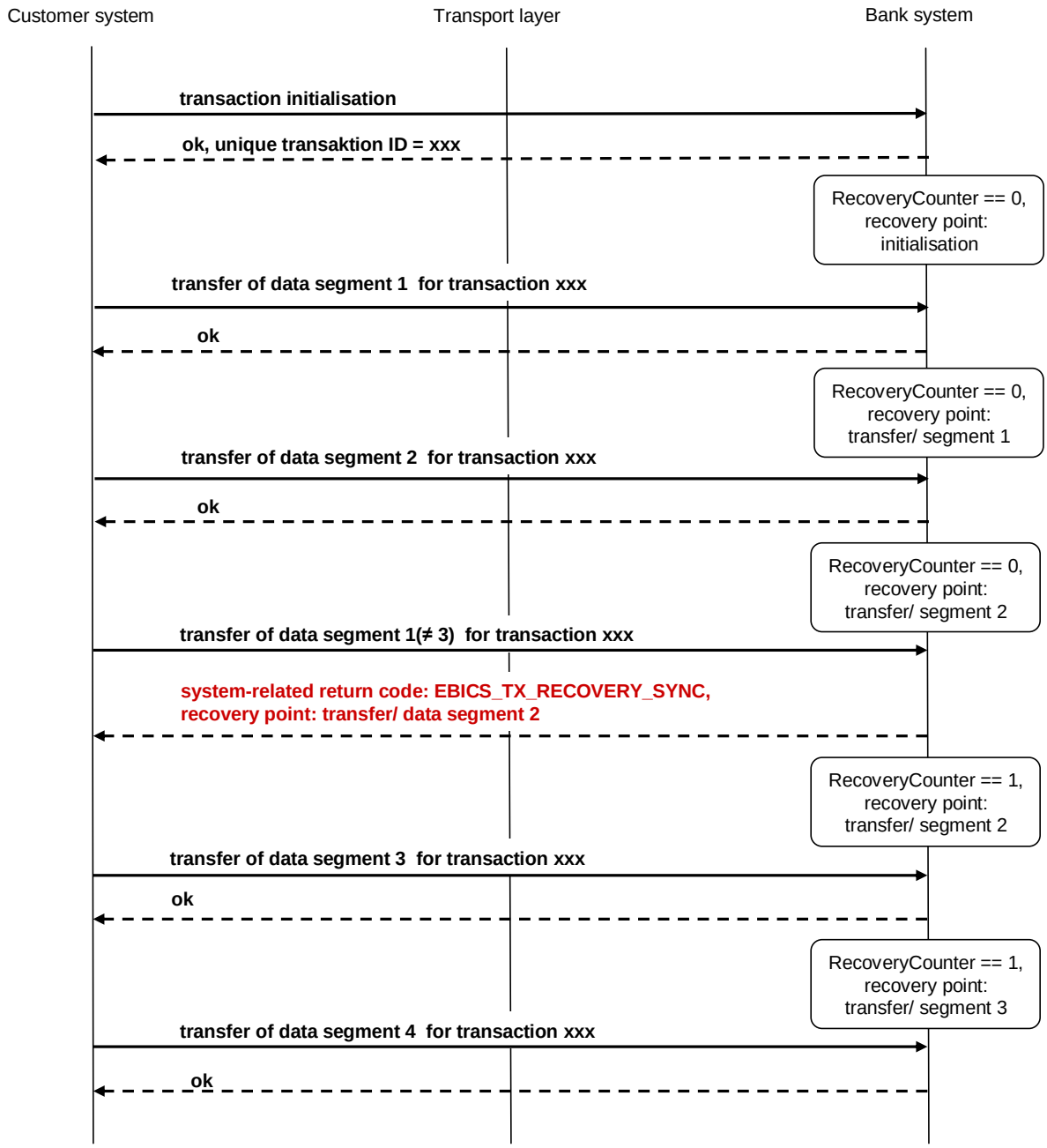


Diagram 49: Recovery of an upload transaction with explicit synchronisation between customer system and bank system

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsResponse
 xmlns="urn:org:ebics:H005"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_response_H005.xsd"
 Version="H005" Revision="1">
 <header authenticate="true">
   <static>
     <TransactionID>ABCDEF413946443634453132 43ABCDEF</TransactionID>
   </static>
   <mutable>
     <TransactionPhase>Transfer</TransactionPhase>
```

```
      <SegmentNumber lastSegment="false">2</SegmentNumber>
      <OrderId>OR01</OrderId>
      <ReturnCode>061101</ReturnCode>
      <ReportText>[EBICS_TX_RECOVERY_SYNC] Synchronisation necessary</ReportText>
    </mutable>
  </header>
  <AuthSignature>
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
      <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
      <ds:Reference URI="#xpointer(//*[@authenticate='true'])">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
        <ds:DigestValue>.. here hash value authentication ..</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>.. here siganture value authentication ..</ds:SignatureValue>
  </AuthSignature>
  <body>
    <ReturnCode authenticate="true">000000</ReturnCode>
  </body>
</ebicsResponse>
```

Diagram 50: EBICS response with technical error EBICS_TX_RECOVERY_SYNC

5.6 Download transactions

5.6.1 Sequence of download transactions

The sequence of a download transaction is shown in Diagram 51 by means of a flow diagram. This sequence diagram shows the exchange of EBICS messages in the individual phases of a download transaction. The first order data segment is contained in the EBICS response of the transaction initialisation. All other order data segments are transmitted in a loop that breaks off as soon as the last data segment has been received by the customer system. (see loop break-off condition “[last data segment has been received]”). Finally, the successful receipt of all order data segments is acknowledged by the customer system.

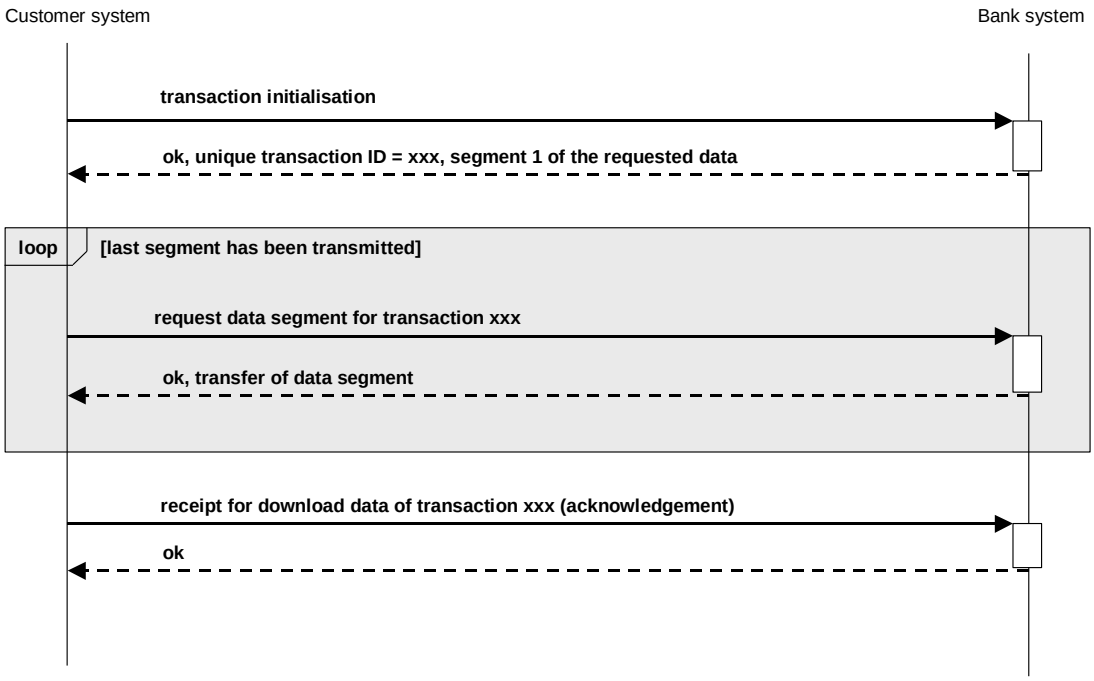


Diagram 51: Error-free sequence of a download transaction

5.6.1.1 Description of EBICS messages

For clarification purposes, the following description of the transaction steps in a download transaction use example messages for the processing of the download of an end of period statement (MT940) taking into account German rulebooks. It refers to elements of these example messages, using XPath notation.

The following chapters describe the EBICS messages in the individual phases of a download transaction. The data that is a component of these messages is listed here. Data that is fundamentally optional is marked “(optional)”. Data that may only be missing under certain conditions is instead marked “(conditional)”. Optional XML elements that are missing in the description of an EBICS message relating to a specific transaction phase may not be present in this EBICS message. Optional XML elements that are present in the description of an EBICS message relating to a specific transaction phase MUST always be placed correspondingly in this EBICS message.

EBICS requests for download transactions are (XML) instance documents that conform to ebics_request_H005.xsd and comprise the top-level element `ebics` which is declared in ebics_request_H005.xsd. EBICS responses for download transactions are instance documents that conform to ebics_response_H005.xsd and comprise the top-level element `ebics` which is again declared in ebics_response_H005.xsd.

5.6.1.1.1 EBICS messages in transaction initialisation

- Transmission of the following data in the EBICS request (see example in Diagram 52):

- Host ID of the EBICS bank computer system (`ebicsRequest/header/static/HostID`)
- Transaction phase (`ebicsRequest/header/mutable/TransactionPhase`) with the setting “Initialisation”
- Combination of Nonce and Timestamp to avoid replaying old EBICS messages (`ebicsRequest/header/static/Nonce`, `ebicsRequest/header/static/Timestamp`)
- Subscriber (`ebicsRequest/header/static/PartnerID`, `ebicsRequest/header/static/UserID`) that is submitting an order or that is providing bank-technical ES’s for an existing order.
- (Conditional) technical subscriber (`ebicsRequest/header/static/PartnerID`, `ebicsRequest/header/static/SystemID`)
  `SystemID` must be present if the customer system is a multi-user system. The technical subscriber is responsible for the generation of the EBICS requests (including the identification and authentication signatures) that belong to orders that are submitted or bank-technically signed by the subscriber.
- (Optional) information on the customer product (`ebicsRequest/header/static/Product`)
- Administrative Order type (`ebicsRequest/header/static/OrderDetails/AdminOrderType`)
- Order parameters (`ebicsRequest/header/static/OrderDetails/OrderParams`)
  The characteristics of the order parameters are dependent on the administrative order type. For the download of business transaction formats the order parameters and usage rules are specified in detail in chapter 5.6.1.1.4
- Hash values of the financial institution’s public keys that are available to the subscriber (`ebicsRequest/header/static/BankPubKeyDigests/Authentication`, `ebicsRequest/header/static/BankPubKeyDigests/Encryption`, `ebicsRequest/header/static/BankPubKeyDigests/Signature`).
  Both the utilised hash algorithm and the version of the corresponding identification and authentication, encryption and signature process will be specified for each of these hash values.
  The SHA-256 hash values of the financial institution's certificates X002 and E002 are composed by calculating the SHA-256 hash value of the certificate in DER binary format In Version “H005” of the EBICS protocol the ES of the financial institutions is only planned (see Chapter 3.5.2). The element `BankPubKeyDigests/Signature` is already contained in this description in preparation for future versions of EBICS, but in Version “H005” its maximum frequency (`maxOccurs`) is set to 0.
- Security medium for the subscriber’s bank-technical key (`ebicsRequest/header/static/SecurityMedium`)
- Identification and authentication signature of the technical subscriber, if such is available, otherwise the identification and authentication signature of the subscriber themselves (`ebicsRequest/AuthSignature`)
  The identification and authentication signature includes all XML elements of the EBICS request whose attribute value for `@authenticate` is equal to “true”. The definition of

the XML schema “ebics_request_H005.xsd“ guarantees that the value of the attribute `@authenticate` is equal to “true” for precisely those elements that also need to be signed.

.

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
 xmlns="urn:org:ebics:H005"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_request_H005.xsd"
 Version="H005" Revision="1">
 <header authenticate="true">
   <static>
     <HostID>EBIXHOST</HostID>
     <Nonce>98498A65465C645E645F64565462C645</Nonce>
     <Timestamp>2005-01-30T15:40:45.123Z</Timestamp>
     <PartnerID>CUSTM001</PartnerID>
     <UserID>USR001</UserID>
     <Product Language="en" InstituteID="Institute ID">Product Identifier</Product>
     <OrderDetails>
       <AdminOrderType>BTD</AdminOrderType>
              <BTDOrderParams>
       <Service>
             <ServiceName>EOP</ServiceName>
             <Scope>DE</Scope>
             <MsgName>mt940</MsgName>
       </Service>         <DateRange>
           <Start>2016-09-01</Start>
           <End>2016-09-30</End>
         </DateRange>
       </BTDOrderParams>
     </OrderDetails>
     <BankPubKeyDigests>
       <Authentication Version="X002"
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256">1H/rQr2Axe9hYTV2n/tCp+3UIQQ=</Authentica
tion>
       <Encryption Version="E002"
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256">2joEROI3092OIFP394+WOIer2WI=</Encryption
>
     </BankPubKeyDigests>
     <SecurityMedium>0000</SecurityMedium>
   </static>
   <mutable>
     <TransactionPhase>Initialisation</TransactionPhase>
   </mutable>
 </header>
 <AuthSignature>
   <ds:SignedInfo>
     <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
     <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
     <ds:Reference URI="#xpointer(//*[@authenticate='true'])">
       <ds:Transforms>
         <ds:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
       </ds:Transforms>
       <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
       <ds:DigestValue> …here hash value for authentication..</ds:DigestValue>
     </ds:Reference>
   </ds:SignedInfo>
```

```
    <ds:SignatureValue> …here authentication signature..</ds:SignatureValue>
  </AuthSignature>
  <body/>
</ebicsRequest>
```

Diagram 52: EBICS request for transaction initialisation for download of an end of period statement (MT940)

- Transmission of the following data in the EBICS response (see example in Diagram 53)

- Bank-technical return code (`ebicsResponse/body/ReturnCode`)
- Technical return code (`ebicsResponse/header/mutable/ReturnCode`)
- Technical report text (`ebicsResponse/header/mutable/ReportText`)
- (Conditional) Transaction ID that is unambiguous throughout the bank system (`ebicsResponse/header/static/TransactionID`), if no technical errors have occurred during the transaction initialisation
- Transaction phase (`ebicsResponse/header/mutable/TransactionPhase`) with the setting “Initialisation”
- (Conditional) Total number of order data segments to be transmitted (`ebicsResponse/header/static/NumSegments`), if no technical or bank-technical errors have occurred
- (Conditional) Serial number of the order data segment transmitted in this response (`ebicsResponse/header/mutable/SegmentNumber`), if no technical or bank-technical errors have occurred.
  `SegmentNumber` is always set to 1 in the initialisation phase. The attribute `ebicsResponse/header/mutable/SegmentNumber@lastSegment` specifies whether it is the last data segment
- Identification and authentication signature of the financial institution (`ebicsResponse/AuthSignature`)
  The identification and authentication signature includes all XML elements of the EBICS response whose attribute value for `@authenticate` is equal to “true”. The definition of the XML schema “ebics_response_H005.xsd“ guarantees that the value of the attribute `@authenticate` is equal to “true” for precisely those elements that also need to be signed.
- (Conditional) information for encryption of the order data and possibly the ES of the order data (`ebicsResponse/body/DataTransfer/DataEncryptionInfo`), if no errors of a technical or bank-technical nature have occurred.
  In particular, `DataEncryptionInfo` also contains the asymmetrically-encrypted transaction key (`ebicsResponse/body/DataTransfer/DataEncryptionInfo/TransactionKey`)
- (Conditional) The first order data segment (`ebicsResponse/body/DataTransfer/OrderData`), if no errors of a technical or bank-technical nature have occurred

- (Conditional) The bank-technical ES of the order data from the financial institution (`ebicsResponse/body/DataTransfer/SignatureData`), if no errors of a technical or bank-technical nature have occurred.
  In the EBICS protocol the ES of the financial institutions is only planned (see Chapter 3.5.2).
- (Optional) time stamp for the last updating of the bank parameters (`ebicsResponse/body/TimestampBankParameter`).

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsResponse
xmlns="urn:org:ebics:H005"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:org:ebics:H005 ebics_response_H005.xsd"
Version="H005" Revision="1">
<header authenticate="true">
  <static>
    <TransactionID>FEDCBA4139464436345313243FEDCBA</TransactionID>
    <NumSegments>2</NumSegments>
  </static>
  <mutable>
    <TransactionPhase>Initialisation</TransactionPhase>
    <SegmentNumber lastSegment="false">1</SegmentNumber>
    <ReturnCode>000000</ReturnCode>
    <ReportText>[EBICS_OK] OK</ReportText>
  </mutable>
</header>
<AuthSignature>
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256">
    </ds:SignatureMethod>
    <ds:Reference URI="#xpointer(//*[@authenticate='true'])">
      <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      </ds:Transforms>
      <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
      <ds:DigestValue> …here data digest authentication …</ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
    <ds:SignatureValue> …here authentication signature..</ds:SignatureValue>
</AuthSignature>
<body>
  <DataTransfer>
    <DataEncryptionInfo authenticate="true">
      <EncryptionPubKeyDigest Version="E002"
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256">..here hash value of the public bank key
for encryption..</EncryptionPubKeyDigest>
      <TransactionKey>En6KEB6ArEzw+iq4N1wm6Eptcyx…</TransactionKey>
      <HostID>EBIXHOST</HostID>
    </DataEncryptionInfo>
    <OrderData>…</OrderData>
  </DataTransfer>
  <ReturnCode authenticate="true">000000</ReturnCode>
</body>
</ebicsResponse>
```

Diagram 53: EBICS response for transaction initialisation for the download of an end of period statement (MT940)

5.6.1.1.2 EBICS messages in the data transfer phase

- Transmission of the following data in the EBICS request (see example in Diagram 54):

Host ID of the EBICS bank computer system (`ebicsRequest/header/static/HostID`)

Data for identification of the current transaction step:

- Transaction ID (`ebicsRequest/header/static/TransactionID`)
- Transaction phase (`ebicsRequest/header/mutable/TransactionPhase`) with the setting “Transfer”
- Serial number of the order data segment that is to be downloaded in this transaction step (`ebicsRequest/header/mutable/SegmentNumber`)
  Attribute `ebicsRequest/header/mutable/SegmentNumber@lastSegment` has no meaning for this EBICS request

Identification and authentication signature of the technical subscriber, if such is available, otherwise the identification and authentication signature of the subscriber themselves (`ebicsRequest/AuthSignature`)
The identification and authentication signature includes all XML elements of the EBICS request whose attribute value for `@authenticate` is equal to “true”. The definition of the XML schema “ebics_response_H005.xsd“ guarantees that the value of the attribute `@authenticate` is equal to “true” for precisely those elements that also need to be signed.

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
 xmlns="urn:org:ebics:H005"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_request_H005.xsd"
 Version="H005" Revision="1">
 <header authenticate="true">
   <static>
     <HostID>EBIXHOST</HostID>
     <TransactionID>FEDCBA4139464436344531324 3FEDCBA</TransactionID>
   </static>
   <mutable>
     <TransactionPhase>Transfer</TransactionPhase>
     <SegmentNumber lastSegment="false">2</SegmentNumber>
   </mutable>
 </header>
 <AuthSignature>
   <ds:SignedInfo>
     <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
     <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256">
     </ds:SignatureMethod>
     <ds:Reference URI="#xpointer(//*[@authenticate='true'])">
       <ds:Transforms>
         <ds:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
       </ds:Transforms>
       <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
       <ds:DigestValue>… here hash value for authentication…</ds:DigestValue>
     </ds:Reference>
   </ds:SignedInfo> <ds:SignatureValue> …here authentication signature..</ds:SignatureValue>
```

```
  </AuthSignature>
  </body>
</ebicsRequest>
```

Diagram 54: EBICS request for transmission of the next order data segment for the download of an end of period statement (MT940)

- Transmission of the following data in the EBICS response (see example in Diagram 55)

- Bank-technical return code (`ebicsResponse/body/ReturnCode`)
- Technical return code (`ebicsResponse/header/mutable/ReturnCode`)
- Technical report text (`ebicsResponse/header/mutable/ReportText`)
- Data for identifying a transaction step

  If the technical return code has the value EBICS_TX_RECOVERY_SYNC, this transaction step identifies the last recovery point of the download transaction. However, if no technical or business related errors have not occurred in this example, this transaction step reflects the current transaction step:

  - Transaction ID (`ebicsResponse/header/static/TransactionID`)
  - Transaction phase (`ebicsResponse/header/mutable/TransactionPhase`)
  - Serial number of the order data segment (`ebicsResponse/header/mutable/SegmentNumber`).
    This is the number of the order data segment that has been requested in the EBICS request or, in the event of the error EBICS_TX_RECOVERY_SYNC, the number of the last order data segment that has been successfully transmitted to the customer system by the bank system. In the event of the error EBICS_TX_RECOVERY_SYNC, the value of `SegmentNumber` is always equal to 1 if the value of `TransactionPhase` is "Initialisation".
    The attribute `ebicsResponse/header/mutable/SegmentNumber@lastSegment` specifies whether it is the last order data segment.
- Identification and authentication signature of the financial institution (`ebicsResponse/AuthSignature`)
  The identification and authentication signature includes all XML elements of the EBICS response whose attribute value for `@authenticate` is equal to "true". The definition of the XML schema "ebics_response_H005.xsd" guarantees that the value of the attribute `@authenticate` is equal to "true" for precisely those elements that also need to be signed.
- (Conditional) The requested order data segment (`ebicsResponse/body/DataTransfer/OrderData`), if no errors of a technical or bank-technical nature have occurred.

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsResponse
 xmlns="urn:org:ebics:H005"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_response_H005.xsd"
 Version="H005" Revision="1">
 <header authenticate="true">
```

```
    <static>
      <TransactionID>FEDCBA41394644363445313243FEDCBA</TransactionID>
    </static>
    <mutable>
      <TransactionPhase>Transfer</TransactionPhase>
      <SegmentNumber lastSegment="true">2</SegmentNumber>
      <ReturnCode>000000</ReturnCode>
      <ReportText>[EBICS_OK] OK</ReportText>
    </mutable>
  </header>
  <AuthSignature>
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
      <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256">
      </ds:SignatureMethod>
      <ds:Reference URI="#xpointer(//*[@authenticate='true'])">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
        <ds:DigestValue>… here hash value for authentication …</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>…here authentication signature… </ds:SignatureValue>
  </AuthSignature>
  <body>
    <DataTransfer>
      <OrderData>…</OrderData>
    </DataTransfer>
    <ReturnCode authenticate="true">000000</ReturnCode>
  </body>
</ebicsResponse>
```

Diagram 55: EBICS response for transmission of the last order data segment for the download of an end of period statement (MT940)

5.6.1.1.3 EBICS- messages in the acknowledgement phase

- Transmission of the following data in the EBICS request (see example in Diagram 56)

  - Host ID of the EBICS bank computer system (`ebicsRequest/header/static/HostID`)

  - Data for identification of the current transaction step:

- Transaction ID (`ebicsRequest/header/static/TransactionID`)
- Transaction phase (`ebicsRequest/header/mutable/TransactionPhase`) with the setting “Receipt”

- Identification and authentication signature of the technical subscriber, if such is available, otherwise the identification and authentication signature of the subscriber themselves (`ebicsRequest/AuthSignature`)
  The identification and authentication signature includes all XML elements of the EBICS request whose attribute value for `@authenticate` is equal to “true”. The definition of the XML schema “ebics_request_H005.xsd“ guarantees that the value of the attribute

`@authenticate` is equal to “true” for precisely those elements that also need to be signed

- Acknowledgement (`ebicsRequest/body/TransferReceipt/ReceiptCode`):
  The value of the acknowledgement is 0 (“positive acknowledgement”) if download and processing of the order data was successful. Otherwise the value of the acknowledgement is 1 (“negative acknowledgement”).

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
 xmlns="urn:org:ebics:H005"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_request_H005.xsd"
 Version="H005" Revision="1">
 <header authenticate="true">
   <static>
     <HostID>EBIXHOST</HostID>
     <TransactionID>FEDCBA41394644363445313243FEDCBA</TransactionID>
   </static>
   <mutable>
     <TransactionPhase>Receipt</TransactionPhase>
   </mutable>
 </header>
 <AuthSignature>
   <ds:SignedInfo>
     <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
     <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256">
     </ds:SignatureMethod>
     <ds:Reference URI="#xpointer(//*[@authenticate='true'])">
       <ds:Transforms>
         <ds:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
       </ds:Transforms>
       <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
       <ds:DigestValue>…here hash value for authentication …</ds:DigestValue>
     </ds:Reference>
   </ds:SignedInfo>
   <ds:SignatureValue>… here authentication signature…</ds:SignatureValue>
 </AuthSignature>
 <body>
   <TransferReceipt authenticate="true">
     <ReceiptCode>0</ReceiptCode>
   </TransferReceipt>
 </body>
</ebicsRequest>
```

Diagram 56: EBICS request for the acknowledgement of download data

- Transmission of the following data in the EBICS response (see example in Diagram 57)

- Bank-technical return code (`ebicsResponse/body/ReturnCode`)
- Technical return code (`ebicsResponse/header/mutable/ReturnCode`)
- Technical report text (`ebicsResponse/header/mutable/ReportText`)
- Data for identification of a transaction step:
  If the technical return code has the value EBICS_TX_RECOVERY_SYNC, this transaction step identifies the last recovery point of the download transaction. However,

if no technical or business related errors have occurred, this transaction step reflects the current transaction step, i.e. acknowledgement of the download data:

- Transaction ID (`ebicsResponse/header/static/TransactionID`)
- Transaction phase (`ebicsResponse/header/mutable/TransactionPhase`)
- (Conditional) Serial number of the order data segment (`ebicsResponse/header/mutable/SegmentNumber`) if the error EBICS_TX_RECOVERY_SYNC has occurred and consequently the value of `TransactionPhase` is “Initialisation” or “Transfer”.
  This is the number of the order data segment that, from the bank system’s perspective, was the last one to have been successfully transmitted to the customer system. The value of `SegmentNumber` is always equal to 1 if the value of `TransactionPhase` is “Initialisation”.
  The attribute `ebicsResponse/header/mutable/SegmentNumber@lastSegment` specifies whether it is the last order data segment.
- Identification and authentication signature of the financial institution (`ebicsResponse/AuthSignature`)
  The identification and authentication signature includes all XML elements of the EBICS response whose attribute value for `@authenticate` is equal to “true”. The definition of the XML schema “ebics_response_H005.xsd“ guarantees that the value of the attribute `@authenticate` is equal to “true” for precisely those elements that also need to be signed.

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsResponse
 xmlns="urn:org:ebics:H005"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_response_H005.xsd"
 Version="H005" Revision="1">
 <header authenticate="true">
   <static>
     <TransactionID>FEDCBA4139464436344531324 3FEDCBA</TransactionID>
   </static>
   <mutable>
     <TransactionPhase>Receipt</TransactionPhase>
     <ReturnCode>011000</ReturnCode>
     <ReportText>[EBICS_POSTPROCESS_DONE] positive receipt received </ReportText>
   </mutable>
 </header>
 <AuthSignature>
   <ds:SignedInfo>
     <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
     <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256">
     </ds:SignatureMethod>
     <ds:Reference URI="#xpointer(//*[@authenticate='true'])">
       <ds:Transforms>
         <ds:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
       </ds:Transforms>
       <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
       <ds:DigestValue>… here hash value for authentication</ds:DigestValue>
     </ds:Reference>
   </ds:SignedInfo>
   <ds:SignatureValue>… here authentication signature…</ds:SignatureValue>
```

```
  </AuthSignature>
  <body>
    <ReturnCode authenticate="true">000000</ReturnCode>
  </body>
</ebicsResponse>
```

Diagram 57: EBICS response for the acknowledgement of download data

5.6.1.1.4 Download Request Structure for Business Transaction Formats (BTF)

The standard process is described in chapter 5.6.1. The values of the order parameters are positioned in `ebicsRequest/header/static/OrderDetails` of the type `BTDOrderParams` which is usable in the upload direction and with order type "BTD":

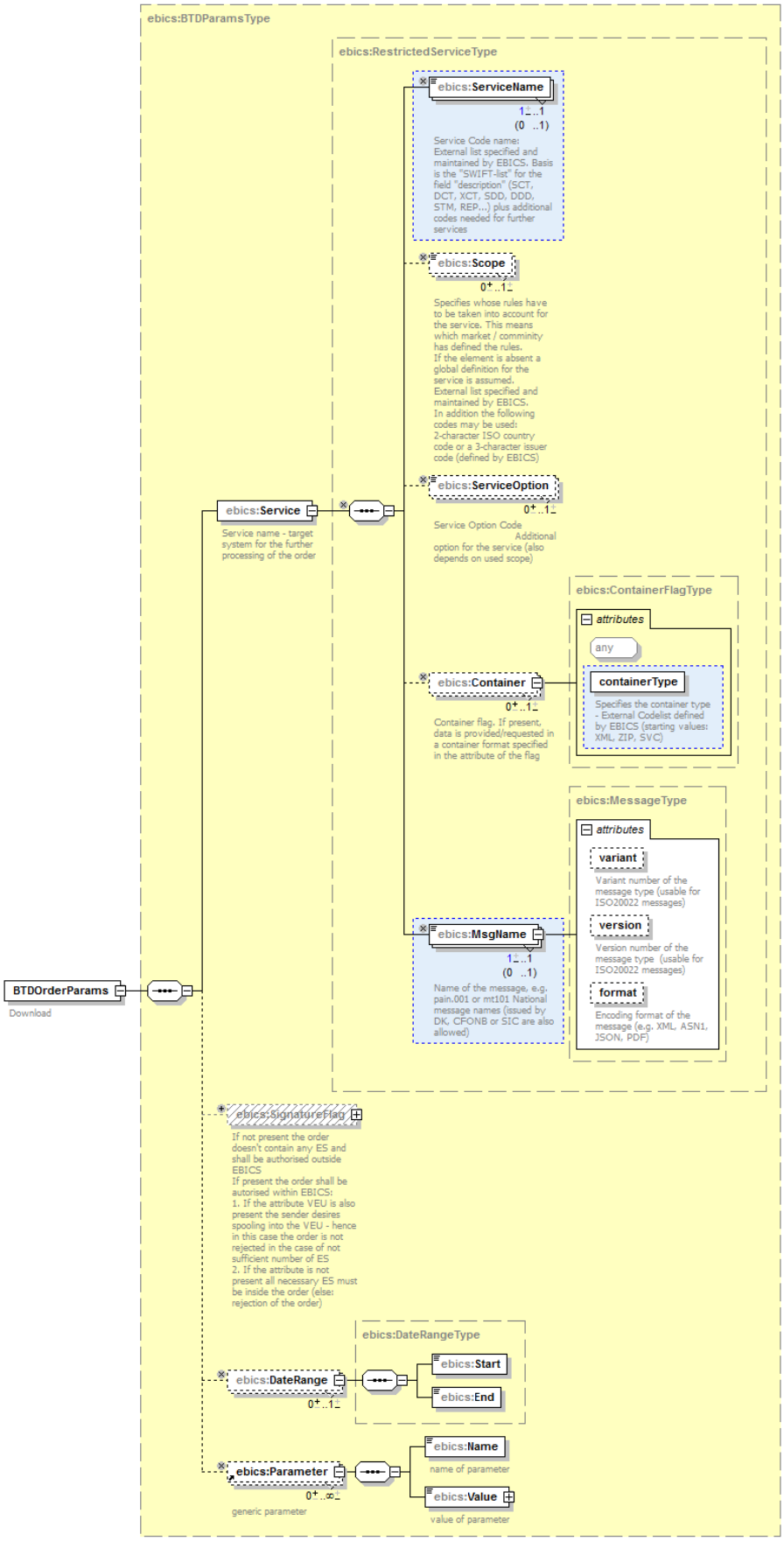


Diagram 58: BTF structure for download (using restricted service type)

XML element/ attribute	Data type	#	Meaning	Example
BTDOrderParams	ebics:BTDParamsType (complex)	1	Required to qualify an download business transaction and the used/corresponding format	- (complex)
Service	RestrictedServiceType (complex)	1	Indicates the target system (nature)/process to handle the Transaction/File	- (complex)
ServiceName	ebics: ServiceNameStringType (simple) - restriction base: minLength value="3" maxLength value="3" pattern = [A-Z0-9]	1	Name of the service	ServiceName is subject to an external code list (maintained by EBICS) - Example: “REP” = Report
Scope	ebics: ScopeStringType (simple) restriction base: minLength value="2" maxLength value="3"	0..1	Specifies whose rules have to be taken into account for the service. This means which market / community defined the rules. External scope name list specified and maintained by EBICS. A missing scope element means globally accepted rules.	Scope is subject to an external code list (maintained by EBICS). 2-char country codes 3-char codes for other scopes “BIL” means bilaterally agreed The meaning of a missing Scope element is global. Instead of a missing scope element it can also be provided as code “GLB”.

ServiceOption	ebics:CodeStringType (simple) restriction base: minLength value="3" maxLength value="6" pattern = [A-Z0-9]	0..1	Optional characteristic(s) of a service	ServiceOption is subject to external code lists (global, market, bilateral) Example: “B2B” means in the context of ServiceName = “REP”: The report only contains information about B2B debits
Container	ebics:ContainerFlagType	0..1	Flag to indicate the use of a container	Only value true allowed. No presence means false
Container@containerType	ebics:ContainerstringType (simple) restriction base: minLength value="3" maxLength value="3" pattern = [A-Z0-9]	1..1	Indicates what type of container is used.	If the container Flag is present, one of the internal values has to be used (internal code list) “XML” “ZIP” “SVC”
MsgName	ebics:MessageNameStringType (simple) restriction base: minLength value="1" maxLength value="10" pattern = [a-z\.0-9]	1	message names starting with a BA code (ISO) or MT (FIN) or string to be evaluated	“pain.002”, “mt940” Message names (issued by markets, specified in “scope”) are also allowed
MsgName@version	ebics:NumString (simple) restriction base: minLength value="2" maxLength value="2" pattern = [0-9]	0..1	Used ISO version of message, ignored if no ISO message name	“03”
MsgName@variant	ebics:NumString (simple) restriction base: minLength value="3"	0..1	Evaluated together with <MsgName>, ignored if no ISO message name	“001”

	maxLength value="3" pattern = [0-9]			
MsgName@format	ebics:CodeString (simple) restriction base: minLength value="1" maxLength value="4" pattern = [A-Z0-9]	0..1	Evaluated together with <MsgName>, admissible for each kind of message name, but only to be used if it is not the standard format for the used message standard (especially non-XML for ISO 20022).	„XML“, „ASN1“, „JSON“, „PDF“
dateRange	DateRangeType (complex)	0..1	Specifies a date range for data in the requested message	
start	DateType	1	Start date (incl.)	2016-10-11
end	DateType	1	End date (incl.)	2016-10-11
Parameter		0..1	Generic oder params: Any number of Name-Value pairs can be specified	
Name	boolean	1..1	Name of parameter	
Value	anySimpleType	1..1	Value of parameter	
Value@type	NCName	1..1	Type of value	Recommen-dation for a default is string

5.6.1.2 Processing the EBICS messages

Chapter 5.6.1.1 describes the contents of the EBICS messages that are exchanged within the framework of a download transaction. The subject of this chapter is the processing of these EBICS messages. Action sequences are pointed out in the flow diagram in Diagram 51, the course of which is described here in greater detail.

In order to simplify the description of the processes, it is assumed that every processing step produces a return code (RC) whose value is equal to EBICS_OK (000000) if it has been possible to successfully complete this step. The technical return code (RCT) and the bank-technical return code (RCF) are set depending on the RC, and their values then flow into EBICS messages.

The validity of the EBICS requests is verified on the basis of the XML schema definition file “ebics_request_H005.xsd”, and with due regard to the restrictions that have been specified for the individual requests in Chapter 5.6.1.1. The validity verification usually takes place in parallel and/or interlocked with the other steps in processing the EBICS request. The following processes dispense with representation of a process step of type “EBICS request validity verification” in favour of the simplest possible representation. In consequence, these processes can be terminated by the following additional technical errors:

EBICS_INVALID_XML, EBICS_INVALID_REQUEST, or EBICS_INVALID_REQUEST_CONTENT. For Return codes relating to CA-issued certificates, refer to Annex 1
.

5.6.1.2.1 Processing in the initialisation phase

Diagram 59 shows processing at the bank's end of the EBICS request which is sent from the customer system to the bank system in the initialisation stage of a download transaction. The individual processing steps are explained in greater detail in the following text:

I. Generation of an EBICS transaction (see 5.5.1.2.1 Point 1 and Diagram 43)

II. Termination of the EBICS transaction

If the requested download data is not available, the EBICS transaction is terminated with the business related return code EBICS_NO_DOWNLOAD_DATA_AVAILABLE.

III. Provision of data

In this processing step the first order data segment is provided for the purpose of being embedded in the EBICS response. If the financial institution uses the bank-technical ES's for the current order type and the current subscriber (submitter), the financial institution's bank-technical ES's are also provided via the order data.

In Version "H005" of the EBICS protocol the ES of the financial institutions is only planned (see Chapter 3.5.2). They are only taken into consideration here in preparation for future EBICS versions.

The provision of download data is not a part of EBICS, it is dependent on the implementation of the bank system.

IV. Generation of the EBICS response

This processing step generates the EBICS response that is afterwards sent to the customer system. If all previous processing steps have been successful, this EBICS message contains the first order data segment and possibly also the bank-technical signature for the (entire) order data. In the event of an error, this EBICS message contains the corresponding technical or business related error code. The contents of this EBICS message are described in greater detail in Chapter 5.6.1.1.1.

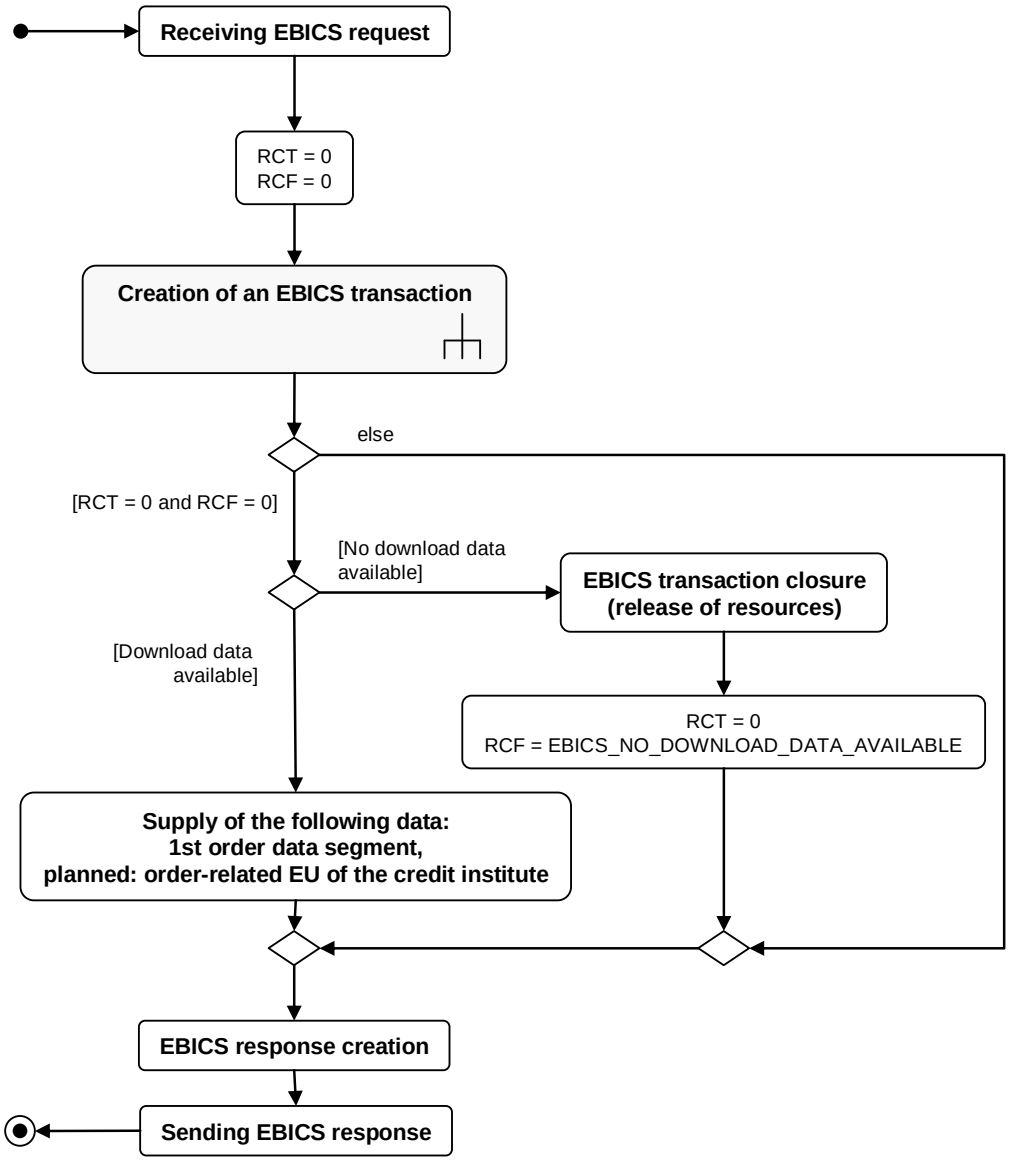


Diagram 59: Processing the EBICS request of the initialisation phase of a download transaction

5.6.1.2.2 Processing in the data transfer phase

Diagram 61 shows processing at the bank's end of the EBICS request which is transferred from the customer system to the bank system in the data transfer stage of an EBICS transaction. The individual processing steps are explained in greater detail in the following text:

I. Verifying the download transaction (see Diagram 60)

I.a. Verifying the EBICS transaction (see 5.5.1.2.2 Point 1 and Diagram 45)

I.b. Evaluation of the EBICS transaction verification results

If the transaction step verification is unsuccessful, then:

- A verification is carried out as to whether the download transaction can be recovered, if the bank system supports the recovery of transactions. This verification is carried out in accordance with the description in Chapter 5.6.2. If the verification is successful, the technical return code EBICS_TX_RECOVERY_SYNC is returned, otherwise the transaction is terminated with the technical return code EBICS_TX_ABORT.

- The download transaction is terminated with the business related error code EBICS_RECOVERY_NOT_SUPPORTED if the bank system does not support transaction recovery. If MAX is set to 0 in the flow diagram, the case is also considered where recovery is not supported.

II. Provision of data

In this processing step the requested order data segment is provided for the purpose of being embedded in the EBICS response. The provision of download data is not a part of EBICS, it is dependent on the implementation of the bank system.

III. Generation of the EBICS response

This processing step generates the EBICS response that is afterwards sent to the customer system. If all previous processing steps have been successful, this EBICS message contains the order data segment that was requested in the corresponding EBICS request. In the event of an error, this EBICS message contains the corresponding technical business related error code. The contents of this EBICS message are described in greater detail in Chapter 5.6.1.1.2.

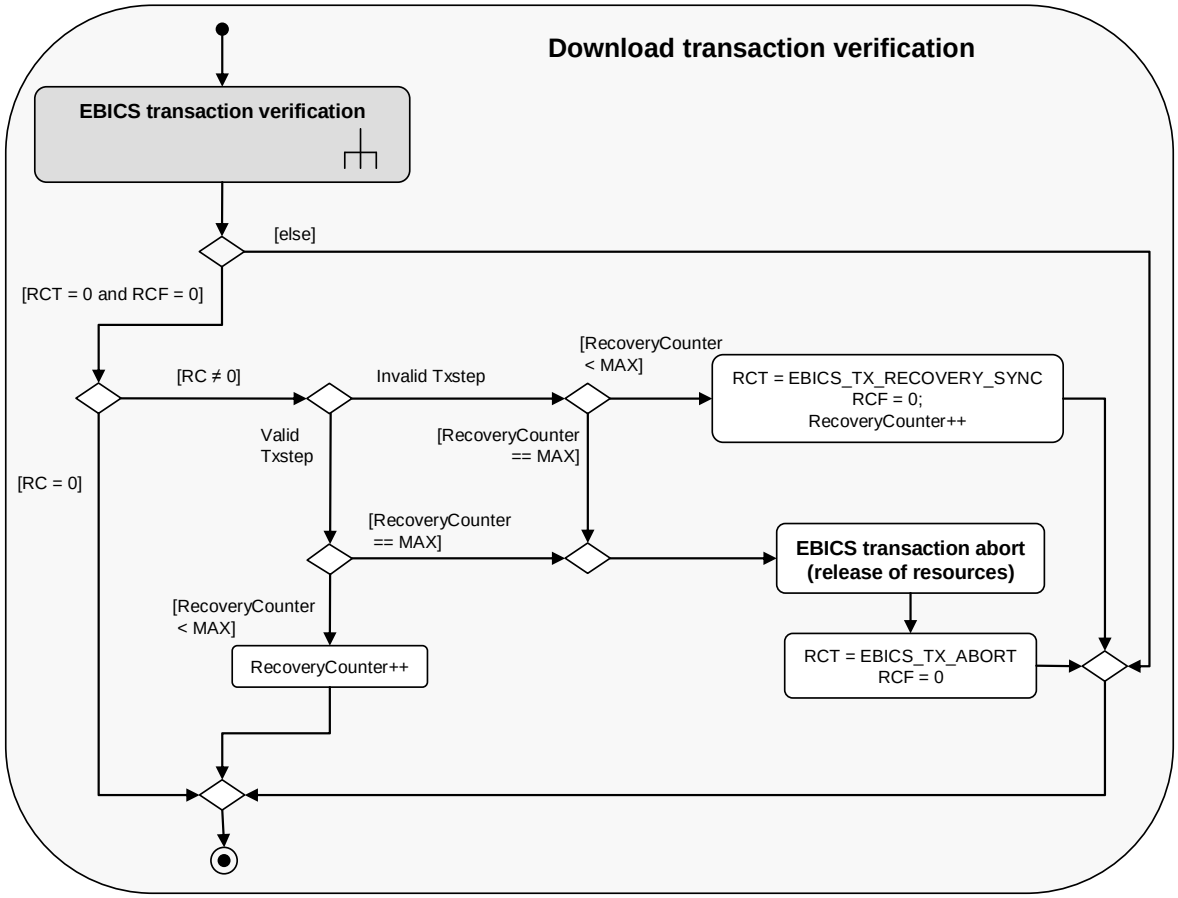


Diagram 60: Detailed description of the process step "Download transaction verification"

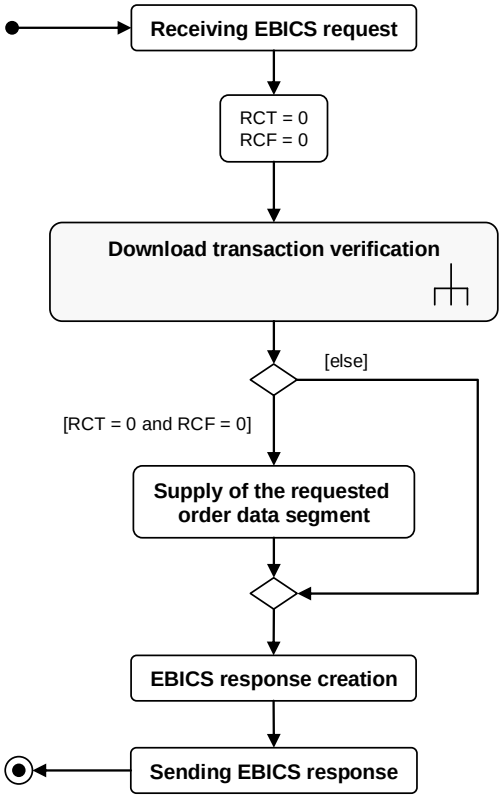


Diagram 61: Processing an EBICS request for requesting a order data segment

5.6.1.2.3 Processing in the acknowledgement phase

Diagram 62 shows processing at the bank's end of the EBICS request which is transferred from the customer system to the bank system in the acknowledgement stage of an EBICS transaction.

The individual processing steps are explained in greater detail in the following text:

I. Verifying the download transaction (see description in Chapter 5.6.1.2.2, Point 1)

II. Download post-processing

Positive acknowledgement means that it was possible to successfully download and process the order data from the customer system. In contrast to negative acknowledgement, the consequence of this is that finishing-off activities can now be carried out on the bank system such as e.g. marking the order data as "downloaded". The EBICS transaction is terminated by the bank system, independent of the type of acknowledgement.

III. Termination of the EBICS transaction

IV. Generation of the EBICS response

This processing step generates the EBICS response that is afterwards sent to the customer system. In the event of positive acknowledgement, the technical return code EBICS_DOWNLOAD_POSTPROCESS_DONE is returned, in the event of negative acknowledgement the technical return code

EBICS_DOWNLOAD_POSTPROCESS_SKIPPED is returned. In the event of an error, this EBICS message contains the corresponding technical or business related error code. The contents of this EBICS message are described in greater detail in Chapter 5.6.1.1.3.

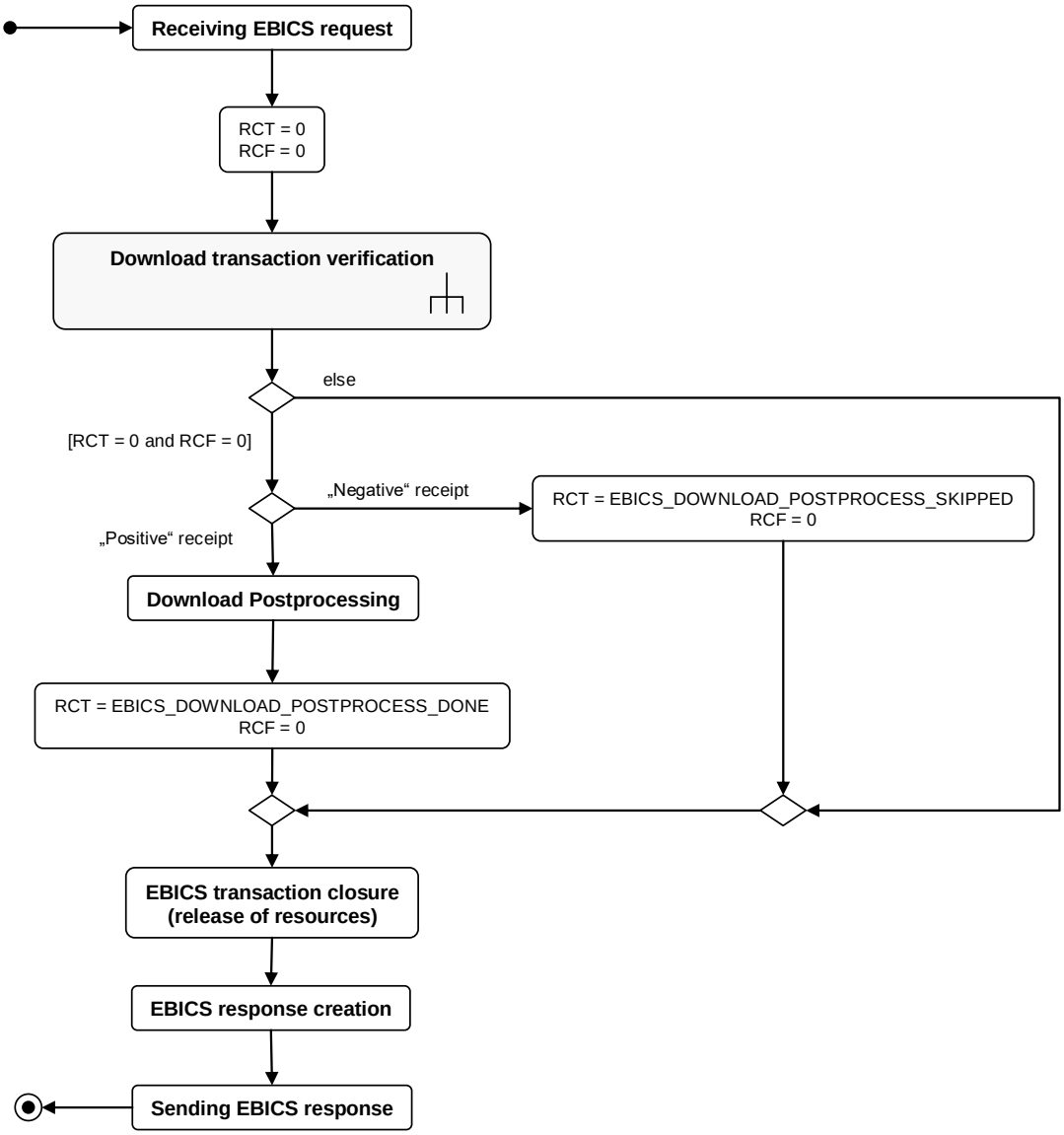


Diagram 62: Processing of an EBICS request for acknowledgement within the framework of a download transaction

5.6.2 Recovery of download transactions

Recovery of download transactions is always initiated by the customer system. The reasons for a recovery are analogous to those of upload transactions:

- Transport error during transmission of an EBICS request in the data transfer or acknowledgement phase of the transaction
- Timeout or transport error when receiving an EBICS transaction in the data transfer or acknowledgement phase of the transaction

- Loss at the subscriber's end of order data segments that have already been received
- Temporary error in the processing of a received EBICS response that necessitates renewed transmission.

If one of the above error situations occurs, the customer system selects a suitable recovery point depending on the number of available order data segments at the subscriber's end. If the selected recovery point is the request for the n^{th} order data segment, then the next transaction step initiated by the subscriber is the request for the $(n+1)^{th}$ order data segment or acknowledgement of the download of all order data segments if n is the last order data segment. EBICS requests within the framework of the recovery of download transactions do not differ from the EBICS request of a normal, error-free flow of a download transaction.

By way of example, the flow of a transaction that repeatedly necessitates recovery is shown in Diagram 63. In each case, the recovery takes place without explicit synchronisation between the customer system and the bank system. The 3^{rd} order data segment is requested three times since the customer system could not receive the corresponding EBICS response due to a timeout or a transport error. On the second and third request of the 3^{rd} order data segment, the customer system assumes that the recovery point is the request for the 2^{nd} order data segment. The value of the recovery counter is equal to 2 after the third (and successful) request of the 3^{rd} order data segment, since the last two requests of the 3^{rd} order data segment were evaluated as recovery attempts by the bank system. The transaction finally fails due to the number of recovery attempts being too high.

If the selected recovery point is not valid from the viewpoint of the bank system, the EBICS response contains the last possible recovery point of the download transaction in addition to the technical return code EBICS_TX_RECOVERY_SYNC. The valid recovery points of a download transaction are defined in Chapter 5.4. If, for example, the selected recovery point is the request for the order data segment with serial number k, the transaction can be continued with the request for the order data segments with serial numbers I+1, I+2, wherein i <= k must hold. If i < k, the i^{th} order data segment is requested again, then the counter for the number of implemented recovery attempts is incremented by one.

Diagram 64 shows the successful flow of a transaction that contains a recovery of the transaction after an explicit synchronisation between the customer system and the bank system. Here, the customer system requests the 5^{th} order data segment in one state without having previously requested the 4^{th} order data segment. The financial institution's EBICS response (see Diagram 65) thus contains the recovery point of the transaction, which in this case is the request of the 3^{rd} order data segment. Following this, the customer system continues with the request of the 4^{th} order data segment and ends the transaction after receipt of the last segment 5.

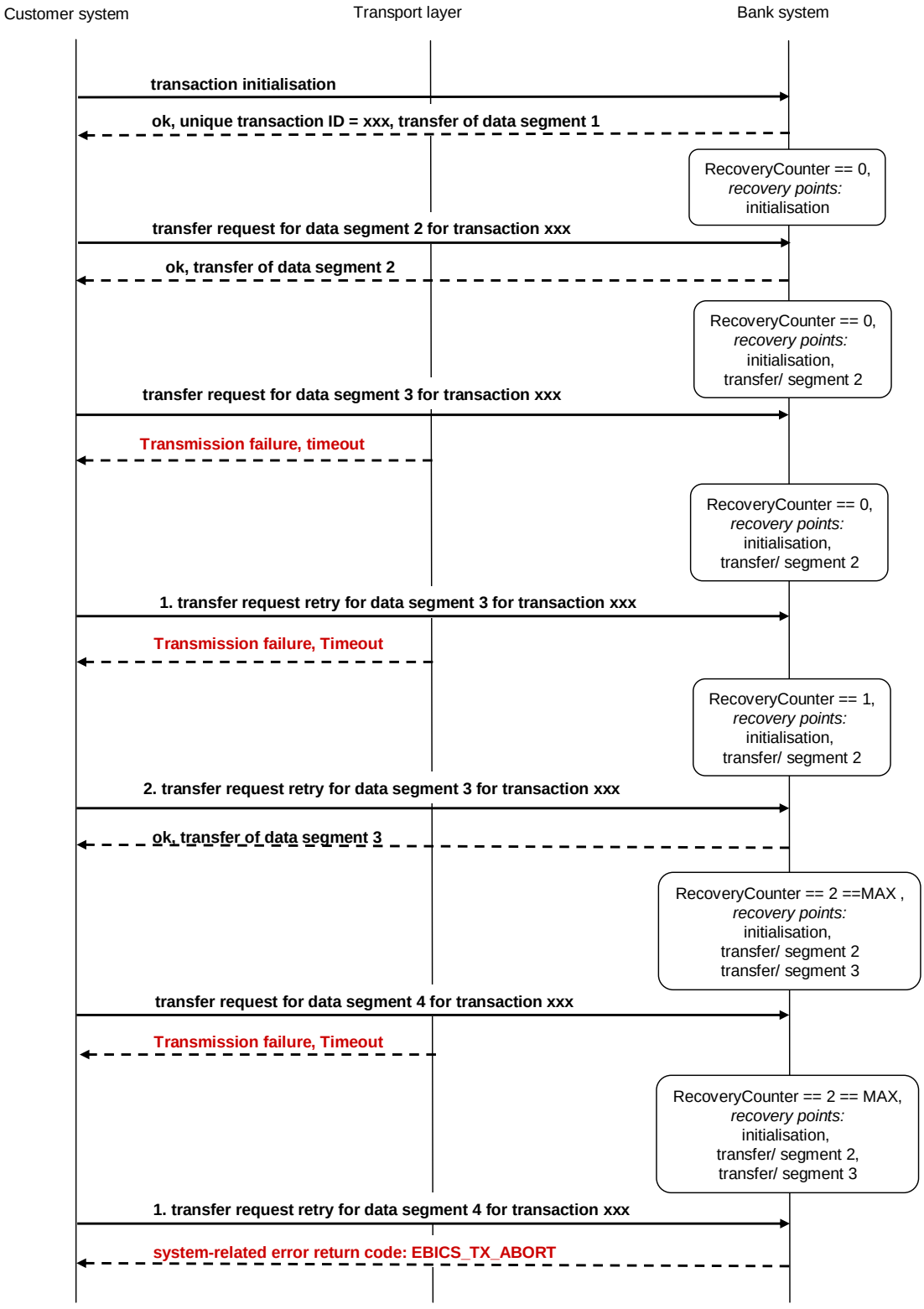


Diagram 63: Termination of the recovery of a download transaction due to the maximum number of recovery attempts being exceeded

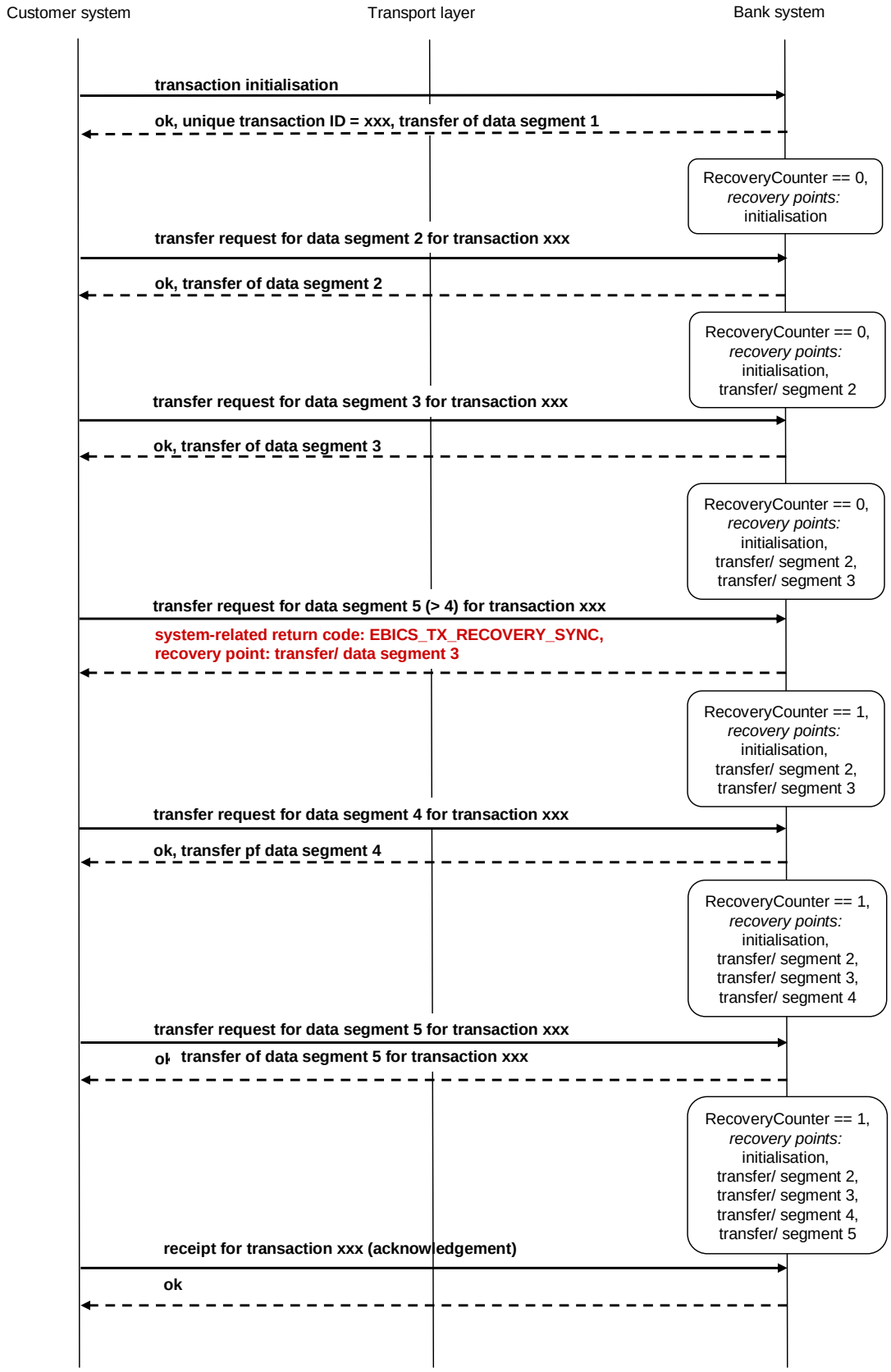


Diagram 64: Recovery of a download transaction with explicit synchronisation between customer system and bank system

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsResponse
 xmlns="urn:org:ebics:H005"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_response_H005.xsd"
 Version="H005" Revision="1">
 <header authenticate="true">
   <static>     <TransactionID>FEDCBA4139464436344531324 3FEDCBA</TransactionID>
   </static>
   <mutable>
     <TransactionPhase>Transfer</TransactionPhase>
     <SegmentNumber lastSegment="false">3</SegmentNumber>
     <ReturnCode>061101</ReturnCode>
     <ReportText>[EBICS_TX_RECOVERY_SYNC] Synchronisation necessary</ReportText>
   </mutable>
 </header>
 <AuthSignature>
   <ds:SignedInfo>
     <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
     <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256">
     </ds:SignatureMethod>
     <ds:Reference URI="#xpointer(//*[@authenticate='true'])">
       <ds:Transforms>
         <ds:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
       </ds:Transforms>
       <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
       <ds:DigestValue>… here hashvalue for authentication …</ds:DigestValue>
     </ds:Reference>
   </ds:SignedInfo>
   <ds:SignatureValue>… here authentication signature …</ds:SignatureValue>
 </AuthSignature>
 <body>
   <ReturnCode authenticate="true">000000</ReturnCode>
 </body>
</ebicsResponse>
```

Diagram 65: EBICS response with technical error EBICS_TX_RECOVERY_SYNC

6 Encryption

EBICS provides encryption on two different protocol levels: On the level of the EBICS XML-based application protocol and on a level between the application and transport level, namely the TLS level.

6.1 Encryption at TLS level

The use of TLS on the external transmission paths between the customer system and the bank system ensures maintenance of the confidentiality and integrity of the EBICS messages on these paths. The cryptographic processes that are used to establish a TLS session between the customer system and the bank system are described in the Appendix (Chapter 11.3.1).

6.2 Encryption at application level

The order data of bank-technical orders are fundamentally deemed to be sensitive and are therefore embedded into EBICS messages in encrypted form. This facilitates maintenance of their confidentiality on the internal paths of the customer system and the bank system on which communication is not necessarily based on TLS.

The order data of system-related key management orders is encrypted as soon as the recipient's (sufficiently verified) encryption key is available to the sender of the order data. The order data of INI or HIA orders is thus embedded into the EBICS message in an unencrypted form, but the order data of HPB, PUB, HCS, or HCA orders is encrypted.

The order data of system-related Distributed Electronic Signature orders is also embedded in the EBICS message in encrypted form.

Analogous to the order data of bank-technical orders, the electronic signatures of an order, i.e. the transport signature or the bank-technical ES's are always encrypted.

Apart from the order data and the ES's, no further data is encrypted at the application level.

Order data that is to be encrypted and ES's of an order are initially compressed via ZIP, then encrypted and finally base64-coded and embedded in the EBICS message. Here, compression and subsequent encryption of the order data takes place before it is segmented. The implemented encryption process is a hybrid process: The data is symmetrically encrypted, the utilised symmetrical key is passed to the recipient of the data in asymmetrically-encrypted form. Details on the encryption process are given in the Appendix (Chapter 11.3.2).

In the event of an upload transaction, a random symmetrical key is generated in the customer system that is used exclusively within the framework of this transaction both for encryption of the ES's and for encryption of the order data. This key is encrypted

asymmetrically with the financial institution's public encryption key and is transmitted by the customer system to the bank system during the initialisation phase of the transaction.

Analogously, in the case of a download transaction a random symmetrical key is generated in the bank system that is used for encryption of the order data that is to be downloaded and for encryption of the bank-technical signature that has been provided by the financial institution. This key is asymmetrically encrypted and is transmitted by the bank system to the customer system during the initialisation phase of the transaction. The asymmetrical encryption takes place with the technical subscriber's public encryption key if the transaction's EBICS messages are sent by a technical subscriber. Otherwise the asymmetrical encryption takes place with the public encryption key of the non-technical subscriber, i.e. the submitter of the order.

From EBICS 2.4 on, the customer system has to use the E002-hash value of the public bank key in a request. This hash value is generated by the customer system according to the E002 process by means of SHA-256.
The transaction is cancelled and the return code EBICS_INVALID_REQUEST_CONTENT is returned if E001 is still used in a request.

7 Segmentation of the order data

7.1 Process description

In Version H005 of the EBICS standard, order data that requires more than 1 MB of storage space in compressed, encrypted and base64-coded form MUST be segmented before transmission, irrespective of the transfer direction (upload/download).

The following procedure is to be followed with segmentation:

1. The order data is ZIP compressed
2. The compressed order data is encrypted in accordance with Chapter 6.2
3. The compressed, encrypted order data is base64-coded.
   In doing this, only the 65 printable characters of the base64 alphabet from RFC 2045 are permitted in EBICS in the resulting coded data block. In particular, so-called “white-space characters” such as spaces, tabs, carriage returns and line feeds (“CR/LF”) are not permitted
4. The result is to be verified with regard to the data volume:

4i. If the resulting data volume is below the threshold of 1 MB = 1,048,576 bytes, the order data can be sent complete as a data segment within one transmission step

4ii. If the resulting data volume exceeds 1,048,576 bytes the data is to be separated sequentially and in a base64-conformant manner into segments that each have a maximum of 1,048,576 bytes.

Step 4i ensures that even order data that does not exceed the permitted maximum segment size of 1 MB when in compressed, encrypted and coded form is handled uniformly within the framework of segmentation.

The recipient executes the algorithmic computations in reverse order to recovere the original order data:

1. The data segment that has just been received is appended (concatenated) to the already-received data segments
2. The complete data block is base64-decoded
3. The results of the base64-decoding are decrypted in accordance with Chapter 6.2
4. The results of the decryption are ZIP expanded to reveal the original order data.

7.2 Implementation in the EBICS messages

The sender of the order data numbers the data segments that are generated in accordance with Chapter 7.1 sequentially in ascending order, beginning with 1.

The server terminates the connection with the technical error code EBICS_TX_SEGMENT_NUMBER_EXCEEDED if the client in an **upload transaction** has specified the total number of segments that are to be transmitted, as stated in the initialisation phase, too low in the field `ebics/header/static/NumSegments`, i.e. if the following applies to the current transaction step:

- `ebicsRequest/header/mutable/SegmentNumber = ebicsRequest/header/static/NumSegments` (from the initialisation phase) and `ebicsRequest/header/mutable/SegmentNumber@lastSegment≠"true"`, or
- `ebicsRequest/header/mutable/SegmentNumber > ebicsRequest/header/static/NumSegments` (from the initialisation phase).

The server terminates the transaction in a regular manner with the technical return code of severity level 'info' EBICS_TX_SEGMENT_NUMBER_UNDERRUN if the client in an **upload transaction** has specified the total number of segments that are to be transmitted, as stated in the initialisation phase, too high in the field `ebicsRequest/header/static/NumSegments`, i.e. if the following applies to the current transaction step:

- `ebicsRequest/header/mutable/SegmentNumber < ebicsRequest/header/static/NumSegments` (from the initialisation phase) and `ebicsRequest/header/mutable/SegmentNumber@lastSegment="true"`.

The server terminates the transaction with the technical error code EBICS_SEGMENT_SIZE_EXCEEDED if the client in an **upload transaction** has exceeded the permitted segment size of 1 MB in the current transaction step.

In the case of **download transactions**, it is the responsibility of the customer system to respond to irregularities regarding the number or size of segments:

- If the actual number of transmitted segments up until attribute setting `ebicsRequest/header/mutable/SegmentNumber@lastSegment="true"` is lower than the specification in the initialisation stage on the part of the server, the client SHOULD nevertheless duly continue the current transaction with the acknowledgement phase.
- If the server exceeds the total number of segments postulated in the initialisation phase, the client CAN nevertheless continue the transaction by requesting further segments. Alternatively, or in the event of a disproportionately-large deviation between the actual segment number and the specified number, the client CAN interrupt the transaction by sending no further requests.
- If the server exceeds the permitted segment size of 1 MB, the client SHOULD terminate the transaction.

8 Electronic Distributed Signature (EDS)

Support by the bank for the Distributed Electronic Signature is compulsory for EBICS-conformant implementation, i.e. a bank server MUST support the administrative order types for the EDS. Information as to whether the financial institution supports the optional EDS order type “HVT” (Retrieve EDS transaction details) is contained in the bank parameters (see Chapter 9.2.2, Parameter “DistribSigTransactionDetails”).

8.1 Process description

The Distributed Electronic Signature (EDS) allows orders to be authorised by multiple subscribers, even from different customers, independently of location and time. Here, an order remains stored in the EDS processing system until via the EDS orders either the necessary number of signatures with suitable authorisation have been received, a time limit set by the bank’s computer system has been exceeded or the order is cancelled. Hence the EDS process is not just an alternative to customer-internal subsequent submission of ES’s relating to an existing order, it also offers a distributed ES among a number of customers with comprehensive possibilities for information on the EDS state and the order.

Authorised signatories of a customer can use signature processes deviating from each other which may support different hash processes resulting in different hash values. In the case of the EDS process, the hash value of the order data is provided when the administrative order types HVD and HVZ are executed. This hash value is derived from the signature version which the subscriber executing HVZ and HVD uses. The hash value is provided with the signature version used as an attribute.

A complete EDS order process generally proceeds as follows:

1. The order party initiates the order (e.g. a SEPA credit transfer) by transmitting the order data in an EBICS transaction with a signature flag (this means that he wants to sign within EBICS) and the attribute RequestEDS (this means that it is possible to add possibly missing ESs via EDS). . For the signature, the order party can either immediately bank-technically sign the order (signature class A, B or E) or can initially carry out the transmission by means of a transport signature (signature class T).
2. The bank system analyses the business transaction identifiers (BTF identifiers) and signatures that have already been submitted, including their class. If further signatures are necessary for processing of the order, it is stored intermediately for the EDS process together with its hash value. The bank system extracts the hash value of the order data from the ES using the signatory’s public signature key.
3. If another subscriber wants to use the EDS process for this order, they have possibly already received the data necessary for authorisation – hash value of the order, BTF identifiers and order number – outside of EBICS (via a third

communication path). In this event, the process continues from Point 4. On the other hand, if they still need the order data they can proceed as follows:

3i. Firstly, they inquire via the administrative order type HVU or HVZ to find out which orders they are authorised to sign within the framework of EDS. The response contains, among other things, information on the business transaction (BTF identifiers), order number, the number of signatures required and already provided (including a note as to whether their own signature is still required or has already been provided), on the original order party and on the size of the uncompressed order data. The HVZ response contains additional information, especially the hash value of the order data. If HVZ is applied, step 3ii may be skipped.

3ii. Next they ascertain via HVD the state of one of these orders, e.g. the SEPA credit transfer order placed within the framework of the EDS. In addition to the hash value of the order data that the bank system has extracted from the order signatory's ES and an accompanying note, they receive a list of the previous signatories together with their authorisation class.

3iii. The subscriber can download additional order details via HVT: Depending on the request parameters, they receive either information on the individual order transactions (account data, amount information, processing date, utilisation data and other descriptions) or the complete order data.

4. The subscriber now has all information needed to sign or cancel the original order:

4i. If they want to add a signature to the original order, they will use HVE. For this, they sign the hash value of the order data received via HVD or worked out themselves from the complete order data via HVT. The HVE control data contains the order parameters for the original order (e.g. the SEPA credit transfer order).

4ii. If they want to cancel the original order they would use HVS. As with HVE, authorisation is confirmed by the bank-technical signature via the hash value of the order data, but in the case of HVS the signature applies as confirmation of the cancellation, not confirmation of the order itself. As with HVE, the HVS control data contains the order parameters of the original order (that is to be cancelled).

Diagram 66 documents the processes when using EDS. The diagram shows the logical concatenation of the EDS order types wherein pure communications connections (e.g. data transmission from bank system to customer system on retrieval of EDS details via HVD), occurrences of errors and the acquisition of information via alternative communication channels (e.g. order hash value by email from the submitter instead of via HVD) are not shown for reasons of clarity.

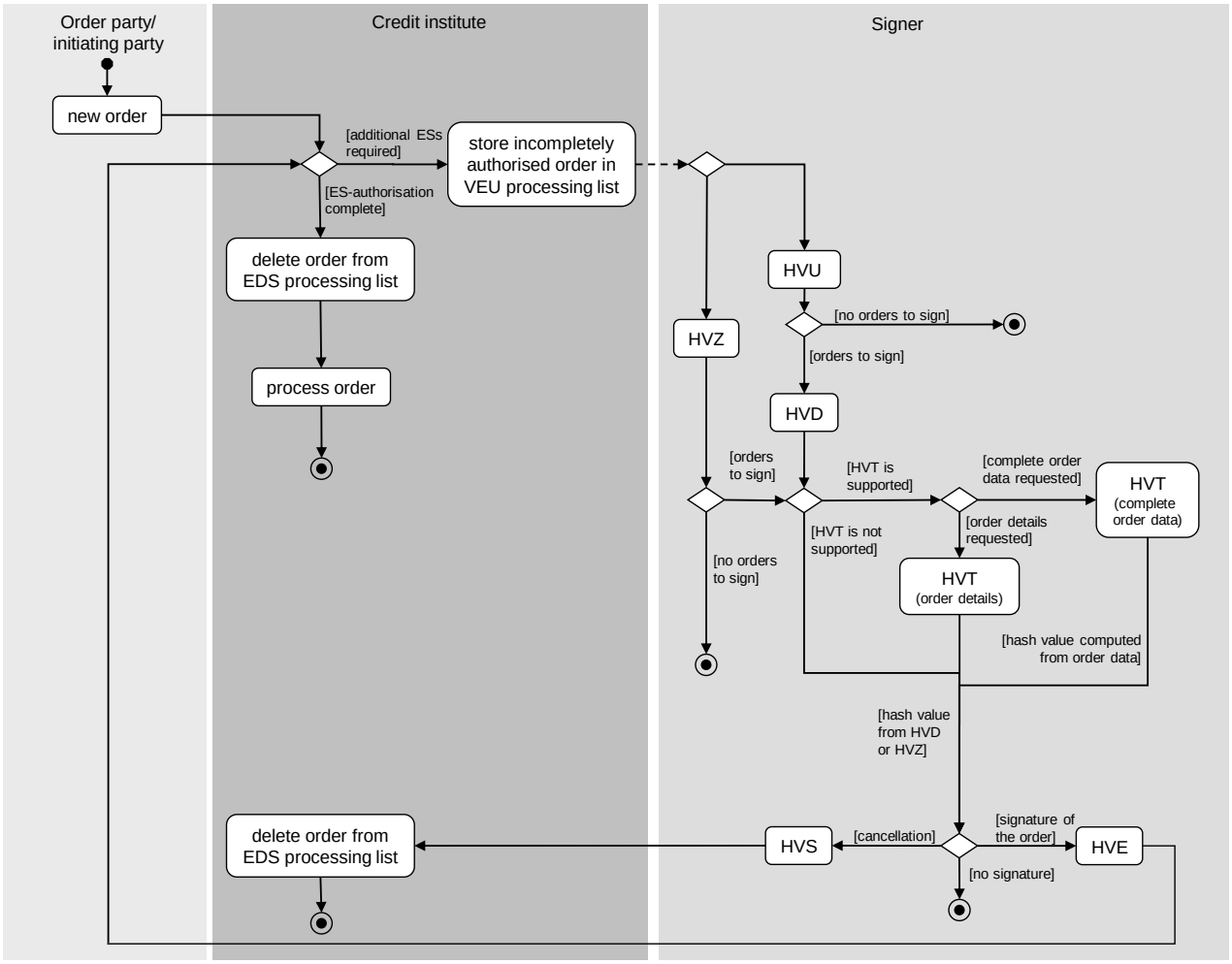


Diagram 66: Flow diagram for EDS

8.2 Technical implementation of the EDS

- A subscriber initiates EDS processing by submitting an order with an insufficient number of bank-technical signatures of the necessary authorisation class. The order is submitted in an EBICS transaction with present signature flag and a present attribute RequestEDS . In all cases, this order must be submitted with a signature (either with a bank-technical signature of class “A”, “B” or “E”, or with a transport signature of signature class “T”).

- The bank system first verifies the supplied ES(s) and the authorisation of the subscribers for the order type in question. It then compares the number and signature class of the supplied ES(s) with the locally-deposited ES requirements for the order type in question. If signatures are still outstanding, the order is placed in EDS processing together with the ES’s that have already been provided.

- Information on orders that are currently in EDS processing can be retrieved via EDS administrative order types “HVU” , “HVZ”, “HVD” and “HVT”. Necessary parameters to demarcate the original orders are transmitted via the additional order parameters `HVUOrderParams`, `HVZOrderParams`, `HVDOrderParams` or `HVTOrderParams`, which

are part of the control data for these administrative order types. “HVU”, , “HVZ” “HVD” and “HVT” are download transactions wherein the reply information is transparently embedded into the order data field in the form of XML documents. “Transparent” means: The XML structures are interpreted in binary and are compressed, encoded and coded before transmission just like the order data of other order types.

- In the case of HVE/HVS, a subscriber can retrieve the necessary data for identification of the original order (hash value of the order, order type, order number) in the following ways:

  - HVU & HVD: With HVU, they retrieve the BTF identifiers and order number, with HVD the hash value of the order. Here, the hash value originates from the ES of the subscriber that submitted the order.
  - HVZ as an alternative of HVU & HVD: With HVZ, the subscriber retrieves the BTF identifiers and order number as well as the hash value of the order. The hash value originates from the submitter's ES of the order.
  - HVU & HVT: With HVU, the subscriber retrieves the BTF identifiers and order number as in the case of „HVU & HVD". With HVT, they can set the switch `completeOrderData="true"` in the request (`HVTOrderParams`) and thus receive the complete order file. They can work out the hash value themselves from this.
  - HVZ & HVT: With HVZ, the subscriber retrieves the BTF identifiers and order number as well as the hash value of the order as described above. HVT allows him to set the switch `completeOrderData="true"` with the request (`HVTOrderParams`), thus giving him the opportunity to obtain the complete order file.
  - Via an alternative communication channel: The subscriber is at liberty to acquire the information without the help of the EBICS interface. If they already know the BTF identifiers and the order number, they can dispense with retrieval via HVU. If they also have the correct hash value for the order, retrieval via HVD, HVT or HVZ respectively can also be dispensed with.

- New ES’s can be assigned to the order via the administrative EDS order type HVE. Here, identification of the original order takes place via the additional order parameters `HVEOrderParams`, which are components of the control data for an HVE order. The ES that is to be supplied for the order data of the original order is transmitted during the initialisation step. HVE contains one or more ES(s) but no order data.

- As soon as the required number of ES’s with suitable authorisation has been submitted for the order type in question, the original order is released from EDS processing and forwarded for further order processing. In this way, the order no longer appears in the return list of orders to be signed when “HVU” (or HVZ) is next implemented.

- EDS cancellation can be initiated via administrative order type HVS. As with HVE, identification of the original order takes place via the additional order parameters (here `HVSOrderParams`), which are components of the control data for an HVS order. As with HVE, the authorising ES for the cancellation via order data of the original order is transmitted in the initialisation step. HVS also contains one or more ES(s) but no order data.
  An order cancellation is effective immediately, and always requires one single authorised signature of class “E”, “A” or “B”. A cancelled order is removed from the EDS processing; it is not forwarded for further order processing. Furthermore, it is no longer contained in the list of orders to be signed in the event of a repeated release of “HVU” (or HVZ).

8.3 Detailed description of the administrative EDS order types

This chapter will exclusively cover the differences and additions in comparison with EBICS standard orders (see Chapter 5). No more process flows will be explained (see Chapter 8.1 and 8.2), instead syntax and semantics for each individual administrative EDS order type (request and response) will be defined for the relevant elements and attributes of the XML schema, and these will be explained by way of examples.

Definition of the EDS order elements (EDS order parameters and EDS order data) is given in the XML schema “ebics_orders_H005.xsd”. Type definitions are given, in part, in the XML schema “ebics_types_H005.xsd”. With the textual representations, the relevant passages from “ebics_orders_H005.xsd” and “ebics_types_H005.xsd” are listed in summary.

8.3.1 HVU (download EDS overview) and HVZ (Download EDS overview with additional information)

A subscriber can use HVU to list the orders for which they are authorised as a signatory. As a filter criterion, they can restrict the list in “request” to specific (groups of) business transactions (`ServiceFilter`, 0..n occurences). In addition to the order designation, the “response” also contains the size of the order data, signature conditions and information on the initiating party and the previous signatories (`OrderDetails`).

Apart from all information of HVU the response message of HVZ also contains data of HVD. Therefore, HVZ ("Download EDS overview with additional information") may be compared to a combination of HVU with 1 to n HVDs.

HVU and HVZ are administrative order types of the type “download”.

8.3.1.1 HVU request

In the HVU request, the subscriber optionally submits a filter criterion. Only orders whose BTF identifiers are contained in the submitted filter are returned. If the subscriber does not submit a restriction, they will receive a list of all BTF identifiers for which they are authorised as a signatory.

Characteristics of `OrderParams` (order parameters) for HVU: `HVUOrderParams`

8.3.1.1.1 XML schema (graphical representation)

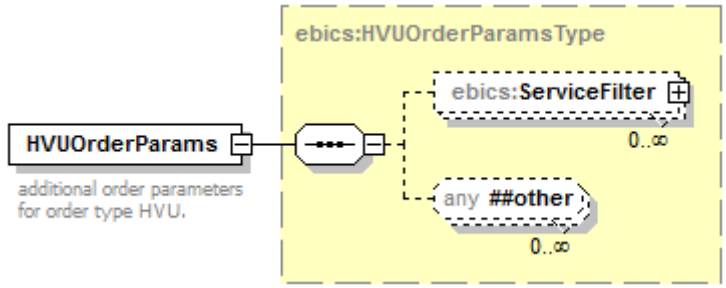


Diagram 67: HVUOrderParams

8.3.1.1.2 Meaning of the XML elements/attributes

XML element/ attribute	Data type	#	Meaning	Example
HVUOrderParams	ebics:HVUOrderParamsType (complex)	1	Order parameters for administrative order type HVU	- (complex)
ServiceFilter	ebics:ServiceType for this structure refer to chapter 8.3.6	0..n	Choose all orders which map with a specific filter of BTF elements, for which orders available for signature are to be retrieved; if not specified, all orders are retrieved for which the subscriber is authorised as a signatory	

8.3.1.1.3 Example XML (abridged)

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
 xmlns="urn:org:ebics:H005"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_request_H005.xsd"
 Version="H005" Revision="1">
  <header authenticate="true">
    <static>
      <!-- […] -->
      <OrderDetails>
        <AdminOrderType>HVU</AdminOrderType>

        <HVUOrderParams>
          <ServiceFilter>
              <ServiceName>SCT</ServiceName>
          </ServiceFilter>
        </HVUOrderParams>
      </OrderDetails>
      <!-- […] -->
    </static>
    <!-- […] -->
```

```
  </header>
  <!-- […] -->
</ebicsRequest>
```

8.3.1.2 HVU response

In the HVU response, the subscriber is given information as to the orders for which they are authorised as signatories.

Characteristics of the (decoded & decrypted & decompressed) `OrderData` (order data) for HVU: `HVUResponseOrderData`

8.3.1.2.1 XML schema (graphic representation)

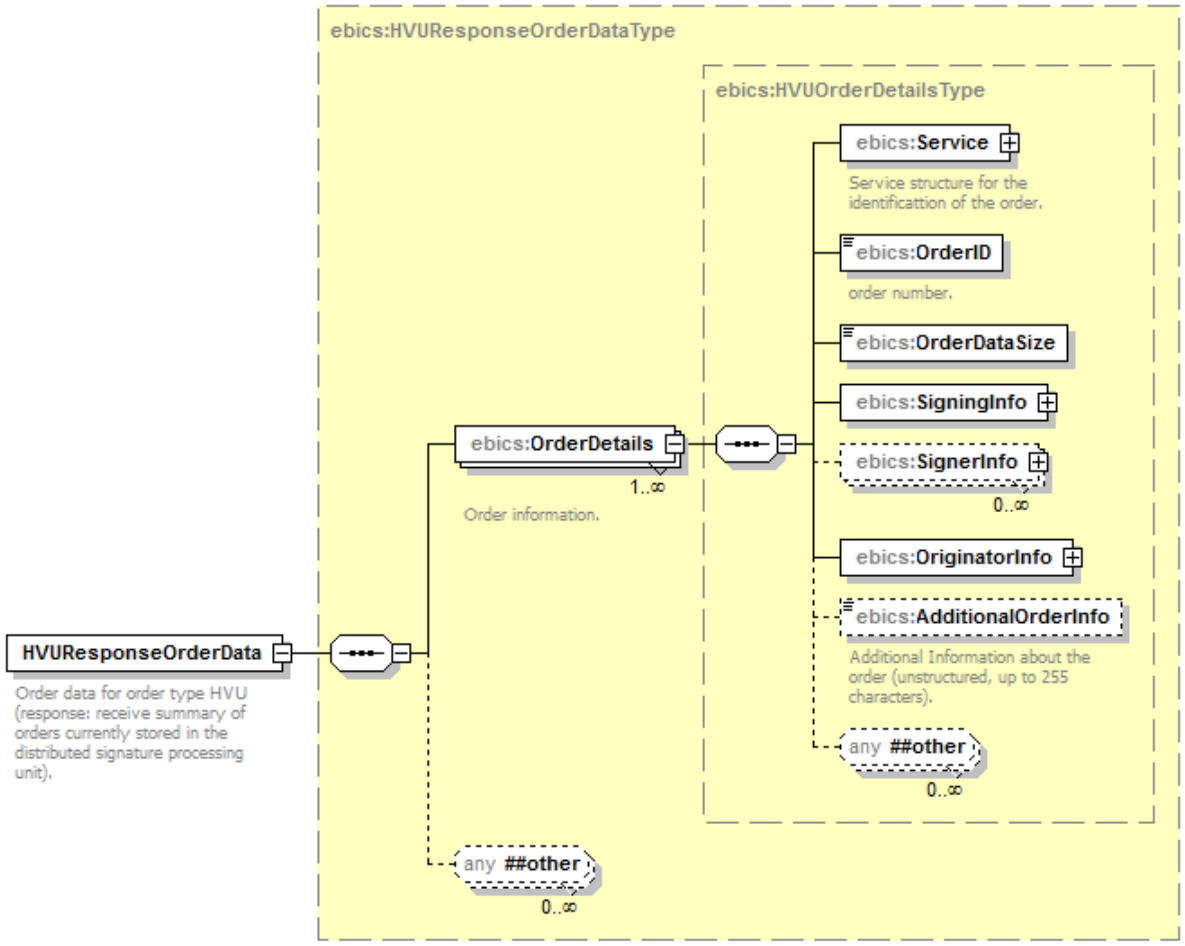


Diagram 68: HVUResponseOrderData

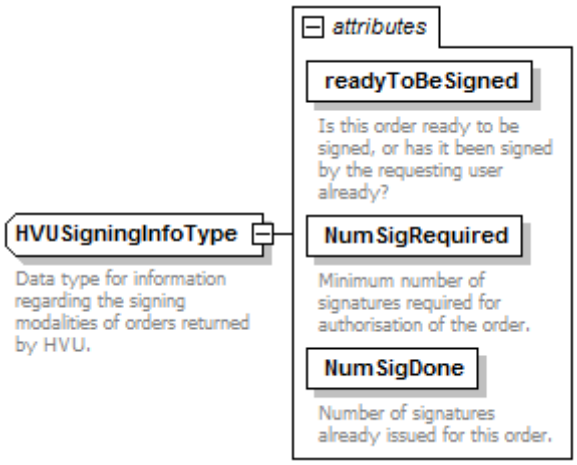


Diagram 69: HVUSigningInfoType (to SigningInfo)

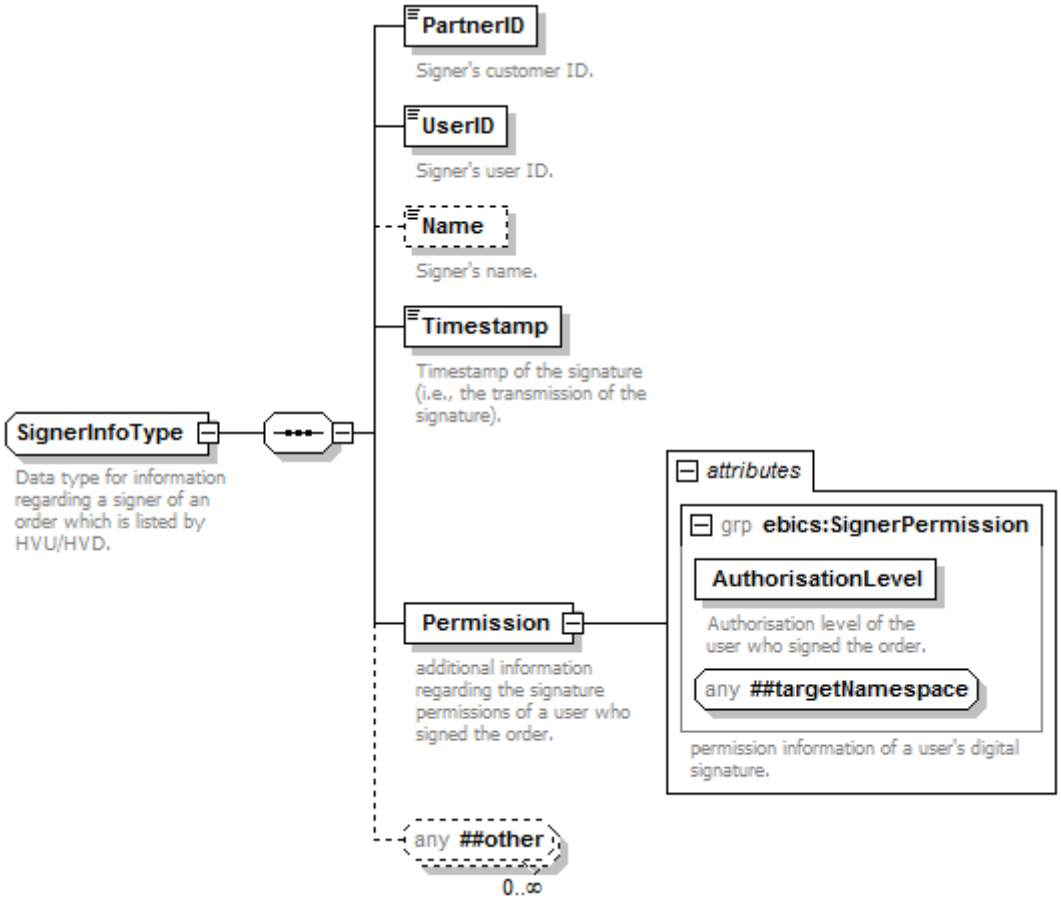


Diagram 70: SignerInfoType (to SignerInfo)

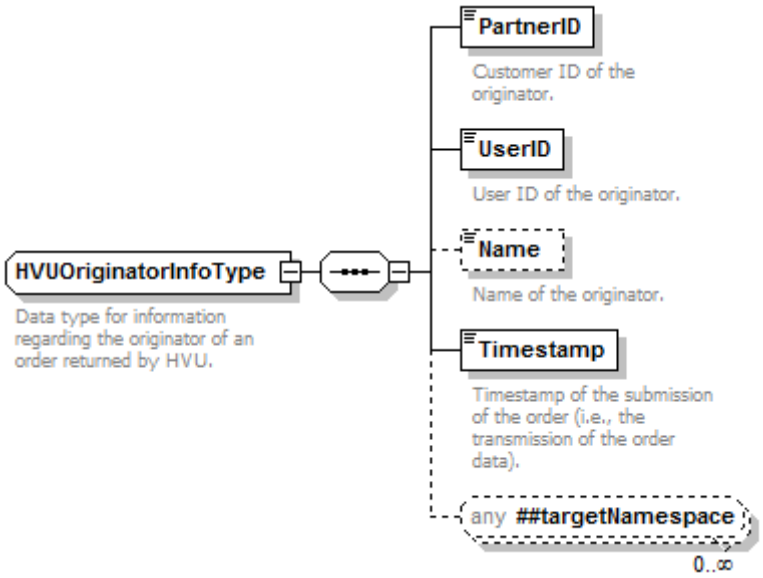


Diagram 71: HVUOriginatorInfoType (to OriginatorInfo)

8.3.1.2.2 Meaning of the XML elements/attributes

XML element/ attribute	Data type	#	Meaning	Example
`HVUResponse» OrderData`	`ebics:HVUResponse» OrderDataType` (complex)	1	XML order data for administrative order type HVU	- (complex)
`OrderDetails`	`ebics:HVUOrder» DetailsType` (complex)	1..∞	Order information for administrative order type HVU	- (complex)
`Service`	`ebics:RestrictedServ iceType` for this structure refer to chapter 8.3.6	1	Kind of business transaction, identified by the service structure, submitted for EDS	
`OrderID`	`ebics:OrderIDType (→token, fixLength=4)`	1	Order number of the order submitted for EDS	“OR01”
`OrderDataSize`	`positiveInteger`	1	Size of the uncompressed order data of the order submitted for EDS in bytes	123456
`SigningInfo`	`ebics:HVUSigning» InfoType` (complex)	1	Information on the signature modalities	- (complex)
`SigningInfo» @readyToBeSigned`	`boolean`	1	Is the order ready for signature (true) or already signed by the subscriber (false)?	“true”
`SigningInfo» @NumSigRequired`	`positiveInteger`	1	Total number of ES’s required for activation	4

SigningInfo» @numSigDone	nonNegativeInteger	1	Number of ES's already provided	2
SignerInfo	ebics:SignerInfo» Type (complex)	0..∞	Information on previous signatories	- (complex)
PartnerID (in SignerInfo)	ebics:PartnerIDType (→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})	1	Customer ID of the signatory	"CUSTM001"
UserID (in SignerInfo)	ebics:UserIDType (→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})	1	Subscriber ID of the signatory	"USR100"
Name (in SignerInfo)	ebics:NameType (→normalizedString)	0..1	Signatory's name	"John Doe"
Timestamp (in SignerInfo)	ebics:TimestampType (→dateTime)	1	Time stamp of the signature (i.e. transmission of the signature)	"2020-11-26T» 16:30:45.123Z"
Permission	– (complex)	1	Additional authorisation information relating to the subscriber that acted as signatory	- (complex)
Permission» @Authorisation» Level	ebics:Authorisation» LevelType (→token, length=1: "E", "A", "B", "T")	1	Signature authorisation of the subscriber that acted as signatory	"A"
OriginatorInfo	ebics:HVUOriginator» InfoType (complex)	1	Information on the initiating party	- (complex)
PartnerID (in OriginatorInfo)	ebics:PartnerIDType (→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})	1	Customer ID of the initiating party	"CUSTM001"
UserID (in OriginatorInfo)	ebics:UserIDType (→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})	1	Subscriber ID of the initiating party	"USR300"
Name (in OriginatorInfo)	ebics:NameType (→normalizedString)	0..1	Name of the initiating party	"Ophelia Originator"
Timestamp (in OriginatorInfo)	ebics:TimestampType (→dateTime)	1	Time stamp of the submission (i.e. trans-mission of the order file)	"2020-11-25T» 15:30:45.123Z"
AdditionalOrderInfo	Max255Text	0..1	Additonal Information about the order up to 255 characters (given by the customer)	e.g. local file name

8.3.1.2.3 Example XML

```
<?xml version="1.0" encoding="UTF-8"?>
<HVUResponseOrderData
 xmlns="urn:org:ebics:H005"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_orders_H005.xsd">
 <OrderDetails>
   <Service>
       <ServiceName>SCT</ServiceName>
       <MsgName>pain.001</MsgName>
   </Service>
   <OrderID>OR01</OrderID>
   <OrderDataSize>123456</OrderDataSize>
   <SigningInfo NumSigRequired="4" readyToBeSigned="true" NumSigDone="2"/>
   <SignerInfo>
     <PartnerID>CUSTM001</PartnerID>
     <UserID>USR100</UserID>
     <Name>John Doe</Name>
     <Timestamp>2020-11-26T16:30:45.123Z</Timestamp>
     <Permission AuthorisationLevel="A"/>
   </SignerInfo>
   <SignerInfo>
     <PartnerID>CUSTM002</PartnerID>
     <UserID>USR200</UserID>
     <Name>Jackie Smith</Name>
     <Timestamp>2020-11-26T17:30:45.123Z</Timestamp>
     <Permission AuthorisationLevel="B"/>
   </SignerInfo>
   <OriginatorInfo>
     <PartnerID>CUSTM001</PartnerID>
     <UserID>USR300</UserID>
     <Name>Ophelia Originator</Name>
     <Timestamp>2020-11-25T15:30:45.123Z</Timestamp>
   </OriginatorInfo>
 </OrderDetails>
</HVUResponseOrderData>
```

8.3.1.3 HVZ request

In the HVZ request, the subscriber optionally submits a list of BTF identifiers as a filter criterion. Only orders whose order type is contained in the submitted list are returned. If the subscriber does not submit an order type list as a restriction, they will receive a list of all order types for which they are authorised as a signatory.

Characteristics of `OrderParams` (order parameters) for HVZ: `HVUOrderParams`

8.3.1.3.1 XML schema (graphical representation)

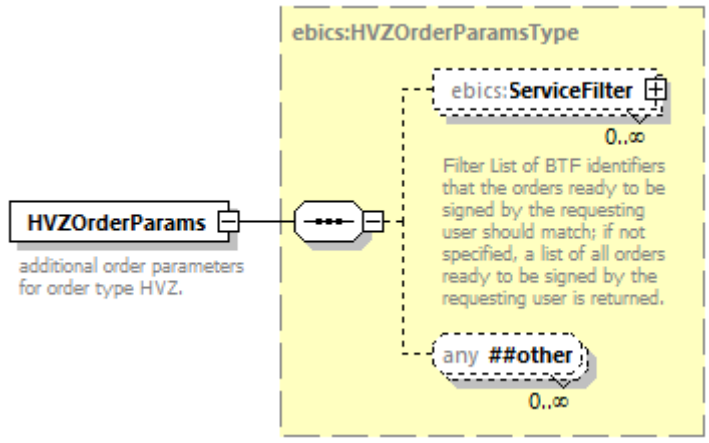


Diagram 72: HVZOrderParams

8.3.1.3.2 Meaning of the XML elements/attributes

XML element/ attribute	Data type	#	Meaning	Example
`HVZOrderParams`	`ebics:HVZOrderParamsType` (complex)	1	Order parameters for administrative order type HVZ	- (complex)
`ServiceFilter`	`ebics: ServiceType` for this structure refer to chapter 8.3.6	0..1	Choose all orders which map with a specific filter of BTF elements, for which orders available for signature are to be retrieved; if not specified, all orders are retrieved for which the subscriber is authorised as a signatory	

8.3.1.3.3 Example XML (abridged)

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
 xmlns="urn:org:ebics:H005"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_request_H005.xsd"
 Version="H005" Revision="1">
  <header authenticate="true">
    <static>
      <!-- […] -->
      <OrderDetails>
        <AdminOrderType>HVZ</AdminOrderType>
                <HVZOrderParams>
          <ServiveFilter>…</ServiceFilter>
        </HVZOrderParams>
      </OrderDetails>
```

```
      <!-- […] -->
    </static>
    <!-- […] -->
  </header>
  <!-- […] -->
</ebicsRequest>
```

8.3.1.4 HVZ response

In the HVZ response, the subscriber is given information as to the orders for which they are authorised as signatories.

HVZResponseOrderData contains the complete information of HVUResponseOrderData and HVDResponseOrderData with the exception of the element "DisplayFile" containing the file display. As with HVD, the order's hash value is extracted from the ES of the first signatory of the order and is recalculated if the subscriber executing HVZ uses a different signature process. In order to make this evident, the hash value is provided with an attribute containing the signature process used.
Only for payment orders additional information of the file display is returned if available:

- total transaction amount for all logical files
- total transaction number for all logical files
- currency (only if identical across all transactions, skip otherwise)

For DTAUS/DTAZV: Ordering party, account number / IBAN and bank code / BIC of the first transaction of the first logical file
Characteristics of the (decoded & decrypted & decompressed) `OrderData` (order data) for HVZ: `HVZResponseOrderData`

8.3.1.4.1 XML-Schema (graphic representation)

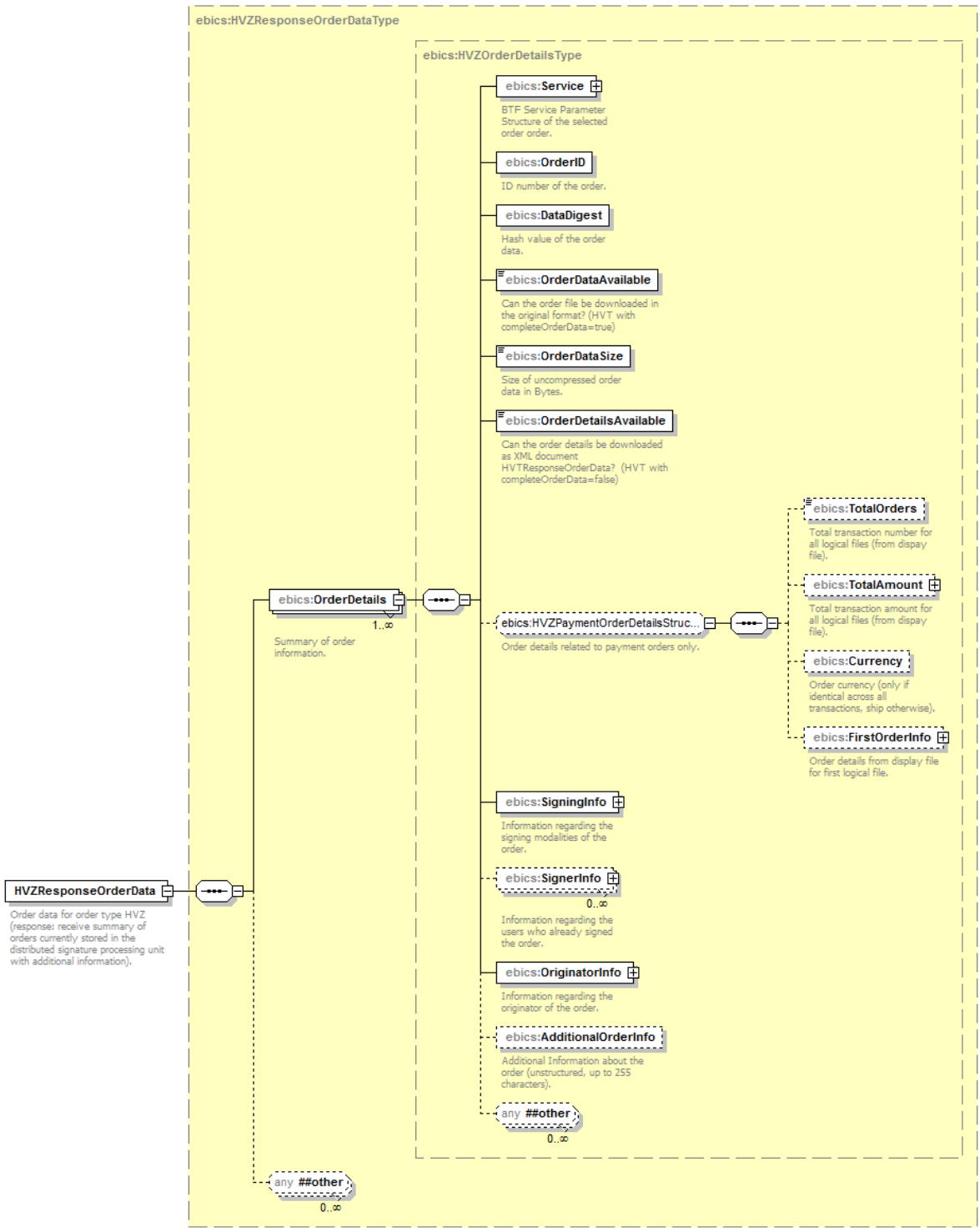


Diagram 73: HVZResponseOrderData

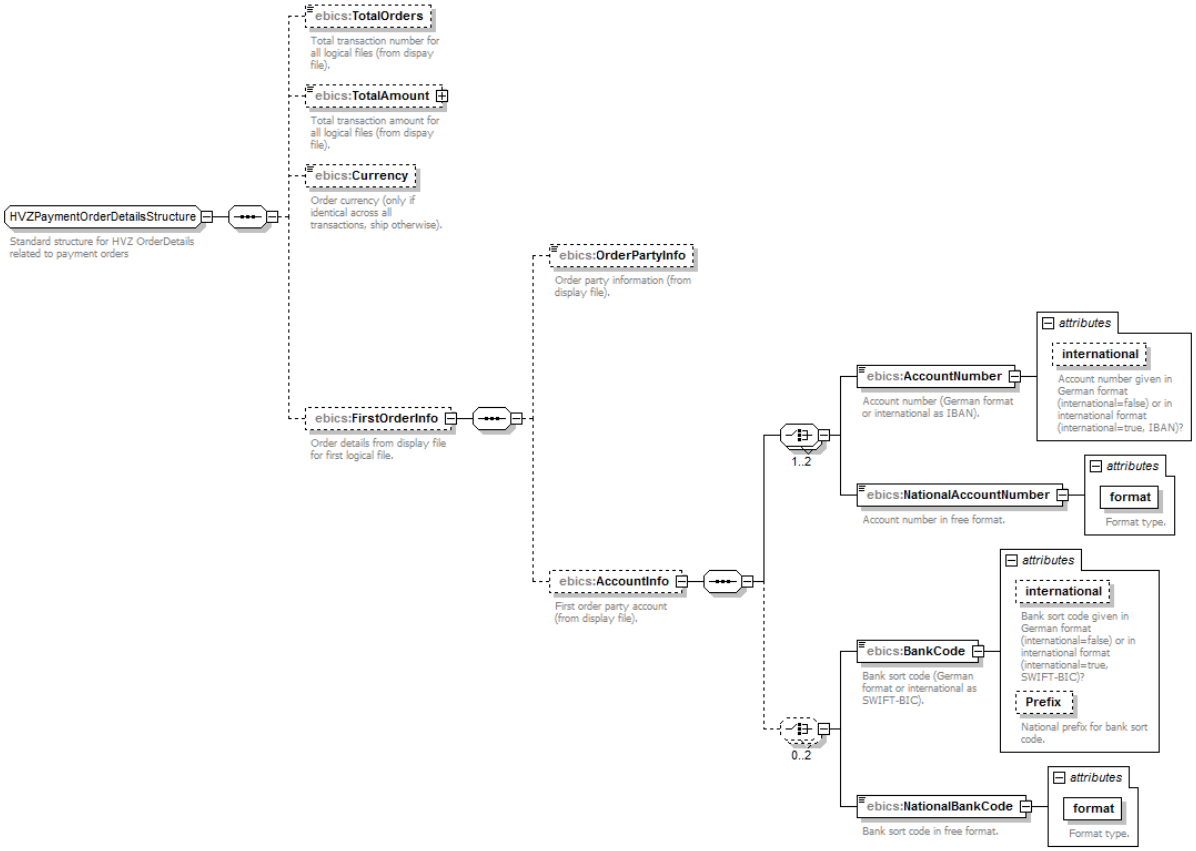


Diagram 74 HVZPaymentOrderDetailsStructure

8.3.1.4.2 Meaning of the XML elements/attributes

XML element/ attribute	Data type	#	Meaning	Example
`HVZResponse» OrderData`	`ebics:HVZResponse» OrderDataType` (complex)	1	XML order data for administrative order type HVZ	- (complex)
`OrderDetails`	`ebics:HVUOrder» DetailsType` (complex)	1..∞	Order information for administrative order type HVZ	- (complex)
`Service`	`ebics:RestrictedServ iceType` for this structure refer to chapter 8.3.6	1	Kind of business transaction, identified by the service structure, submitted for EDS	
`OrderID`	`ebics:OrderIDType (→token, maxLength=4)`	1	Order number of the order submitted for EDS	“OR01”
`DataDigest`	`ebics:DigestType (→dsig:DigestValue» Type →base64Binary)`	1	Hash value from the ES for the signature process used by the subscriber according to the	- (base64 data)

			Request-Element User-ID	
DataDigest» @SignatureVersion	ebics:Signature»VersionType (→token, length=4, pattern="A\d{3}"		Version of the signature process used by the subscriber according to the Request-Element UserID	e.g. „A005“
OrderDataAvailable	boolean	1	Can the order file be downloaded in the original format? (HVT with completeOrderData=true)	true
OrderDataSize	positiveInteger	1	Size of the uncompressed order data of the order submitted for EDS in bytes	123456
OrderDetails» Available	boolean	1	Can the order details be downloaded as XML document HVTResponseOrderData? (HVT with completeOrderData=false)	true
TotalOrders	nonNegativeInteger	0..1	Total transaction number for all logical files (from dispay file).	15
TotalAmount	ebics:AmountValue» Type (→decimal, totalDigits=24, fractionDigits=4)	0..1	Total transaction amount for all logical files (from dispay file).	129.00
TotalAmount» @isCredit	boolean	0..1	Flag for differentiation between credit notes (isCredit="true") and debit notes (isCredit="false"). (optional use; usable if identical across all transactions, skip otherwise).	“false“
Currency	ebics:CurrencyBase» Type (→token, length=3, pattern="[A-Z]{3}")	0..1	Order currency (only if identical across all transactions, skip otherwise).	„USD“
FirstOrderInfo	(complex)	0..1	Order details from display file for first logical file.	- (complex)
OrderPartyInfo	normalizedString	0..1	Order party information (from display file).	„Arnold Smith“

AccountInfo	complex	0..1		- (complex)
–	–	1..2	Information about the account number: AccountNumber and/or NationalAccountNumber	-
AccountNumber	ebics:AccountNumber»Type (→token, maxLength=40, pattern="\d{3,10}|([A-Z]{2}\d{2}[A-Za-z0-9]{3,30}")	1	Account number (German format or international format = IBAN)	„12345678“
AccountNumber»@international	boolean	0..1	Account number given in German format (international=false) or in international format (international=true, IBAN)? Default="false"	„false“
NationalAccount»Number	ebics:National»AccountNumberType (→token, maxLength=40)	1	Account number in free format (neither German nor IBAN)	„1234567890123456“
NationalAccount»Number@format	token	1	format type	„other“
–	–	1..2	Information about Bank sort code: BankCode and/or NationalBankCode	-
BankCode	ebics:BankCodeType (→token, maxLength=11, pattern="\d{8}|([A-Z]{6}[A-Z0-9]{2}([A-Z0-9]{3})?)")	0..1	German Format or international format (= SWIFT-BIC).	„50010060“ Note: Element cannot be provided in case of IBAN Only
BankCode»@international	boolean	0..1	Bank sort code given in German format (BankCode»@international="false") or in international format (BankCode»@international="true", SWIFT-BIC)? Default="false"	„false“
NationalBankCode	ebics:National»BankCodeType (→token, maxLength=30)	1	Bank sort code in free format (neither German format nor SWIFT-BIC)	„123456789012“
NationalBankCode»@format	token	1	format type	„other“

SigningInfo	ebics:HVUSigning» InfoType (complex)	1	Information on the signature modalities	- (complex)
SigningInfo» @readyToBeSigned	boolean	1	Is the order ready for signature (true) or already signed by the subscriber (false)?	“true”
SigningInfo» @NumSigRequired	positiveInteger	1	Total number of ES’s required for activation	4
SigningInfo» @numSigDone	nonNegativeInteger	1	Number of ES’s already provided	2
SignerInfo	ebics:SignerInfo» Type (complex)	0..∞	Information on previous signatories	- (complex)
PartnerID (in SignerInfo)	ebics:PartnerIDType (→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})	1	Customer ID of the signatory	“CUSTM001”
UserID (in SignerInfo)	ebics:UserIDType (→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})	1	Subscriber ID of the signatory	“USR100“
Name (in SignerInfo)	ebics:NameType (→normalizedString)	0..1	Signatory’s name	“John Doe“
Timestamp (in SignerInfo)	ebics:TimestampType (→dateTime)	1	Time stamp of the signature (i.e. transmission of the signature)	“2020-11-26T» 16:30:45.123Z“
Permission	– (complex)	1	Additional authorisation information relating to the subscriber that acted as signatory	- (complex)
Permission» @Authorisation» Level	ebics:Authorisation» LevelType (→token, length=1: "E", "A", "B", "T")	1	Signature authorisation of the subscriber that acted as signatory	“A“
OriginatorInfo	ebics:HVUOriginator» InfoType (complex)	1	Information on the initiating party	- (complex)
PartnerID (in OriginatorInfo)	ebics:PartnerIDType (→token, maxLength= 35, pattern="[a-zA-Z0-9,=]{1,35})	1	Customer ID of the initiating party	“CUSTM002”
UserID (in OriginatorInfo)	ebics:UserIDType (→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})	1	Subscriber ID of the initiating party	“USR300“
Name (in OriginatorInfo)	ebics:NameType (→normalizedString)	0..1	Name of the initiating party	“Ophelia Originator“
Timestamp (in OriginatorInfo)	ebics:TimestampType (→dateTime)	1	Time stamp of the submission (i.e. trans-mission of the order file)	“2020-11-25T» 15:30:45.123Z“

AdditionalOrderInfo	Max255Text	0..1	Additonal Information about the order up to 255 characters (given by the customer)	e.g. local file name

8.3.1.4.3 Example XML

```
<?xml version="1.0" encoding="UTF-8"?>
<HVZResponseOrderData xmlns="urn:org:ebics:H005"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:org:ebics:H005 ebics_orders_H005.xsd">
   <OrderDetails>

      <Service>
        <ServiceName>SCT</ServiceName>
        <MsgName>pain.001</MsgName>
      </Service>
      <OrderID>OR01</OrderID>
      <DataDigest SignatureVersion="A006">MTIzNDU2Nzg5MDEyMzQ1Njc4OTAxMjM0NTY3ODkwMTI=
</DataDigest>
      <OrderDataAvailable>true</OrderDataAvailable>
      <OrderDataSize>123456</OrderDataSize>
      <OrderDetailsAvailable>true</OrderDetailsAvailable>
      <TotalOrders>22</TotalOrders>
      <TotalAmount>500.00</TotalAmount>
      <Currency>EUR</Currency>
      <FirstOrderInfo>
         <OrderPartyInfo>Arnold Auftraggeber</OrderPartyInfo>
         <AccountInfo>
            <AccountNumber international="true">
               DE68210501700012345678
            </AccountNumber>
            <BankCode international="false" Prefix="DE">
               21050170
            </BankCode>
         </AccountInfo>
      </FirstOrderInfo>
      <SigningInfo NumSigRequired="4" readyToBeSigned="true"
         NumSigDone="2" />
      <SignerInfo>
         <PartnerID>PARTNER1</PartnerID>
         <UserID>USER0001</UserID>
         <Name>Max Mustermann</Name>
         <Timestamp>2020-11-26T16:30:45.123Z</Timestamp>
         <Permission AuthorisationLevel="A" />
      </SignerInfo>
      <SignerInfo>
         <PartnerID>PARTNER2</PartnerID>
         <UserID>USER0002</UserID>
         <Name>Maxime Musterfrau</Name>
         <Timestamp>2020-11-26T17:30:45.123Z</Timestamp>
         <Permission AuthorisationLevel="B" />
      </SignerInfo>
      <OriginatorInfo>
         <PartnerID>PARTNER1</PartnerID>
         <UserID>USER0001</UserID>
         <Name>Erich Einreicher</Name>
         <Timestamp>2020-11-25T15:30:45.123Z</Timestamp>
      </OriginatorInfo>
   </OrderDetails>
</HVZResponseOrderData>
```

8.3.2 HVD (retrieve EDS state)

With HVD, a subscriber can retrieve the state of an order that is currently in EDS processing and for which the subscriber is authorised as a signatory. They receive information about the

order in the form of an electronic accompanying note (`DisplayFile`) and the order hash value (`DataDigest`) as well as the previous signatories (`SignerInfo`). The bank system has extracted the order's hash value from the ES of the first signatory of the order or it is recalculated if the subscriber executing HVD is using a different signature process. The data of the accompanying note MUST correspond in terms of contents with the order data, the hash value of which is also delivered.

The bank system has to verify whether the subscriber possesses a bank-technical authorisation of signature (signature class E, A or B) for the order on hand and the order is still in the signature folder. If the authorisation is missing, the transaction has to be cancelled and the error code EBICS_DISTRIBUTED_SIGNATURE_AUTHORISATION_FAILED is issued.

- In case of some underlying administrative order types / business transactions, detailed information on a specific order in the EDS processing system cannot be retrieved by means of the transaction HVT. Whether this is possible for the ongoing order or not, is signalized in the HVD response by the bank system. Before the execution of HVD, the bank system verifies whether the order is currently located in the EDS processing system and, in case of an error, terminates the transaction returning the business related error code EBICS_ORDERID_UNKNOWN.

HVD is an administrative order type of type "download".

8.3.2.1 HVD request

In the HVD request, the subscriber transfers the relevant data for identification of the order for which they want to retrieve the EDS state.

Characteristics of `OrderParams` (order parameters) for HVD: `HVDOrderParams`

8.3.2.1.1 XML schema (graphical representation)

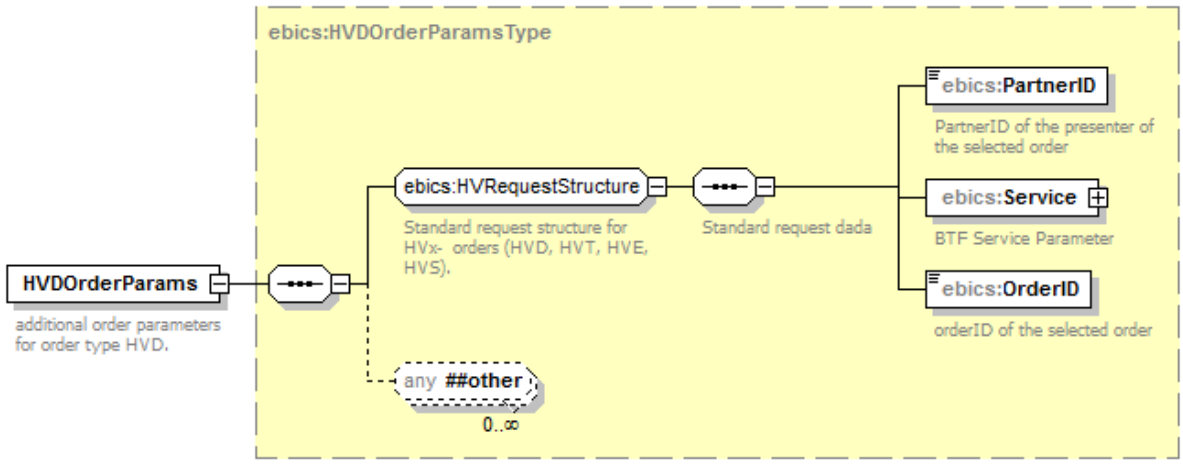


Diagram 75: HVDOrderParams

8.3.2.1.2 Meaning of the XML elements/attributes

XML element/ attribute	Data type	#	Meaning	Example
HVDOrderParams	ebics:HVDOrderParamsType (complex)	1	Order parameters for order type HVD	- (complex)
PartnerID	ebics:PartnerIDType (→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})	1	Customer ID of the initiating party	“CUSTM001”
Service	ebics:RestrictedServiceType for this structure refer to chapter 8.3.6	1	Kind of business transaction, identified by the service structure, submitted for EDS	
OrderID	ebics:OrderIDType (→token, fixLength=4)	1	Order number of the order submitted for EDS	“OR01”

8.3.2.1.3 Example XML (abridged)

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
 xmlns="urn:org:ebics:H005"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_request_H005.xsd"
 Version="H005" Revision="1">
  <header authenticate="true">
    <static>
      <!-- […] -->
      <OrderDetails>
        <AdminOrderType>HVD</AdminOrderType>
                <HVDOrderParams>
          <PartnerID>PARTNER1</PartnerID>
                  <Service>
              <ServiceName>SCT</ServiceName>
              <MsgName>pain.001</MsgName>
          </Service>
          <OrderID>OR01</OrderID>
        </HVDOrderParams>
      </OrderDetails>
      <!-- […] -->
    </static>
    <!-- […] -->
  </header>
  <!-- […] -->
</ebicsRequest>
```

8.3.2.2 HVD response

The HVD response contains EDS information relating to the order that the subscriber has requested in the HVD request. In particular, the hash value of the order data is returned from

the ES of the first signatory, along with the accompanying note. In addition, the information is contained whether the bank system supports the transaction HVT for the particular order.
The following distinction is made:

- `OrderDataAvailable` : Download of the complete order file wíth HVT and `completeOrderData=true` possible?

- `OrderDetailsAvailable` : Download of the edited order details in XML format with HVT and `completeOrderData=false` possible?

The HVD response provides the subscriber with all data that they require for acknowledgement of the order via HVE or cancellation via HVS.

Characteristics of the (decoded & decrypted & decompressed) `OrderData` (order data) for HVD: `HVDResponseOrderData`

8.3.2.2.1 XML schema (graphical representation)

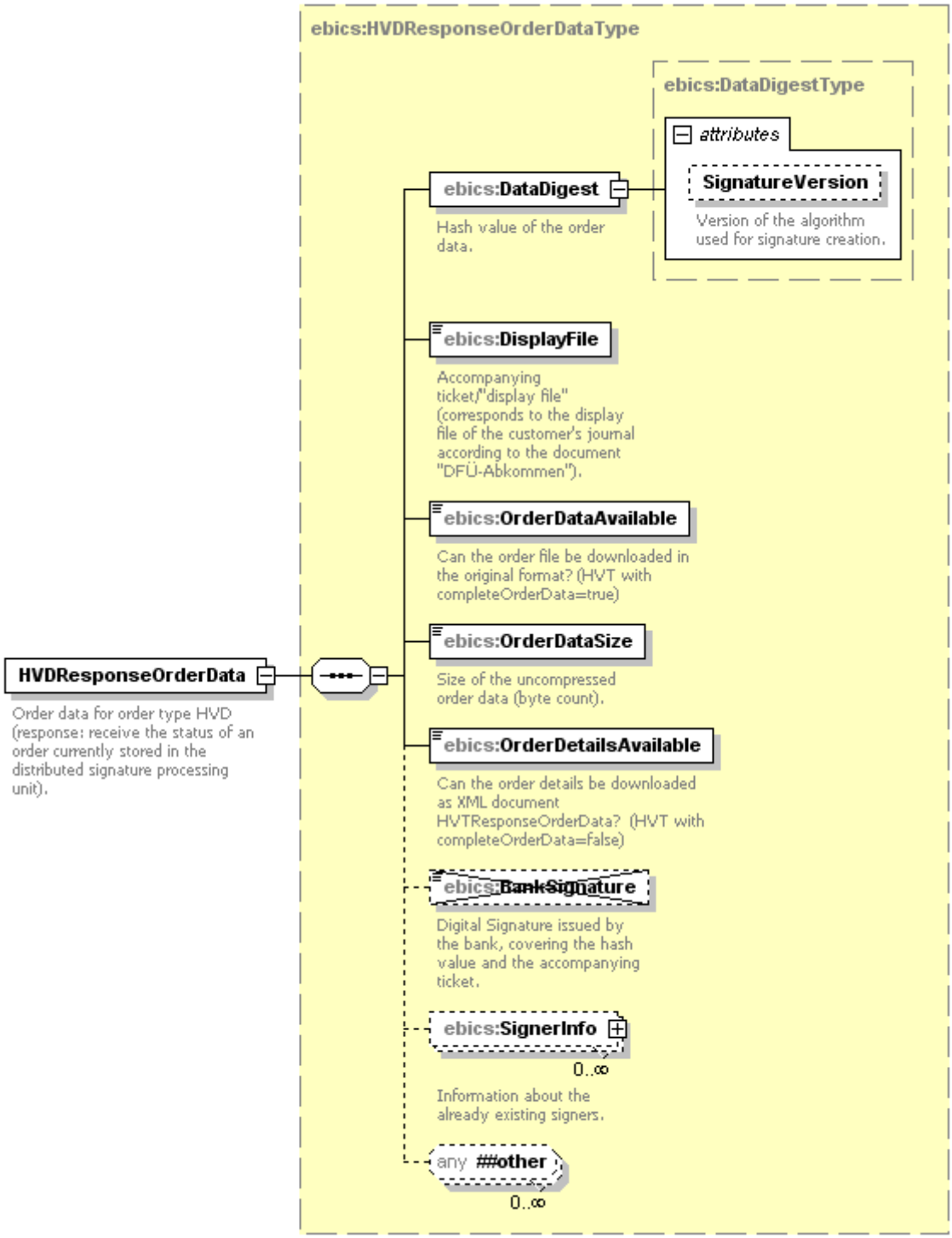


Diagram 76: HVDResponseOrderData

8.3.2.2.2 Meaning of the XML elements/attributes

XML element/ attribute	Data type	#	Meaning	Example
`HVDResponseOrderData`	`ebics:HVDResponse`	1	XML order data for order	-

	» OrderDataType (complex)		type HVD	(complex)
DataDigest	ebics:DigestType (→dsig:DigestValue» Type →base64Binary)	1	Hash value of the order for the signature process used by the subscriber according to the Request-Element UserID	- (base64 data)
DataDigest» @SignatureVersion	ebics:Signature»VersionType (→token, length=4, pattern="A\d{3}"		Version for the signature process used by the subscriber according to the Request-Element UserID	e.g. „A006“
DisplayFile	base64Binary	1	Accompanying note/“display file“ for submitted order	- (base64 data)
OrderDataAvailable	boolean	1	Can the order file be downloaded in the original format? (HVT with completeOrderData=true)	true
OrderDataSize	positiveInteger	1	Size of the uncompressed order data (byte count)	1280
OrderDetailsAvailable	boolean	1	Can the order details be downloaded as XML document HVTResponseOrderData? (HVT with completeOrderData=false)	true
BankSignature	ebics:SignatureType (→base64Binary)	0	ES of the financial institution via hash value and accompanying note, planned feature	- (base64 data)
SignerInfo	ebics:SignerInfo» Type (complex)	0..∞	Information on previous signatories	- (complex)

For the remaining XML elements and attributes: See administrative order type HVU (Chapter 8.3.1.2).

8.3.2.2.3 Example XML

```
<?xml version="1.0" encoding="UTF-8"?>
<HVDResponseOrderData
 xmlns="urn:org:ebics:H005"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_orders_H005.xsd">
 <DataDigest SignatureVersion="A006">
MTIzNDU2Nzg5MDEyMzQ1Njc4OTAxMjM0NTY3ODkwMTI=</DataDigest>
 <DisplayFile>…</DisplayFile>
 <OrderDataAvailable>true</OrderDataAvailable>
 <OrderDataSize>1280</OrderDataSize>
 <OrderDetailsAvailable>true</OrderDetailsAvailable>
```

```
 <SignerInfo>
   <PartnerID>CUSTM001</PartnerID>
   <UserID>USR100</UserID>
   <Name>John Doe</Name>
   <Timestamp>2005-01-31T16:30:45.123Z</Timestamp>
   <Permission AuthorisationLevel="A"/>
 </SignerInfo>
 <SignerInfo>
   <PartnerID>CUSTM002</PartnerID>
   <UserID>USR200</UserID>
   <Name>Jackie Smith</Name>
   <Timestamp>2005-01-31T17:30:45.123Z</Timestamp>
   <Permission AuthorisationLevel="B"/>
 </SignerInfo>
</HVDResponseOrderData>
```

8.3.3 HVT (retrieve EDS transaction details)

HVT provides the subscriber with detailed information about an order from EDS processing for which the subscriber is authorised as a signatory. Depending on the request (`OrderFlags@completeOrderData`), they either receive the complete order file or account details, implementation deadline, amounts and other descriptions (`OrderInfo`).

The subscriber can transmit other filter criteria (e.g. for selection of individual orders within an overall order) via request in the generic key value structure (`Parameter`).

In the case of some administrative order types / business transactions, it is not possible to retrieve detailed information by means of `OrderFlags@completeOrderData="false"`. In this case, the bank system returns the business related error code EBICS_UNSUPPORTED_REQUEST_FOR_ORDER_INSTANCE. With `OrderDataAvailable` and `OrderDetailsAvailable` in the HVD response, the bank system signals if an HVT transaction for a specific order within the EDS administration can be executed.
Before the execution of HVT, the bank system verifies whether the order is currently located in the EDS processing system and, in case of an error, terminates the transaction returning the business related error code EBICS_ORDERID_UNKNOWN.

The bank system has to verify whether the subscriber possesses a bank-technical authorisation of signature (signature class E, A or B) for the order on hand and the order is still in the signature folder. If the authorisation is missing, the transaction has to be cancelled and the error code EBICS_DISTRIBUTED_SIGNATURE_AUTHORISATION_FAILED is issued.

HVT is an administrative order type of the type “download”.

8.3.3.1 HVT request

In the HVT request, the subscriber specifies the order for which they want to retrieve the EDS transaction details. In addition, they decide whether they want to have order details (`completeOrderData="false"`) or the complete order file (`completeOrderData="true"`) as a response by setting the OrderFlag `completeOrderData`.

If `completeOrderData="false"`, the customer system may limit the number of order details that the bank system is to provide. By means of the attribute `fetchLimit` for the element `OrderFlags` the maximum number of order details to be transmitted can be defined (a proposal for that is `fetchLimit`=100). If `fetchLimit`=0, all order details of an order are requested.
By means of the attribute `fetchOffset` the customer system is able to define an offset position in the original order file. From this position onwards the order details are returned. If `fetchOffset=0`, order details are requested from the starting point of the order file.
If the value for `fetchOffset` is higher than the total number of order details, the business related error EBICS_INVALID_ORDER_PARAMS is returned.

The generic key value structure (`Parameter`) is available for further filter criteria.

Characteristics of `OrderParams` (order parameters) for HVT: `HVTOrderParams`

8.3.3.1.1 XML schema (graphical representation)

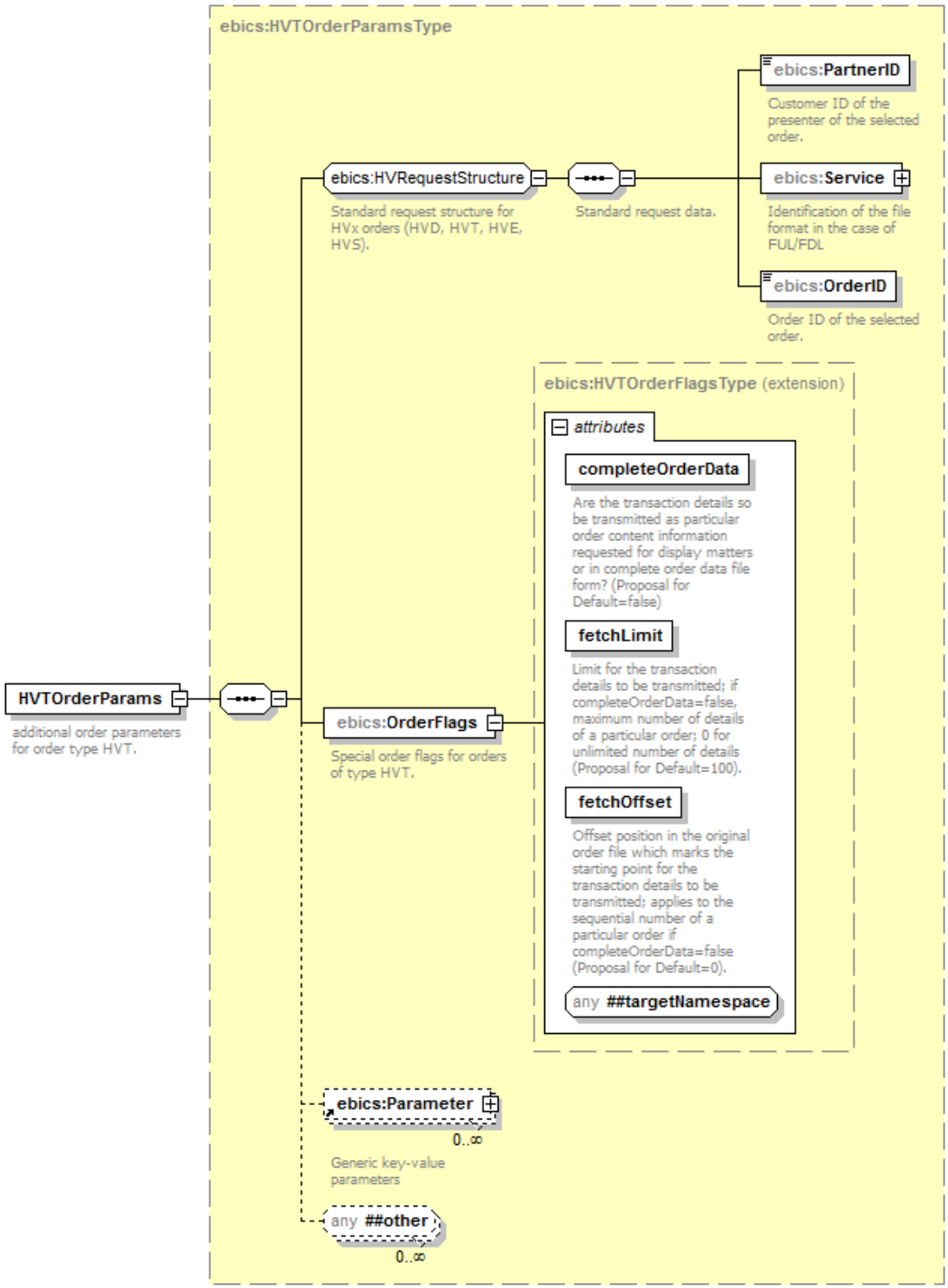


Diagram 77: HVTOrderParams

8.3.3.1.2 Meaning of the XML elements/attributes

XML element/ attribute	Data type	#	Meaning	Example
HVTOrderParams	ebics:HVTOrderParams» Type (complex)	1	Order parameters for order type HVT	- (complex)
PartnerID	ebics:PartnerIDType (→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})	1	Customer ID of the initiating party	“CUSTM001”
Service	ebics:RestrictedServiceType for this structure refer to chapter 8.3.6	1	Kind of business transaction, identified by the service structure	
OrderID	ebics:OrderIDType (→token, fixLength=4)	1	Order number of the order submitted for EDS	“OR01”
OrderFlags	ebics:HVTOrderFlags» Type (complex)	1	Specific “switch“ for HVT orders	- (complex)
OrderFlags» @complete» OrderData	boolean	1	Should the transaction details be transmitted as individual order detailed information (@completeOrderData= "false") or as a complete order file (@completeOrderData= "true")? (Proposal for default="false")	“false“
OrderFlags» @fetchLimit	nonNegativeInteger	1	Maximum number of order details to be transmitted if @completeOrderData= "false", "0“ for unlimited number of details (Proposal for default=“100“)	10
OrderFlags» @fetchOffset	nonNegativeInteger	1	Offset position in the orginal order file which marks the starting point for the transaction details to be transmitted; applies to the sequential number of a particular order if completeOrderData=false. (Proposal for default=“0“)	20

`Parameter`	Reference to global element (complex)	0..∞	Structure for generic key value parameters with optional type specification	- (complex)

8.3.3.1.3 Example XML (abridged)

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
 xmlns="urn:org:ebics:H005"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_request_H005.xsd"
 Version="H005" Revision="1">
  <header authenticate="true">
    <static>
      <!-- […] -->
      <OrderDetails>
        <AdminOrderType>HVT</AdminOrderType>
                <HVTOrderParams>
          <PartnerID>PARTNER1</PartnerID>
          <Service>
              <ServiceName>SCT</ServiceName>
              <MsgName>pain.001</MsgName>
          </Service>
          <OrderID>OR01</OrderID>
          <OrderFlags completeOrderData="false" fetchLimit="50" fetchOffset="0"/>
        </HVTOrderParams>
      </OrderDetails>
      <!-- […] -->
    </static>
    <!-- […] -->
  </header>
  <!-- […] -->
</ebicsRequest>
```

8.3.3.2 HVT response

Depending on the selection of the attribute `completeOrderData` at the element `OrderFlags` the HVT response contains two different formats for the order specified in the HVT request.

If the flag `completeOrderData=true` is set, the customer system requests the download of order data in the original format. This download is a standard download without any additional embedding of order data into an XML document, i.e. the order data are transmitted to the customer system after having been compressed, encrypted and, if required, segmented.

If the flag `completeOrderData=false` is set, the customer system requests the download of order details in the edited XML format. This comprises an XML document with the root element `HVTResponseOrderData` that is transmitted to the customer system after having been compressed, encrypted and, if required, segmented. In this case, the response stores the total number of order details of the original order file in the element `NumOrderInfos`.

Characteristics of the (decoded & decrypted & decompressed) `OrderData` (order data) for HVT: `HVTResponseOrderData`

If a subscriber executes HVT, although the bank does not support HVT for the order on hand, the transaction has to be cancelled and the return code EBICS_UNSUPPORTED_REQUEST_FOR_ORDER_INSTANCE is to be issued.

8.3.3.2.1 XML schema (graphical representation)

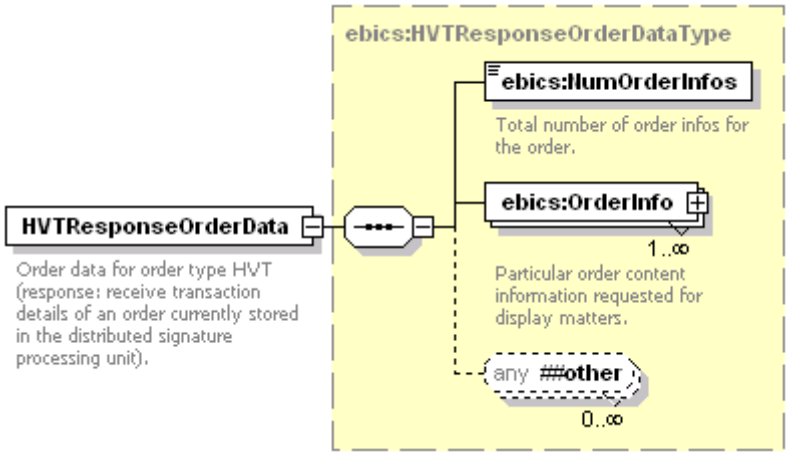


Diagram 78: HVTResponseOrderData

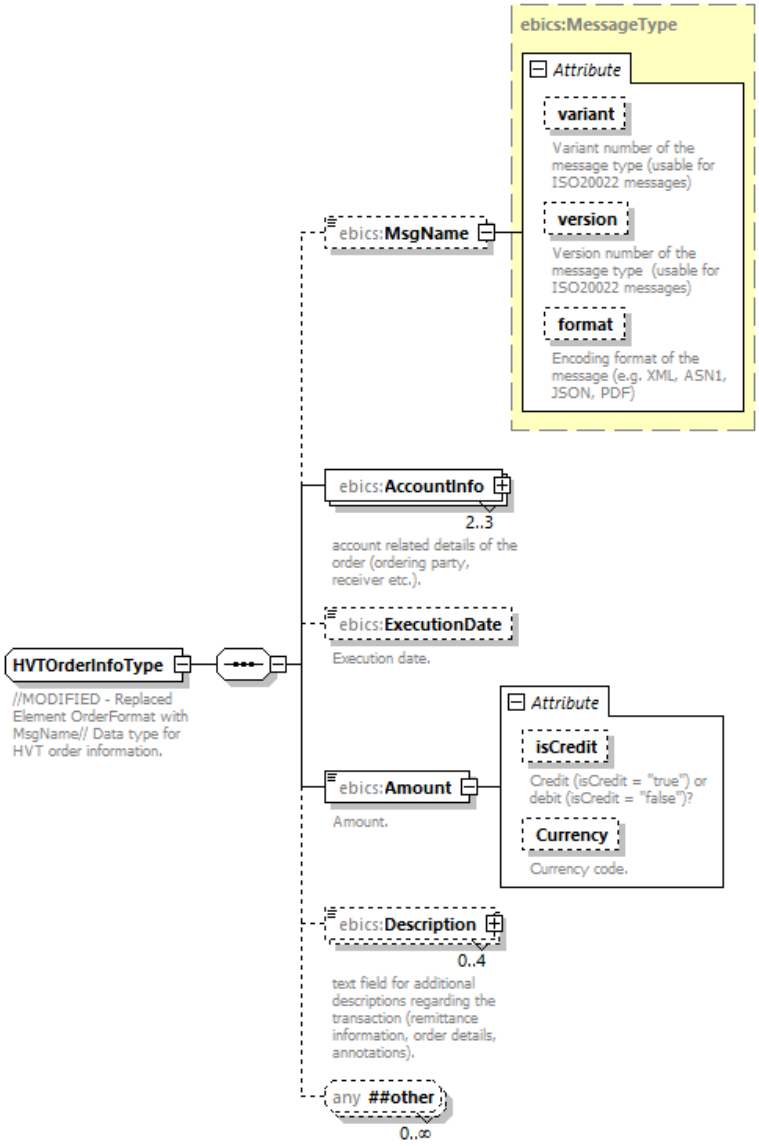


Diagram 79: HVTOrderInfoType (to OrderInfo)

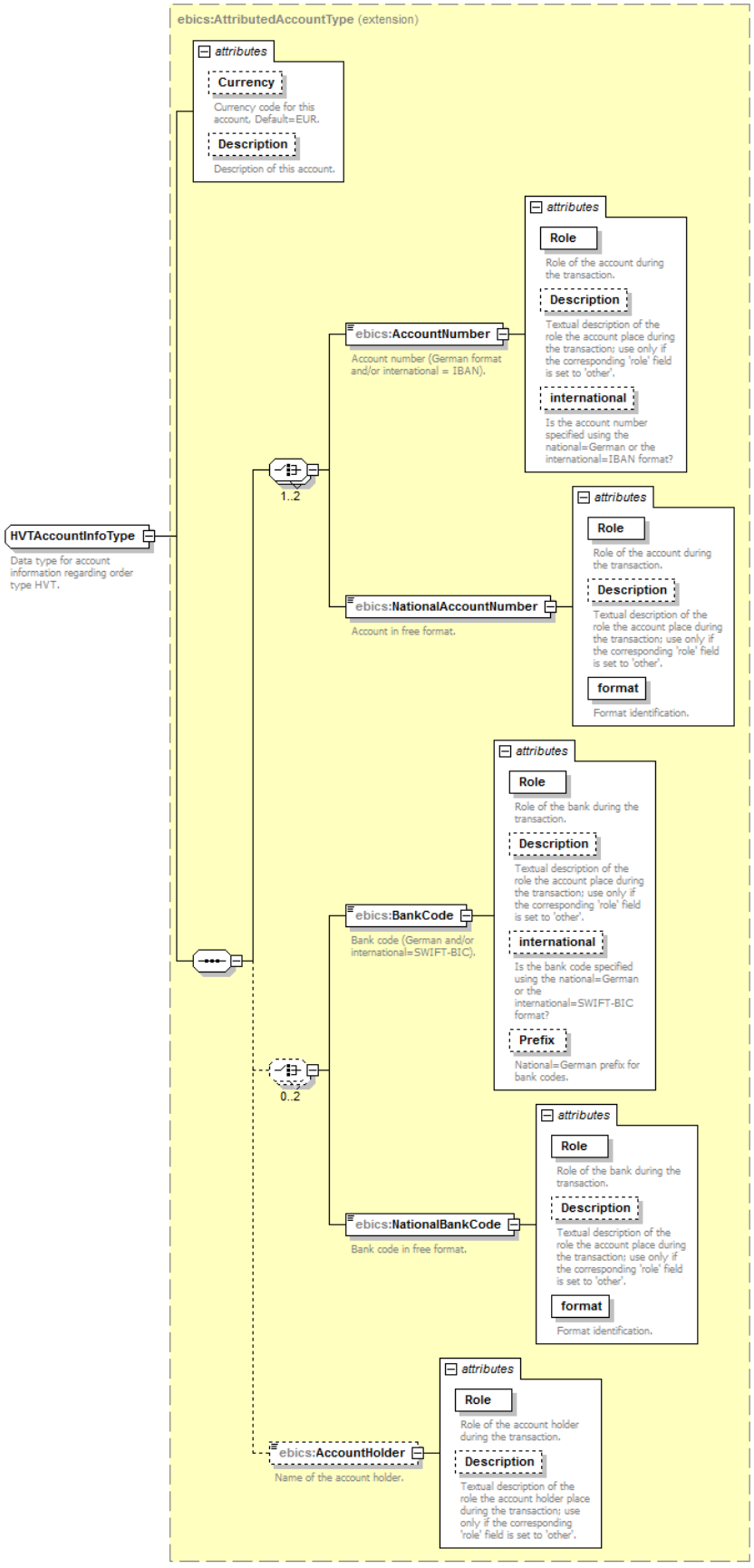


Diagram 80: HVTAccountInfoType (to AccountInfo)

8.3.3.2.2 Meaning of the XML elements/attributes

XML element/ attribute	Data type	#	Meaning	Example
NumOrderInfos	ebics:NumOrderInfosType	1	Total number of particular orders in the original file	42
OrderInfo	ebics:HVTOrderInfo (complex)	1..∞	Individual order information	- (complex)
MsgName	ebics: MessageNameStringType (simple) restriction base: minLength value="1" maxLength value="10" pattern = [a-z\.0-9]	0..1	message names starting with a BA code (ISO) or MT (FIN) or string to be evaluated In the HVT response MsgName is an optional informaton	"pain.001", "mt103" Message names (issued by markets, specified in "scope") are also allowed
MsgName@version	ebics:NumString (simple) restriction base: minLength value="2" maxLength value="2" pattern = [0-9]	0..1	Used ISO version of message, ignored if no ISO message name	"03"
MsgName@variant	ebics:NumString (simple) restriction base: minLength value="3" maxLength value="3" pattern = [0-9]	0..1	Evaluated together with <MsgName>, ignored if no ISO message name	"001"
MsgName@format	ebics:CodeString (simple) restriction base: minLength value="1" maxLength value="4" pattern = [A-Z0-9]	0..1	Evaluated together with <MsgName>, admissible for each kind of message name, but only to be used if it is not the standard format for the used message standard (especially non-XML for ISO 20022).	„XML", „ASN1", „JSON", „PDF"
AccountInfo	ebics:HVTAccount» InfoType (complex)	2..3	Account-related detailed information on the individual order (order party, recipient, opt. initiating party)	- (complex)
AccountInfo» @Currency	ebics:CurrencyBase» Type (→token, length=3, pattern="[A-Z]{3}")	0..1	Currency code of the account in accordance with ISO 4217; default = "EUR"	"EUR"
AccountInfo» @Description	ebics:Account» DescriptionType (→normalizedString	0..1	Textual description of the account	"Savings"

	)			
ExecutionDate	date	0..1	Implementation date of the individual order in accordance with ISO 8601	2005-01-31
Amount	ebics:AmountValue» Type (→decimal, totalDigits=24, fractionDigits=4)	1	Amount of the individual order	1234.567
Amount» @Currency	ebics:CurrencyBase» Type (→token, length=3, pattern="[A-Z]{3}")	0..1	Currency code of the individual order amount in accordance with ISO 4217	“EUR”
Amount» @isCredit	boolean	0..1	Flag for differentiation between credit note (isCredit="true") and debit note (isCredit="false")	“false“
Description	string	0..4	Text fields for further description of the order transaction (purpose, order details, comment)	“Account no. 2345“
Description» @Type	token: "Purpose", "Details", "Comment")	1	Type of description: „Purpose“=reason for payment, „Details“=order details, „Comment“=comment	“Purpose“
-	-	1..2	Information on the account number: AccountNumber and/or NationalAccountNumber	
AccountNumber	ebics:AccountNumber» Type (→token, maxLength=40, pattern="\d{3,10}|([A-Z]{2}\d{2}[A-Za-z0-9]{3,30}")	1	Account number, either in national (= German) or international format (IBAN)	„12345678“
AccountNumber» @Role	ebics:AccountNumber» RoleType (→token: "Originator", "Recipient", "Charges", "Other")	1	Role of the account within the payment transaction: “Originator”=account of the ordering party, “Recipient”=account of the recipient, “Charges”=account for charges, “Other”= other role (see AccountNumber» @Description)	“Originator“
AccountNumber» @Description	normalizedString	0..1	Textual description of the role of the account within	“Nostro“

			the payment transaction if AccountNumber@Role= "Other" is selected.	
AccountNumber» @international	boolean	0..1	Is the account number specified in national = German (AccountNumber» @international="false") or in international = IBAN format (AccountNumber» @international="true")? Default="false"	"false"
NationalAccountNumber		1	Account number in free format (for national account numbers that correspond to neither German nor international standards)	„123456789012 3456"
NationalAccountNumber» @Role	ebics:AccountNumber» RoleType (→token: "Originator", "Recipient", "Charges", "Other")	1	Role of the account within the transaction: "Originator"=account of the ordering party, "Recipient"=account of the recipient, "Charges"=account for charges, "Other"= other role (see AccountNumber» @Description)	„Originator"
NationalAccountNumber» @Description	normalizedString	0..1	Textual description of the account within the transaction if AccountNumber@Role= "Other" is selected	„Nostro"
National» AccountNumber» @format	token	1	Description of the account number's format	„other"
-	-	0..2	Information on the bank code: BankCode and/or NationalBankCode	-
BankCode	ebics:BankCodeType (→token, maxLength=11, pattern="\d{8}|([A-Z]{6}[A-Z0-9]{2}([A-Z0-9]{3})?)")	0..1	Bank code, either in national (= German) or international format (SWIFT)	„50010060"
BankCode@Role	ebics:BankCodeRole» Type (→token: "Originator",	1	Role of the financial institution within the payment transaction: „Originator"=ordering bank, „Recipient"=receiving bank,	"Originator"

	"Recipient", "Correspondent", "Other")		„Correspondent"= correspondent bank, „Other"=other role (see BankCode@Description)	
BankCode» @Description	normalizedString	0..1	Textual description of the role of the financial institution within the payment transaction, if BankCode@Role="Other" is selected	"Clearing"
BankCode» @international	boolean	0..1	Is the bank code specified in national = German (BankCode» @international= "false") or international = SWIFT format (BankCode» @international="true")? Default="false"	"false"
BankCode@Prefix	token, maxLength=2	0..1	National prefix for bank codes	"DE"
NationalBank» Code	ebics:National» BankCodeType (→token, maxLength=30)	1	Bank code in free format (neither German format nor SWIFT-BIC)	„123456789012"
NationalBank» Code@Role	ebics:BankCodeRole» Type (→token: "Originator", "Recipient", "Correspondent", "Other")	1	Role of the financial institution within the transaction: „Originator"=ordering bank, „Recipient"=receiving bank, „Correspondent"=correspondent bank, „Other"=other role (see BankCode@Description)	„Originator"
BankCode» @Description	normalizedString	0..1	Textual description of the role the financial institution plays within the transaction, if BankCode@Role="Other" is chosen	„Clearing"
NationalBank» Code@format	token	1	Format type	"other"
AccountHolder	ebics:AccountHolder» Type (→normalizedString)	0..1	Name of the account holder	"John Doe"
AccountHolder» @Role	ebics:AccountHolder» RoleType (→token: "Originator", "Recipient", "Presenter",	0..1	Role of the account holder within the payment transaction: „Originator"=ordering party, „Recipient"=recipient,	"Originator"

	"Other")		„Presenter“=submitting party of the order, „Other“=other role (see AccountHolder» @Description)	
AccountHolder» @Description	normalizedString	0..1	Textual description of the role of the account holder within the payment transaction if AccountHolder@Role= "Other" is selected.	“Trustee“

8.3.3.2.3 Example XML

```
<?xml version="1.0" encoding="UTF-8"?>
<HVTResponseOrderData
 xmlns="urn:org:ebics:H005"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_orders_H005.xsd">
 <NumOrderInfos>42</NumOrderInfos>
 <OrderInfo>
   <AccountInfo Currency="EUR">
     <AccountNumber Role="Originator" international="false">1234567890</AccountNumber>
     <BankCode Role="Originator" international="false" Prefix="DE">50010060</BankCode>
     <AccountHolder Role="Originator">Ophelia Originator</AccountHolder>
   </AccountInfo>
   <AccountInfo Currency="EUR">
     <AccountNumber Role="Recipient" international="false">1122334455</AccountNumber>
     <BankCode Role="Recipient" international="false">50070010</BankCode>
     <AccountHolder Role="Recipient">Ray Recipient</AccountHolder>
   </AccountInfo>
   <ExecutionDate>2005-01-31</ExecutionDate>
   <Amount isCredit="true" Currency="EUR">500.00</Amount>
   <Description Type="Purpose">Test transer</Description>
 </OrderInfo>
</HVTResponseOrderData>
```

8.3.4 HVE (add electronic signature)

With HVE, the subscriber adds a further bank-technical signature for authorisation to an order from EDS processing.

The bank system has to verify whether the subscriber possesses a bank-technical authorisation of signature (not signature class T) for the referenced order. If the authorisation is missing, the transaction has to be cancelled and the existing return code EBICS_AUTHORISATION_ORDER_IDENTIFIER_FAILED is issued.

Before HVE is executed, the bank system verifies whether the order is currently located in the EDS processing system and terminates the transaction in case of an error returning the business related error code EBICS_ORDERID_UNKNOWN.

HVE is an administrative order type of type “upload”. Only the ES is transmitted via the hash value of the order from EDS processing (no order data, no ES for the administrative order type HVE itself) whereas only the hash value of the EDS processing is signed.

8.3.4.1 HVE request

With the HVE request, the subscriber specifies the order to which they want to add a bank-technical signature, and supplies this signature in the same request in the XML body element `ebicsRequest/body/DataTransfer/SignatureData` in compressed, encrypted and base64-coded form. An HVE request does not contain any order data, i.e. the XML body element `ebicsRequest/body/DataTransfer/OrderData` remains unfilled.
Since there is no data digest of order data the mandatory element `ebicsRequest/body/DataTransfer/DataDigest` hast to be delivered without content (empty tag) in the HVE request.

In order to provide the bank-technical signature, the subscriber needs either the hash value of the original order data (e.g. retrievable via HVD or HVZ) or the order data itself (e.g. via HVT with `completeOrderData="true"`).

Characteristics of `OrderParams` (order parameters) for HVE: `HVEOrderParams`

8.3.4.1.1 XML schema (graphical representation)

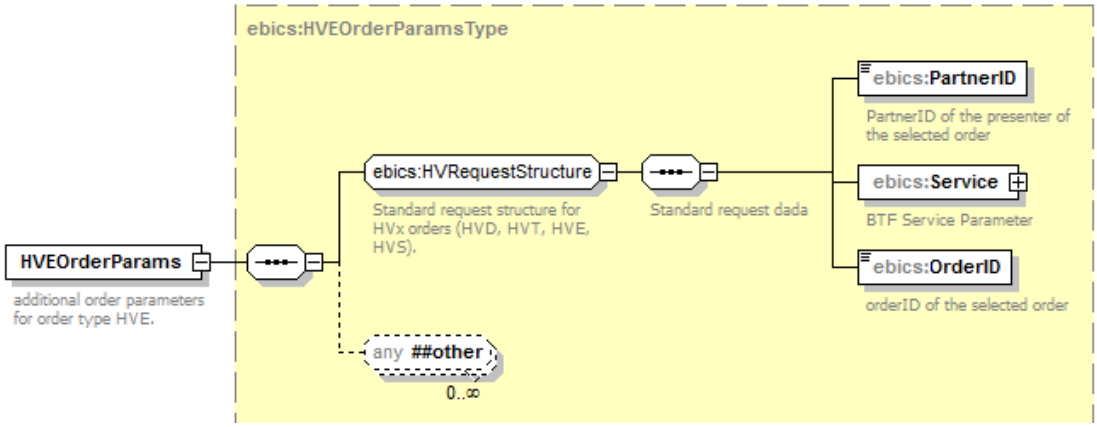


Diagram 81: HVEOrderParams

8.3.4.1.2 Meaning of the XML elements/attributes

XML element/ attribute	Data type	#	Meaning	Example

HVEOrderParams	ebics:HVEOrderParamsType (complex)	1	Order parameters for order type HVE	- (complex)
PartnerID	ebics:PartnerIDType (→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})	1	Customer ID of the initiating party	"PARTNER1"
Service	ebics:RestrictedServiceType for this structure refer to chapter 8.3.6	1	Kind of business transaction, identified by the service structure	
OrderID	ebics:OrderIDType (→token, fixLength=4)	1	Order number of the order in EDS processing	"OR01"

8.3.4.1.3 Example XML (abridged)

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
 xmlns="urn:org:ebics:H005"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_request_H005.xsd"
 Version="H005" Revision="1">
  <header authenticate="true">
    <static>
      <!-- […] -->
      <OrderDetails>
        <AdminOrderType>HVE</AdminOrderType>
        <OrderID>HO04</OrderID>
              <HVEOrderParams>
          <PartnerID>CUSTM001</PartnerID>
                    <Service>
              <ServiceName>SCT</ServiceName>
              <MsgName>pain.001</MsgName>
          </Service>
          <OrderID>OR01</OrderID>
        </HVEOrderParams>
      </OrderDetails>
      <!-- […] -->
    </static>
    <!-- […] -->
  </header>
  <!-- […] -->
</ebicsRequest>
```

8.3.4.2 HVE response

The HVE response does not contain any EDS-specific data.

8.3.5 HVS (Cancellation of orders in the EDS)

The subscriber uses HVS to permanently cancel an existing order from EDS processing.

Before HVS is executed, the bank system verifies whether the order is currently located in the EDS processing system and terminates the transaction in case of an error returning the business related error code EBICS_ORDERID_UNKNOWN.

HVS is an administrative order type of type “upload”. For cancellation authorisation, the ES is transmitted via the hash value of the order that is to be cancelled (no order data, no ES for the administrative order type HVS itself).

8.3.5.1 HVS request

The subscriber uses the HVS request to specify the order that is to be cancelled and delivers the bank-technical signature that is necessary for the cancellation via the hash value of the order data.

The bank system has to verify whether the subscriber possesses a bank-technical authorisation of signature (not signature class T) for the referenced order. If the authorisation is missing, the transaction will be cancelled and the existing return code EBICS_AUTHORISATION_ORDER_IDENTIFIER_FAILED is issued.

The signature is transported in compressed, encrypted and base64-coded form in the XML body element `ebicsRequest/body/DataTransfer/SignatureData` . The order cancellation is permanent, and always requires one single authorised signature of class “E”, “A” or “B”.

In order to provide the bank-technical signature, the subscriber needs either the hash value of the original order data (e.g. retrievable via HVD or HVZ) or the order data itself (e.g. via HVT with `completeOrderData="true"`).

An HVS request does not contain order data (similar to HVE request), i.e. the XML body element `ebicsRequest/body/DataTransfer/OrderData` remains unfilled.
Therefore there is no order data digest - the mandatory element `ebicsRequest/body/DataTransfer/DataDigest` must therefore be delivered without content (empty tag) in the HVS request.

Characteristics of `OrderParams` for HVS: `HVSOrderParams`

8.3.5.1.1 XML schema (graphic representation)

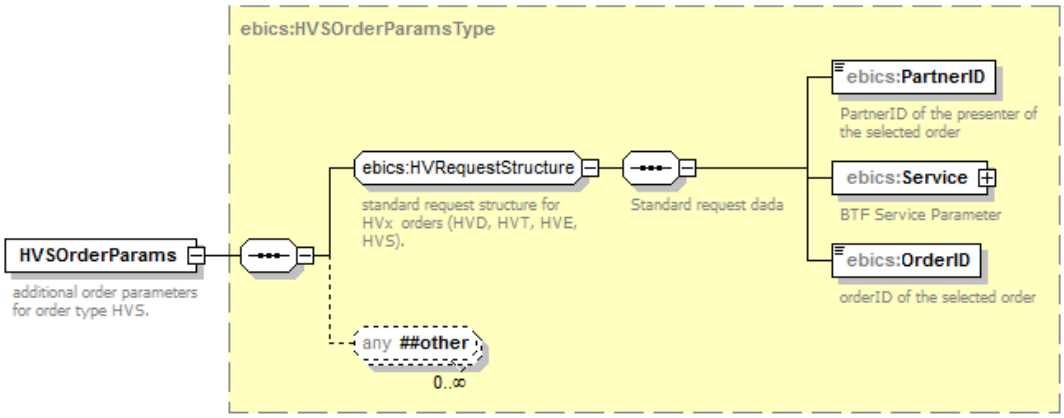


Diagram 82: HVSOrderParams

8.3.5.1.2 Meaning of the XML elements/attributes

XML element/ attribute	Data type	#	Meaning	Example
HVSOrderParams	ebics:HVSOrderParamsType (complex)	1	Order parameters for order type HVS	- (complex)
PartnerID	ebics:PartnerIDType (→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})	1	Customer ID of the initiating party.	“CUSTM001”
Service	ebics:RestrictedServiceType for this structure refer to chapter 8.3.6	1	Kind of business transaction, identified by the service structure	
OrderID	ebics:OrderIDType (→token, fixLength=4)	1	Order number of the order that is to be cancelled in EDS processing	“OR01”

8.3.5.1.3 Example XML (abridged)

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
 xmlns="urn:org:ebics:H005"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_request_H005.xsd"
 Version="H005" Revision="1">
  <header authenticate="true">
    <static>
      <!-- […] -->
      <OrderDetails>
        <AdminOrderType>HVS</AdminOrderType>
        <OrderID>HO05</OrderID>
                <HVSOrderParams>
          <PartnerID>CUST001</PartnerID>
```

```
          <Service>
            <ServiceName>SCT</ServiceName>
            <MsgName>pain.001</MsgName>
          </Service>
          <OrderID>OR01</OrderID>
        </HVSOrderParams>
      </OrderDetails>
      <!-- […] -->
    </static>
    <!-- […] -->
  </header>
  <!-- […] -->
</ebicsRequest>
```

8.3.5.2 HVS response

The HVS response does not contain any EDS-specific data.

8.3.6 Used Service Structures (restricted and not restricted)

The service structure (of type `RestrictedServiceType`) with optional and mandatory elements is used for nearly all upload and download requests where BTF information is needed. Only for the HVU and HVZ request, however, ALL elements in the structure have to be optional (type of structure is `ServiceType`). Hence the user can request a list of orders which map with a specific filter of BTF elements. If the structure is not specified, all orders are retrieved for which the subscriber is authorised as a signatory.

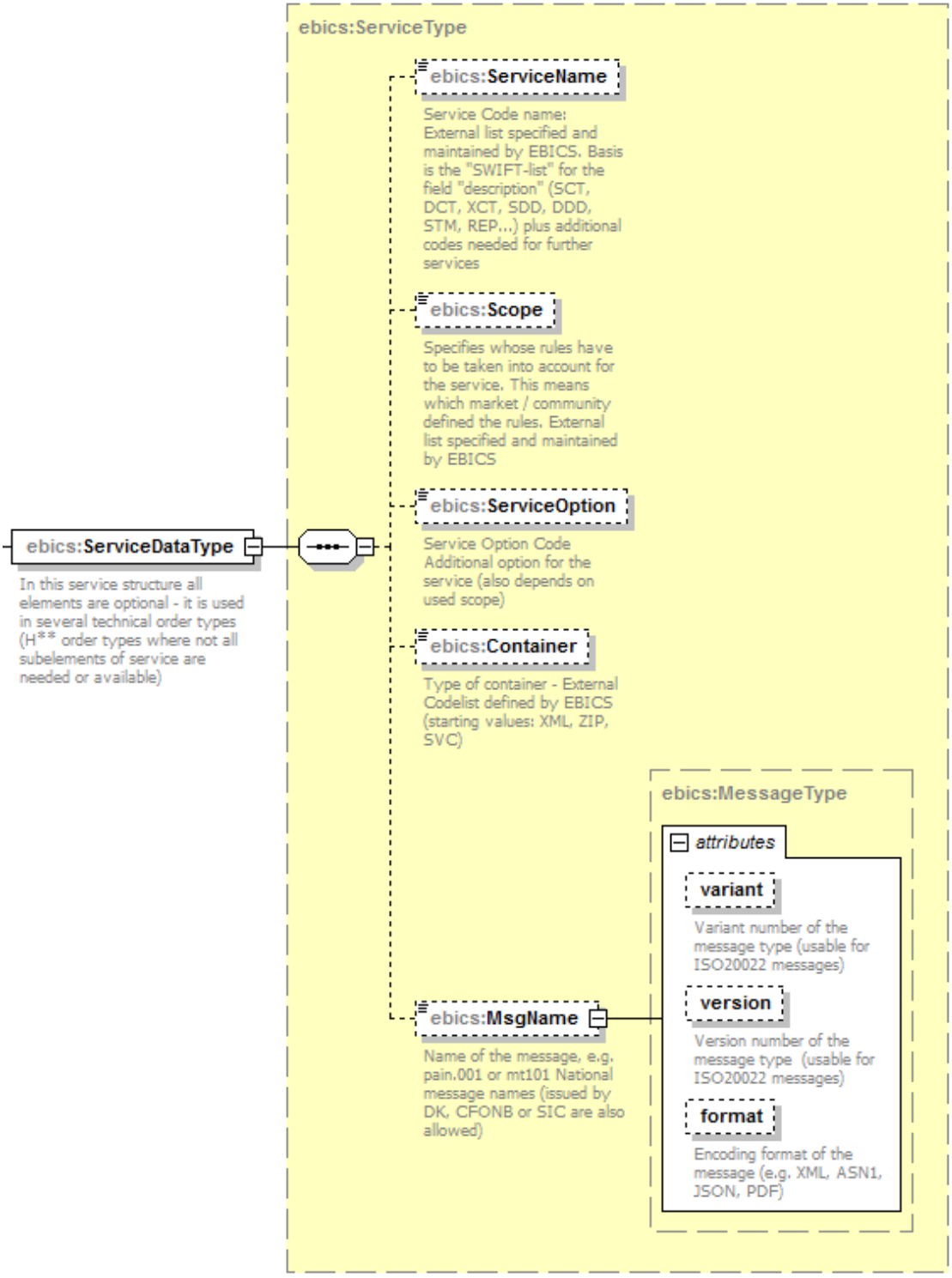


Diagram 83: non-restricted BTF service structure only for HVU and HVZ request

For all other requests and all responses including description about the business transaction format (BTF) the standard restricted service structure is used:

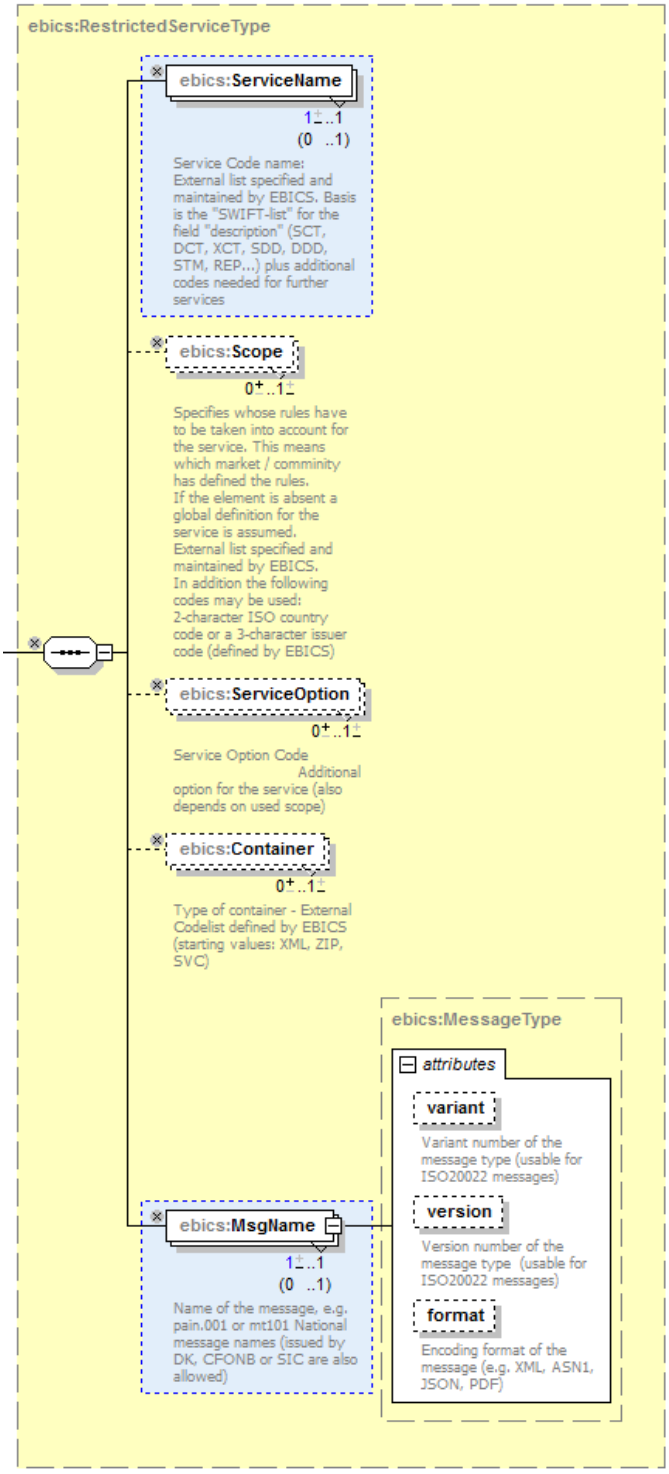


Diagram 84: "standard" BTF service structure for all other cases

For the meaning of the element/attributes please refer to chapter 5.5.1.1.3.

9 “Other” administrative EBICS order types

The following sections contain descriptions of the following order types:

- HAA (download retrievable services / BTF)
- HPD (download bank parameter)
- HKD (download customer’s customer and subscriber information)
- HTD (download subscriber’s customer and subscriber information)

HEV (download supported EBICS versions)Information about the support on the part of the bank (mandatory, optional) see chapter 13.

9.1 HAA (download retrievable business transaction formats BTF)

With HAA, the subscriber may retrieve all kinds of business transaction formats for which updated customer data are ready for download in the bank system.

HAA is an administrative order type of type “download”.

9.1.1 HAA request

The HAA request does not contain specific data that goes beyond that named in the general transaction description (see Chapter 5.6.1.1).

9.1.2 HAA response

Characteristics of the (decoded & decrypted & decompressed) `OrderData` (order data) for HAA: `HAAResponseOrderData`

9.1.2.1.1 XML schema (graphic representation)

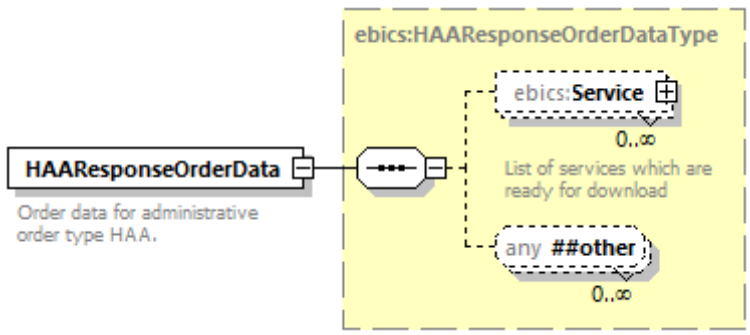


Diagram 85: HAAResponseOrderData

9.1.2.1.2 Meaning of the XML elements/attributes

XML element/ attribute	Data type	#	Meaning	Example
`HAAResponse» OrderData`	`ebics:HAAResponse» OrderDataType` (complex)	1	Order data for administrative order type HAA	- (complex)
`Service`	`ebics:RestrictedServiceType` for this structure refer to chapter 8.3.6	0..∞	Kind of business transaction, identified by the service structure	

9.2 HPD (download bank parameters)

With HPD, the subscriber can receive information relating to the financial institution's specific access (`AccessParams`) and protocol parameters (`ProtocolParams`).

The access parameters include:

- `URL`: URL or IP address for electronic access to the financial institution. The optional attribute `valid_from` specifies the commencement of validity (timestamp) of the specification
- `Institute`: Designation of the financial institution
- `HostID` (optional): EBICS host ID of the bank system.

In the case of the protocol parameters, the following information is transmitted:

- `Version`: Permitted versions (listed in each case) for EBICS protocol (`Protocol`), identification and authentication (`Authentication`), encryption (`Encryption`) and signature (`Signature`)
- `Recovery` (optional): Support of transaction recovery of (`@supported`)
- `PreValidation` (optional): Support of preliminary verification (`@supported`). If this parameter is set, the financial institution merely ensures that the subscriber can transmit data to the financial institution within the framework of preliminary verification. However, the financial institution is not obliged to comprehensively verify this data.
- `ClientDataDownload` (optional): Support of administrative order types HKD (download customer data) and HTD (download subscriber data) (`@supported`). See Chapter 9.3 (HKD) and 9.4 (HTD)
- `DownloadableOrderData` (optional): Support of administrative order type HAA (download retrievable services / BTF) (`@supported`). See Chapter 9.1 for details.

The following standard procedure is defined for all optional elements of the protocol parameters – insofar as not explicitly stated otherwise:

- If the parameter is missing, the subscriber MUST evaluate this as meaning that the corresponding functionality is not supported, i.e. the result corresponds to *Parameter*`@supported="false"`
- If the parameter is specified, but the attribute is missing, the subscriber MUST evaluate this as support of the corresponding functionality, i.e. the result corresponds to *Parameter*`@supported="true"`.

This specification simplifies the inter-operability of customer product and bank system: On the one hand, it is ensured that a financial institution that does not support a function does not also have to explicitly state that it is "not supported" in the bank parameters. On the other hand, it is assumed that if a functionality is named then it is also supported, which means that in this case the `@supported` flag can be dispensed with.

HPD is an administrative order type of type "download".

9.2.1 HPD request

The HPD request does not contain specific data that goes beyond that named in the general transaction description.

9.2.2 HPD response

The HPD response contains the bank parameters, divided into access parameters (`AccessParams`) and protocol parameters (`ProtocolParams`).

Characteristics of the (decoded & decrypted & decompressed) `OrderData` (order data) for HPD: `HPDResponseOrderData`

9.2.2.1.1 XML schema (graphic representation)

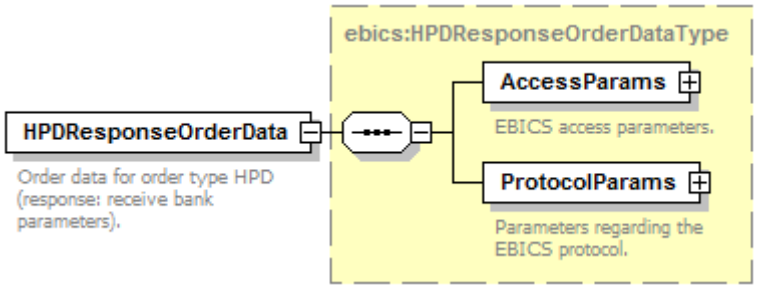


Diagram 86: HPDResponseOrderData

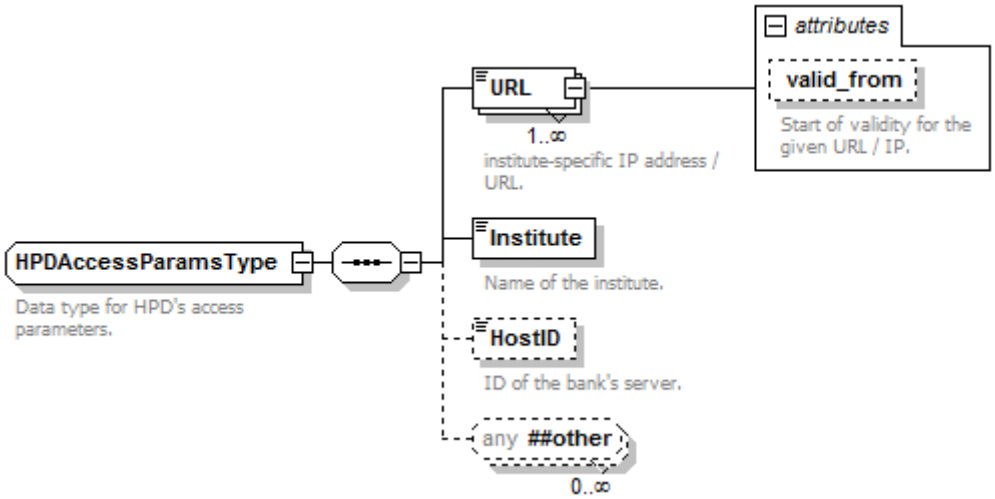


Diagram 87: HPDAccessParamsType (to AccessParams)

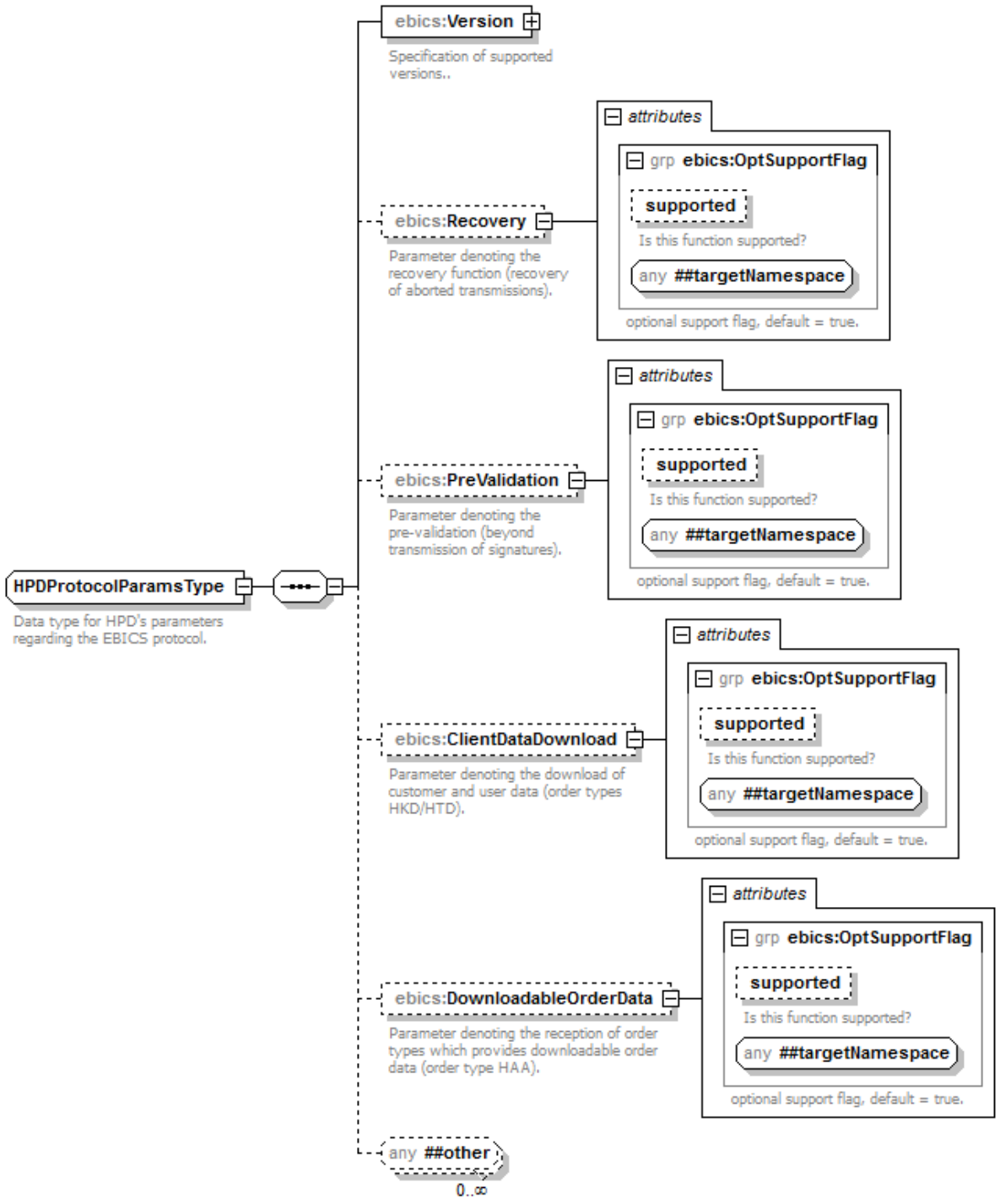


Diagram 88: HPDProtocolParamsType (to ProtocolParams)

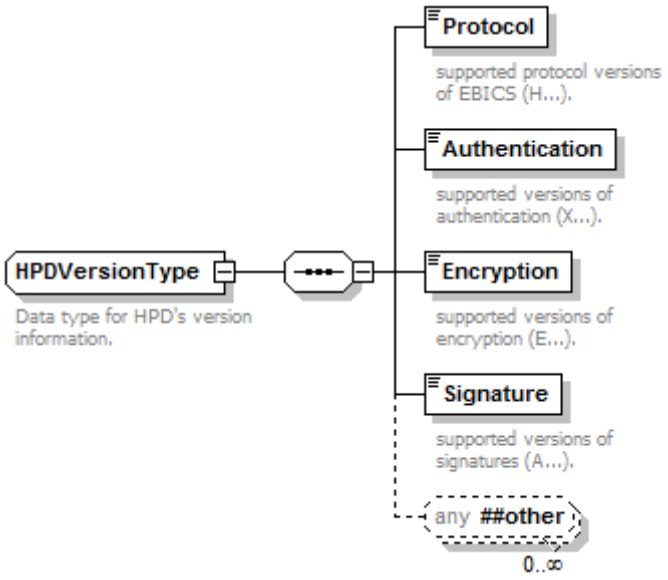


Diagram 89: HPDVersionType (to Version)

9.2.2.1.2 Meaning of the XML elements/attributes

XML element/ attribute	Data type	#	Meaning	Example
`HPDResponse» OrderData`	`ebics:HPDResponse» OrderDataType` (complex)	1	Order data for administrative order type HPD	- (complex)
`AccessParams`	`ebics:HPDAccessParams» Type` (complex)	1	Access parameters	- (complex)
`ProtocolParams`	`ebics:HPDProtocol» ParamsType` (complex)	1	Protocol parameters	- (complex)
`URL`	`anyURI`	1..∞	Institute-specific IP address / URL	“www.the-bank.de”
`URL@valid_from`	`ebics:TimestampType (→dateTime)`	0..1	Commencement of validity for the specified URL/IP; if not specified, the URL/IP is valid with immediate effect	“2005-02-28T» 15:30:45.123Z“
`Institute`	`normalizedString, maxLength=80`	1	Financial institution designation	“The Bank“
`HostID`	`ebics:HostIDType (→token, maxLength=35)`	0..1	EBICS bank system ID	“EBIXHOST“
`Version`	`ebics:HPDVersionType` (complex)	1	Specification of supported versions	- (complex)
`Protocol`	`list<ebics:Protocol» VersionType> (→list<token, length=4, pattern="H\d{3}">)`	1	List of supported EBICS protocol versions	“H005“
`Authentication`	`list<ebics:Authentica» tionVersionType>`	1	List of supported identification and	“X002”

	(→list<token, length=4, pattern= "X\d{3}">)		authentication versions	
Encryption	list<ebics:Encryption» VersionType> (→list<token, length=4, pattern="E\d{3}">)	1	List of supported encryption versions	“E002”
Signature	list<ebics:Signature» VersionType> (→list<token, length=4, pattern="A\d{3}">)	1	List of supported ES versions	“A005 A006“
Recovery	- (complex)	0..1	Parameters for recovery function (recovery of broken connections); if not specified, the function is not supported.	- (complex)
Recovery» @supported	boolean	0..1	Is recovery supported? (Default=true)	“true”
PreValidation	- (complex)	0..1	Parameters for preliminary verification; if not specified, the function is not supported	- (complex)
PreValidation» @supported	boolean	0..1	Is preliminary verification supported? (Default=true)	“true”
ClientData» Download	- (complex)	0..1	Parameters for downloading customer and subscriber data (HKD/HTD); if not specified, the function is not supported	“true”
ClientData» Download» @supported	boolean	0..1	Are administrative order types HKD/HTD supported? (Default=true)	“true”
Downloadable» OrderData	- (complex)	0..1	Parameters for retrieving services / BTF for which order data is available (HAA); if not specified, the function is not supported	- (complex)
Downloadable» OrderData» @supported	boolean	0..1	Is administrative order type HAA supported? (Default=true)	“true”

9.2.2.1.3 Example XML

```
<?xml version="1.0" encoding="UTF-8"?>
<HPDResponseOrderData
 xmlns="urn:org:ebics:H005"
```

```
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_orders_H005.xsd">
 <AccessParams>
   <URL>http://www.the-bank.de</URL>
   <URL valid_from="2005-02-15T15:30:45.123Z">192.168.0.1</URL>
   <Institute>The Bank</Institute>
   <HostID>EBIXHOST</HostID>
 </AccessParams>
 <ProtocolParams>
   <Version>
     <Protocol>H005</Protocol>
     <Authentication>X002</Authentication>
     <Encryption>E002</Encryption>
     <Signature>A005 A006</Signature>
   </Version>
   <Recovery supported="true"/>
   <PreValidation supported="true"/>
   <DownloadableOrderData supported="true"/>
 </ProtocolParams>
</HPDResponseOrderData>
```

9.3 HKD (retrieve customer's customer and subscriber information)

With HKD, the subscriber can retrieve information stored by the bank relating to his company and all associated subscribers (including themselves).

The bank's response contains a list of the accounts of the customer.
An account is only included in the HKD response if at least one of the following conditions is complied with:

1. The customer possesses an agreement on the provision of bank statements for the account.
2. At least one of the customer's subscribers is authorised to sign for the account.

It is not relevant whether the account holder is the same customer the HKD is retrieved for.

HKD is an administrative order type of type "download".

9.3.1 HKD request

The HKD request does not contain specific data that goes beyond that named in the general transaction description.

9.3.2 HKD response

Characteristics of the (decoded & decrypted & decompressed) `OrderData` (order data) for HKD: `HKDResponseOrderData`

9.3.2.1.1 XML schema (graphic representation)

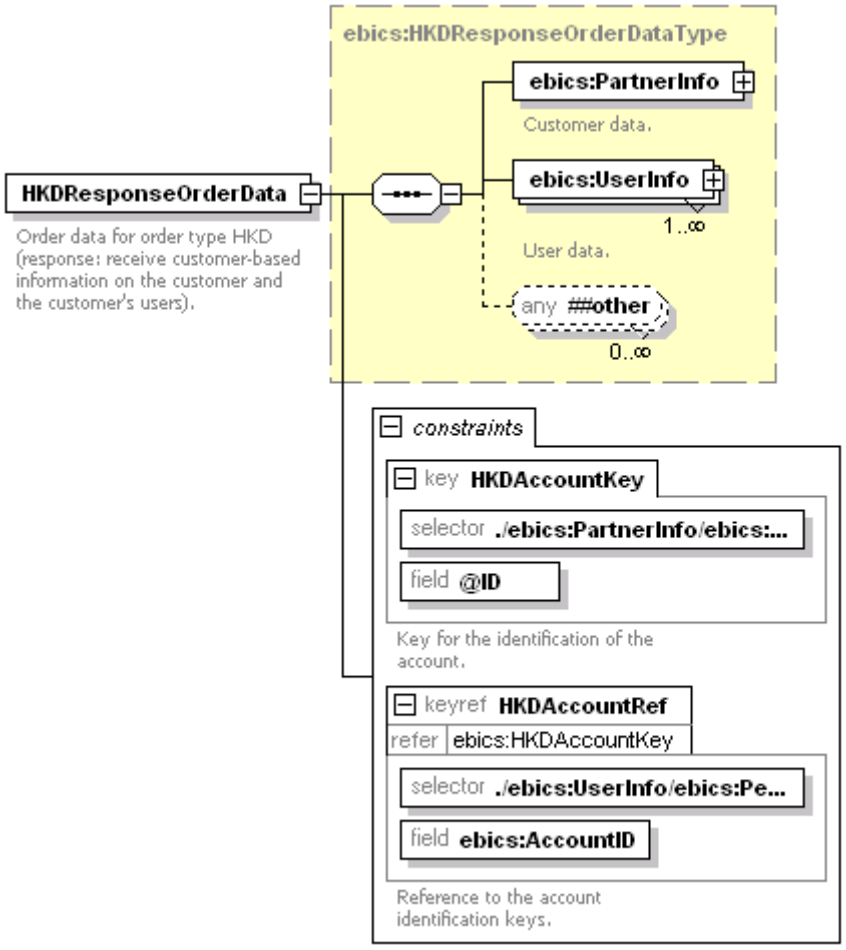


Diagram 90: HKDResponseOrderData

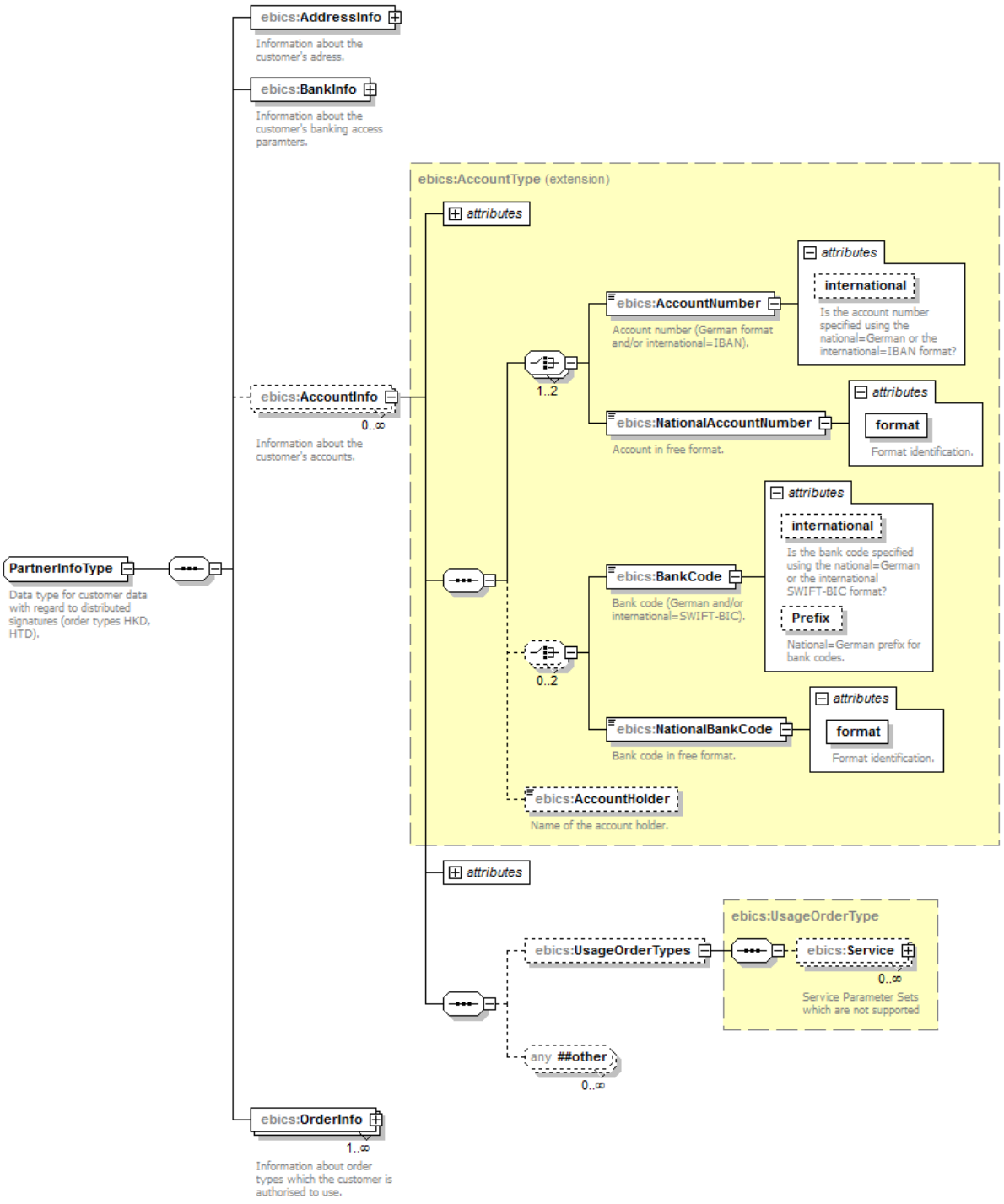


Diagram 91: PartnerInfoType (to PartnerInfo)

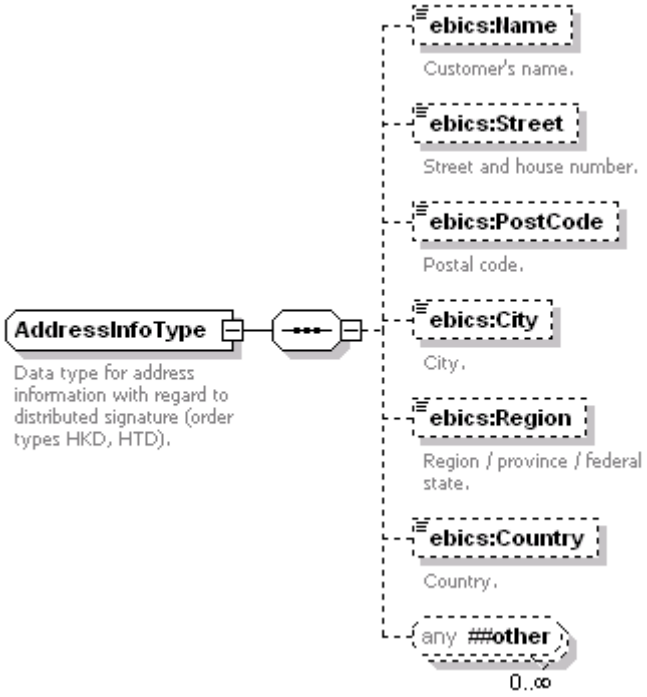


Diagram 92: AddressInfoType (to AddressInfo)

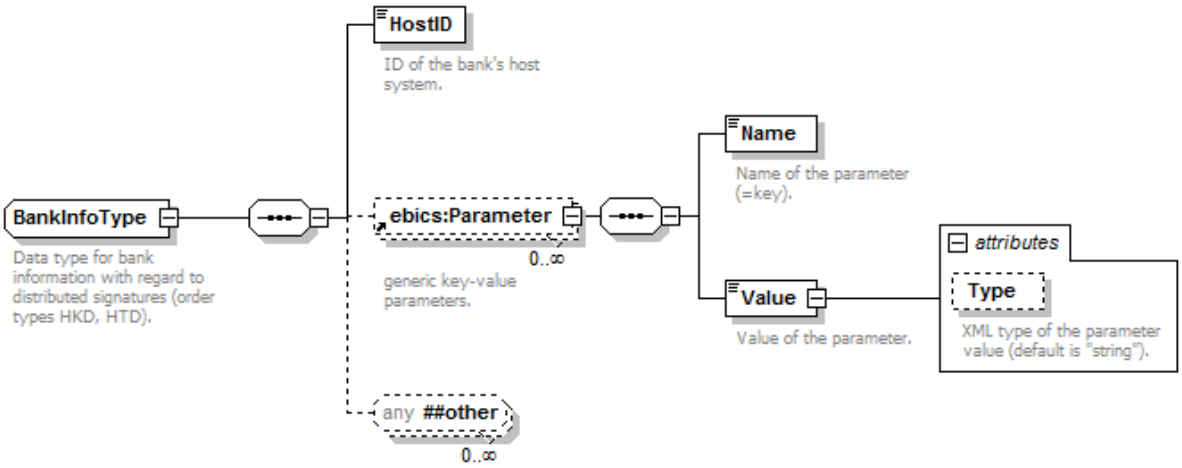


Diagram 93: BankInfoType (to BankInfo)

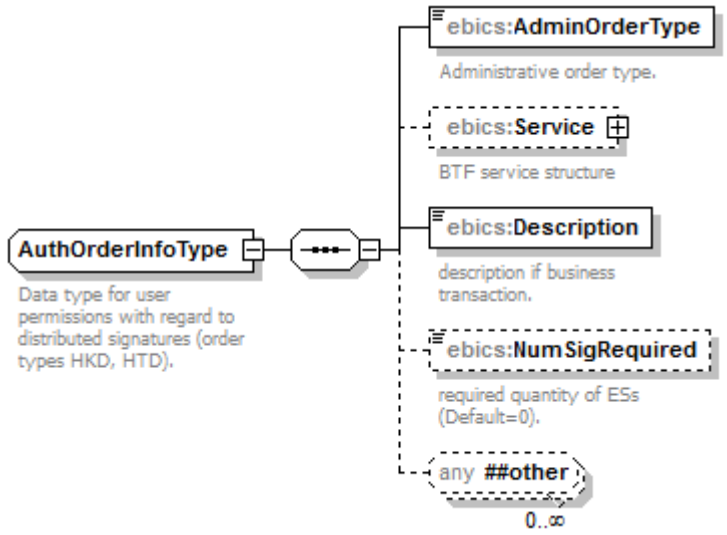


Diagram 94: AuthOrderInfoType (to OrderInfo)

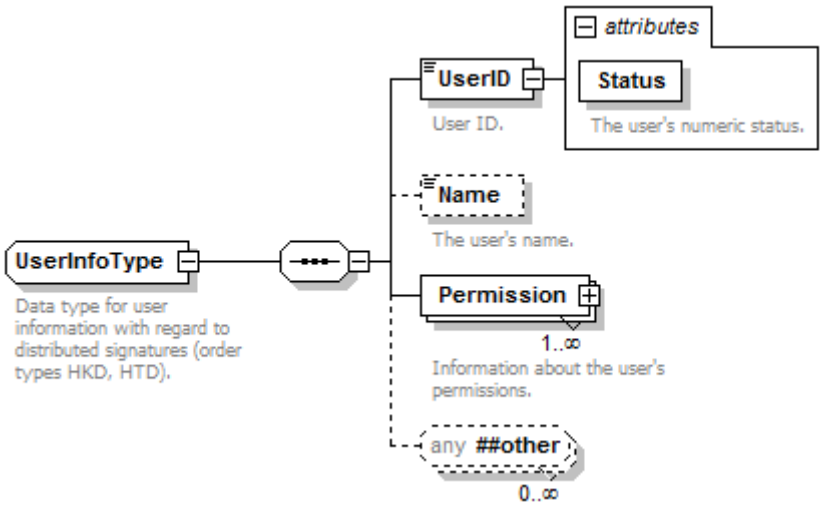


Diagram 95: UserInfoType (to UserInfo)

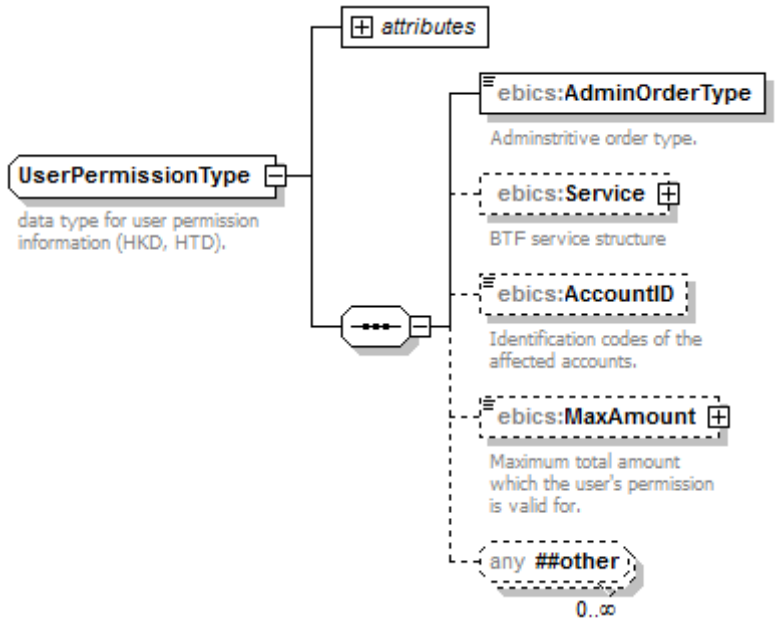


Diagram 96: UserPermissionType (to Permission)

9.3.2.1.2 Meaning of the XML elements/attributes

XML element/ attribute	Data type	#	Meaning	Example
`HKDResponse» OrderData`	`ebics:HKDResponse» OrderDataType` (complex)	1	Order data for administrative order type HKD	- (complex)
`PartnerInfo`	`ebics:PartnerInfoTyp e` (complex)	1	Customer data	- (complex)
`AddressInfo`	`ebics:AddressInfoTyp e`	1	Customer's address information	- (complex)
`Name` (in `AddressInfo`)	`ebics:NameType (→normalizedString)`	0..1	Customer's name	"John Doe"
`Street`	`ebics:NameType (→normalizedString)`	0..1	Customer's street and house number	"Elmstreet 1"
`PostCode`	`token`	0..1	Customer's post code	"12345"
`City`	`ebics:NameType (→normalizedString)`	0..1	Customer's city	"Smallville"
`Region`	`ebics:NameType (→normalizedString)`	0..1	Customer's region / Federal State	"Virginia"
`Country`	`ebics:NameType (→normalizedString)`	0..1	Customer's country	"USA"
`BankInfo`	`ebics:BankInfoType` (complex)	1	Information on customer's financial institution connection	- (complex)
`HostID`	`ebics:HostIDType (→token, maxLength=35`	1	EBICS bank system ID	"EBIXHOST" "EBICS-HOST4711"

				"BANKFRPP"
Parameter	Reference to global element (complex)	0..∞	Structure for generic key value parameters with optional type specification	- (complex)
AccountInfo	ebics:AccountType (complex)	0..∞	Information on customer's accounts. An account is only listed in the HKD response if the customer possesses an agreement on the provision for it, OR if at least one of the customer's subscribers is authorised to sign for the account. The account holder does not have to be the same customer as the one the HKD is retrieved for.	- (complex)
AccountInfo» @Currency	ebics:CurrencyBaseType (→token, length=3)	0..1	Currency code for the account in question, according to ISO 4127; if not specified, "EUR" is assumed	"EUR"
Description	ebics:Account» DescriptionType (→normalizedString)	0..1	Textual description of the account	"Giro account"
AccountInfo@ID	ebics:AccountIDType (→token, maxLength=64)	1	Unambiguous account identification code	"ABCDEFG» abcdefg» 1234567890"
-	-	1..2	Information on the account number: AccountNumber and/or NationalAccountNumber	-
AccountNumber	ebics:AccountNumber» Type (→token, maxLength=40, pattern="\d{3,10}|([A-Z]{2}\d{2}[A-Za-z0-9]{3,30})")	1	Account number (German format or international as IBAN)	„123456789"
AccountNumber» @international	boolean	0..1	Is the account number given in national=German (false, default) or in international=IBAN format (true)?	"false"
National» AccountNumber	ebics:National» AccountNumberType (→token, maxLength=40)	1	Account number in free format (for national account numbers which comply neither to German nor international standards)	„1234567890123456"
National» Account»	token	1	Description of the format of the account number	„other"

Number@format				
-	-	0..2	Information on the bank code: BankCode and/or NationalBankCode	-
BankCode	ebics:BankCodeType (→token, maxLength=11, pattern="\d{8}|([A-Z]{6}[A-Z0-9]{2}([A-Z0-9]{3})?)")	1	Bank sort code (German format or international as SWIFT-BIC)	„50010070“
BankCode» @international	boolean	0..1	Is the bank sort code given in national=German (false, default) or in international=SWIFT-BIC format (true)?	“false“
BankCode» @Prefix	ebics:BankCodePrefix» Type (→token, length=2)	0..1	National bank sort code prefix	“DE“
NationalBank» Code	ebics:National» BankCodeType (→token, maxLength=30)	1	Bank code in free format (neither German format nor SWIFT-BIC)	„123456789012“
NationalBank» Code@format	token	1	Description of the bank code format	“other”
AccountHolder	ebics:AccountHolder» Type (→normalizedString)	0..1	Name of the account holder	“John Doe“
UsageOrder» Types	ebics:UsageOrderType	0..1	Order restrictions for the account in question; if not specified, there are no restrictions as to specific BTF identifiers for the account in question; if the empty tag is used, the account in question has not been activated for any BTF identifiers	“STA IZV”
Service	ebics:RestrictedServiceType for this structure refer to chapter 8.3.6	0.. ∞	Kind of business transaction, identified by the service structure	
OrderInfo	ebics:OrderInfoType (complex)	1..∞	Information on the administrative order types and services /BTF assigned to the customer	- (complex)
AdminOrderType	ebics:OrderTBaseType (→token, length=3, pattern="[A-Z0-9]{3}")	1	The administrative order type assigned to the customer	“BTD”, BTU” (for up-download of business transactions
Service	ebics:RestrictedServ	0..1	Kind of business	

	iceType for this structure refer to chapter 8.3.6		transaction, identified by the service structure	
Description	ebics:Order» DescriptionType (→normalizedString, maxLength=128)	1	Textual description of the kind of order	“SEPA credit transfer“
NumSig» Required	nonNegativeInteger	0..1	Number of ES’s required for the kind of order; default=0, unless specified	2
UserInfo	ebics:UserInfoType (complex)	1..∞	Subscriber information	- (complex)
UserID	ebics:UserIDType (→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})	1	Subscriber ID	“USR100“
UserID@Status	ebics:UserStatusType (→nonNegativeInteger, maxInclusive=99)	1	Subscriber’s state: 1: Ready: Subscriber is permitted access 2: New: Initial state after establishing the subscriber for EBICS ("established") 3: Partly initialised (INI): Subscriber has sent INI file, yet no HIA 4:Partly initialised (HIA): Subscriber has sent HIA order, but no INI file yet 5: Initialised: Subscriber has sent HIA order and INI file 6: Suspended (several failed attempts), new initialisation via INI and HIA possible) 7: New_FTAM: Not applicable 8: Suspended (by the customer's SPR order), new initialisation via INI and HIA possible 9: Suspended (by bank), new initialisation via INI and HIA is not possible, suspension can only be revoked by the bank	1
Name (in UserInfo)	ebics:NameType (→normalizedString)	0..1	Subscriber’s name	“John Doe“
Permission	ebics:PermissionType (complex)	1..∞	Information on the subscriber’s authorisations	- (complex)
Permission» @Authorisa» tionLevel	ebics:Authorisation» LevelType (→token, length=1: "E", "A", "B", "T")	0..1	Signature class for which the subscriber is authorised: “E“=Individual signature, “A“=First	“A“

			signature, “B“=Second signature, “T“=Transport signature. Not to be specified in the case of downloads	
AdminOrderType	ebics:OrderTBaseType (→token, length=3, pattern="[A-Z0-9]{3}")	1	The administrative order type assigned to the customer	“BTD”, BTU” (for up-download of business transactions
Service	ebics:RestrictedServiceType for this structure refer to chapter 8.3.6	0..1	Kind of business transaction, identified by the service structure	
AccountID	ebics:AccountIDType (→token, maxLength=64)	0..∞	Reference to the identification code of an authorised account	“ABCDEFG» abcdefg» 1234567890“
MaxAmount	ebics:AmountType (→ebics:AmountValue» Type →decimal, totalDigits=24, fractionDigits=4)	0..1	Amount upper threshold up to which the subscriber’s signature authorisation is valid (Validity of the reference is enforced by the EBICS XML schema)	5000.00
MaxAmount» @Currency	ebics:CurrencyBaseType (→token, length=3)	0..1	Currency of the maximum amount, according to ISO 4127; if not specified, “EUR” is assumed	“EUR”

Notes on the clarification:
The allocation of account authorisations for the particular subscribers is effected by means of the element `UserInfo/Permission` in the following way:
If the element `AccountID` is not transferred with `UserInfo/Permission`, the administrative order types and service identifier transferred with `UserInfo/Permission` apply automatically to *all* accounts of the respective customer.
However, if the element `AccountID` is transferred with `UserInfo/Permission`, the authorisations transferred with the respective element `UserInfo/Permission/OrderTypes` apply exclusively to the account IDs referenced via `AccountID`.

9.3.2.1.3 Example XML

```
<?xml version="1.0" encoding="UTF-8"?>
<HKDResponseOrderData
 xmlns="urn:org:ebics:H005"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_orders_H005.xsd">
 <PartnerInfo>
   <AddressInfo>
     <Name>John Doe</Name>
     <Street>Elmstreet 1</Street>
```

```
      <PostCode>12345</PostCode>
      <City>Smallville</City>
      <Region>Virginia</Region>
      <Country>USA</Country>
    </AddressInfo>
    <BankInfo>
      <HostID>EBIXHOST</HostID>
    </BankInfo>
    <AccountInfo ID="accid01" Currency="EUR" Description="Girokonto">
      <AccountNumber international="false">123456789</AccountNumber>
      <BankCode international="false" Prefix="DE">50010070</BankCode>
      <AccountHolder>John Doe</AccountHolder>
    </AccountInfo>
    <OrderInfo>
      <AdminOrderType>BTD</AdminOrderType>
      <Service>
          <ServiceName>EOP</ServiceName>
          <MsgName>camt.053</MsgName>
      </Service>
      <Description>Download ISO20022 Account statement</Description>
    </OrderInfo>
    <OrderInfo>
      <AdminOrderType>BTU</AdminOrderType>
      <Service>
          <ServiceName>SCT</ServiceName>
          <MsgName>pain.001</MsgName>
      </Service>
      <Description>SEPA Credit transfer order</Description>
      <NumSigRequired>2</NumSigRequired>
    </OrderInfo>
  </PartnerInfo>
  <UserInfo>
    <UserID Status="1">USR100</UserID>
    <Permission>
<AdminOrderType>BTD</AdminOrderType>
      <Service>
        <ServiceName>EOP</ServiceName>
        <MsgName>camt.053</MsgName>
      </Service>
    </Permission>
  </UserInfo>
  <UserInfo>
    <UserID Status="1">USR200</UserID>
    <Permission AuthorisationLevel="A">
<AdminOrderType>BTD</AdminOrderType>
      <Service>
        <ServiceName>SDD</ServiceName>
        <MsgName>pain.008</MsgName>
      </Service>
      <AccountID>accid01</AccountID>
      <MaxAmount Currency="EUR">6000.00</MaxAmount>
    </Permission>
    <Permission><AdminOrderType>BTD</AdminOrderType>
      <Service>
        <ServiceName>EOP</ServiceName>
        <MsgName>camt.053</MsgName>
      </Service>
    </Permission>
  </UserInfo>
</HKDResponseOrderData>
```

9.4 HTD (retrieve subscriber's customer and subscriber information)

With HTD, the subscriber can retrieve information stored by the bank relating to their company or themselves; however, in contrast to HKD they are not given information on the company's other subscribers.

HTD is an administrative order type of type "download".

9.4.1 HTD request

The HTD request does not contain specific data that goes beyond that named in the general transaction description.

9.4.2 HTD response

Characteristics of the (decoded & decrypted & decompressed) `OrderData` (order data) for HTD: `HTDResponseOrderData`

9.4.2.1.1 XML schema (graphic representation)

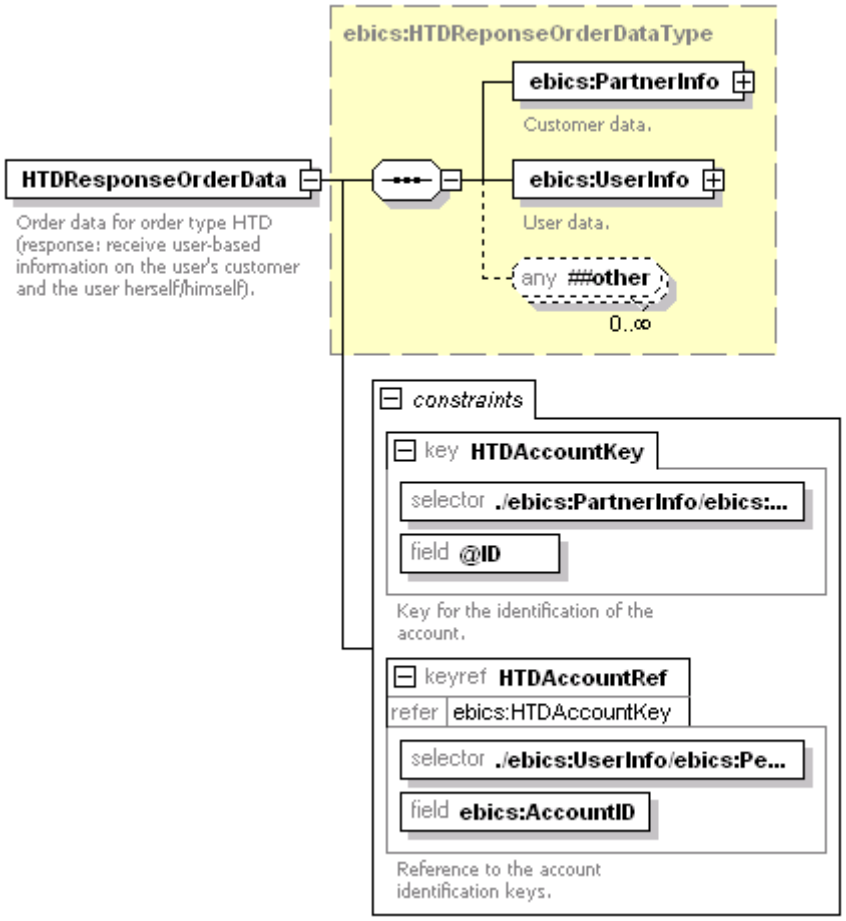


Diagram 97: HTDResponseOrderData

9.4.2.1.2 Meaning of the XML elements/attributes

XML element/ attribute	Data type	#	Meaning	Example
HTDResponse» OrderData	ebics:HTDResponse» OrderDataType (complex)	1	Order data for administrative order type HTD	- (complex)
PartnerInfo	ebics:PartnerInfoType (complex)	1	Customer data	- (complex)
UserInfo	ebics:UserInfo (complex)	1	Subscriber information	- (complex)

For the remaining XML elements and attributes: See administrative order type HKD (Chapter 9.3.2.1.2).
The clarification on the allocation of account authorisations itemised in this chapter applies as well.

9.4.2.1.3 Example XML

```
<?xml version="1.0" encoding="UTF-8"?>
<HTDResponseOrderData
 xmlns="urn:org:ebics:H005"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_orders_H005.xsd">
 <PartnerInfo>
   <AddressInfo>
     <Name>John Doe</Name>
     <Street>Elmstreet 1</Street>
     <PostCode>12345</PostCode>
     <City>Smallville</City>
     <Region>Virginia</Region>
     <Country>USA</Country>
   </AddressInfo>
   BankInfo
     <HostID>EBIXHOST</HostID>
   </BankInfo>
   <AccountInfo ID="accid01" Currency="EUR" Description="Giro account">
     <AccountNumber international="false">123456789</AccountNumber>
     <BankCode international="false" Prefix="DE">50010070</BankCode>
     <AccountHolder>John Doe</AccountHolder>
   </AccountInfo>
   <OrderInfo>
     <AdminOrderType>BTD</AdminOrderType>
       <Service>
            <ServiceName>EOP</ServiceName>
            <Scope>DE</Scope>
            <MsgName>mt940</MsgName>
</Service>         <Description>Download SWIFT daily accounts</Description>
   </OrderInfo>
   <OrderInfo>
     <AdminOrderType>BTU</AdminOrderType>
       <Service>
            <ServiceName>SCT</ServiceName>
            <MsgName>pain.001</MsgName>
       </Service>         <Description>Send SEPA credit transfer order</Description>
```

```
      <NumSigRequired>2</NumSigRequired>
    </OrderInfo>
  </PartnerInfo>
  <UserInfo>
    <UserID Status="1">USR100</UserID>
    <Permission>
      <AdminOrderType>BTD</AdminOrderType>
        <Service>
             <ServiceName>EOP</ServiceName>
             <Scope>DE</Scope>
             <MsgName>camt.053</MsgName>
        </Service>
    </Permission>
    <Permission AuthorisationLevel="A">
      <AdminOrderType>BTU</AdminOrderType>
        <Service>
             <ServiceName>DCT</ServiceName>
             <Scope>BIL</Scope>
             <MsgName>abcd1123</MsgName>
        </Service>
      <AccountID>accid01</AccountID>
      <MaxAmount Currency="EUR">6000.00</MaxAmount>
    </Permission>
  </UserInfo>
</HTDResponseOrderData>
```

9.5 HEV (Download of supported EBICS versions)

By means of HEV the subscriber can inform himself of the EBICS versions supported at the bank's end. The bank's response contains a list of supported EBICS versions and the version of the relevant schema.

HEV is an administrative order type of type “download”.

9.5.1 HEV request

The HEV request retrieves only EBICS versions which are supported by the bank. This request can also be executed by subscribers not initialised. Therefore, an identification and authentication signature is not required.
Only the following information is mandatorily transmitted along with the HEV request:

- Host ID of the EBICS bank computer

The transaction is cancelled and the return code EBICS_INVALID_HOST_ID is returned if the transmitted HostID is unknown on the bank’s side.
Note: This return code is only allowed for the HEV request!

9.5.2 HEV response

The response provides the following information:

- technical return code

- technical report text
- See document “EBICS Annex 1 Return Codes” for the value ranges of both fields. As the EBICS version of the customer system is unknown to the bank system at the time of the HEV request, the bank system assigns values to the fields which are defined in the most updated EBICS version supported at the bank's end. As there is no language-attribute available in the request, the report text is always transmitted in English.
- List of the EBICS versions supported by the bank system and names of the schema versions relevant for these

9.5.3 Schema for HEV request / HEV response

For HEV request und HEV response the neutral schema ebics_hev.xsd is used which is independent of the EBICS versions currently supported by the bank and can be retrieved from https://www.ebics.org/en/ebics-schema. The schema contains request and response. In the case of a request, ebicsHEVRequest must be filled in, in case of a response, ebicsHEVResponse must be filled in.

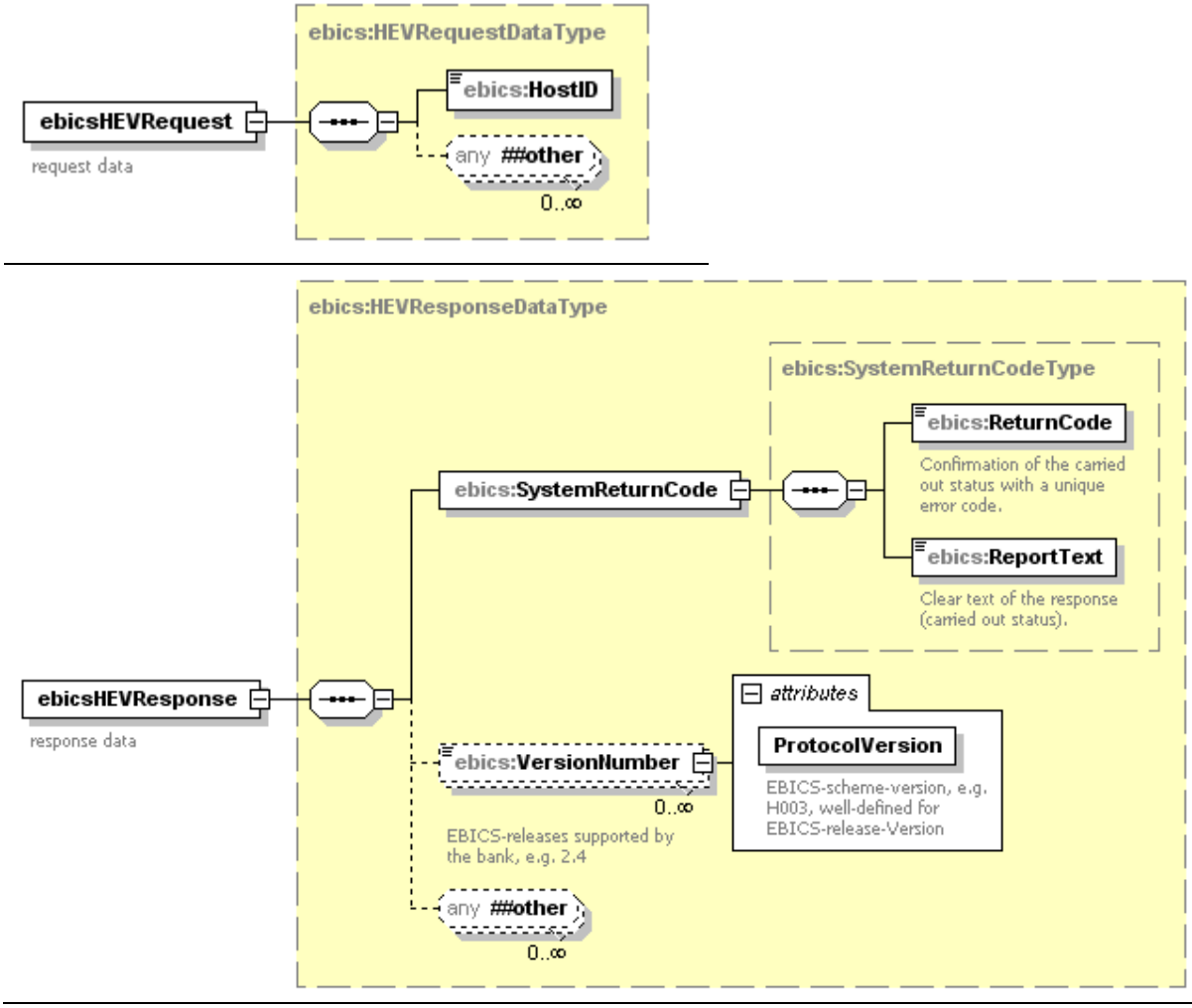


Diagram 98: HEVRequest / HEVResponse

See https://www.ebics.org/en/ebics-schema for the textual representation of the schema ebics_hev.xsd .

9.5.3.1 Meaning of the XML elements and XML attributes of the HEV response

XML element/ attribute	Data type	#	Meaning	Example
`System ReturnCode`	`ebics:SystemReturnCodeType` (complex)	1	Technical return code and error message (in English)	Value range for code according to document "EBICS Annex 1 Return Codes"
`VersionNumber`	`ebics:VersionNumberType` (complex) (→token, length=5, pattern="[0-9]{2}[.][0-9]{2}"	0..∞	EBICS version supported by the bank	02.40 (complies also to 2.4)
`ProtocolVersion`	`ebics:ProtocolVersionType` (→token, length=4)	1	Schema version relevant for the supported EBICS version	H005

9.5.3.2 Example XML for the HEV response

```
<?xml version="1.0" encoding="UTF-8"?>
<ebics:ebicsHEVResponse xsi:schemaLocation="http://www.ebics.org/H000 ebics_hev.xsd"
xmlns:ebics="http://www.ebics.org/H000" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <ebics:SystemReturnCode>
                <ebics:ReturnCode>000000</ebics:ReturnCode>
                <ebics:ReportText>EBICS_OK</ebics:ReportText>
        </ebics:SystemReturnCode>
        <ebics:VersionNumber ProtocolVersion="H004">02.50</ebics:VersionNumber>
        <ebics:VersionNumber ProtocolVersion="H005">03.00</ebics:VersionNumber>
</ebics:ebicsHEVResponse>
```

10 EBICS Customer acknowledgement (HAC)

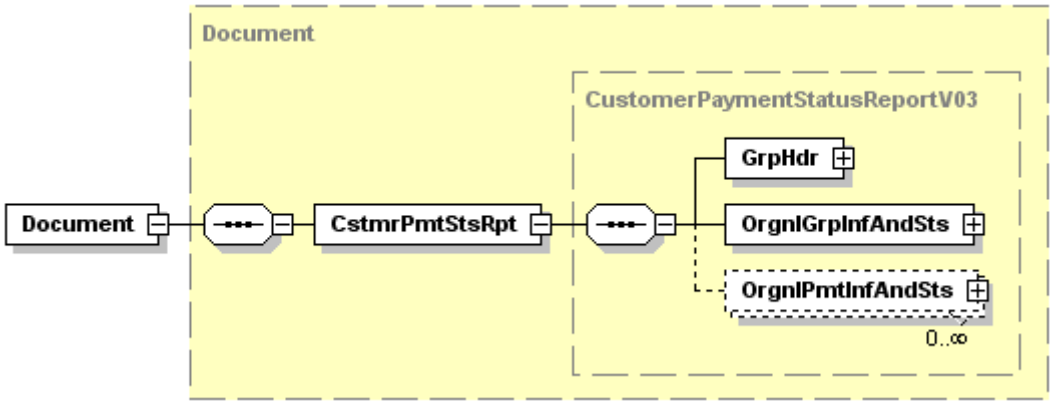
10.1 Preliminary Notes

1. The following stipulations for the allocation of the pain.002 message (ISO Edition 2009) only apply to the EBICS customer acknowledgement (HAC: H label for technical EBICS administrative order type; A = Acknowledgement; C = Customer).
   HAC describes all actions and results that occur while uploading, downloading, or signing files and may give – in addition – information about the content of the order/file (display file).

2. The aim is to use an international standard (schema); we have chosen ISO20022 pain.002
   - Because at present pain.002 is the best alternative although it is not ideal.
   - As a long-term solution, an ISO message especially designed for this purpose should be requested
3. When downloading HAC, the customer receives all status information since the download of the last HAC. It contains all actions and status information of the PartnerID. For this the element group <OrgnlPmtInfAndSts> contains 1..n occurrences. Every occurrence is one protocol step.
   Note: In fact this element group is optional. An essential rule for the EBICS customer acknowledgement is that the element group <OrgnlPmtInfAndSts> occurs at least once.
4. The main focus of the XML-based HAC is the automatic evaluation (suitable preparation by the client system is necessary).

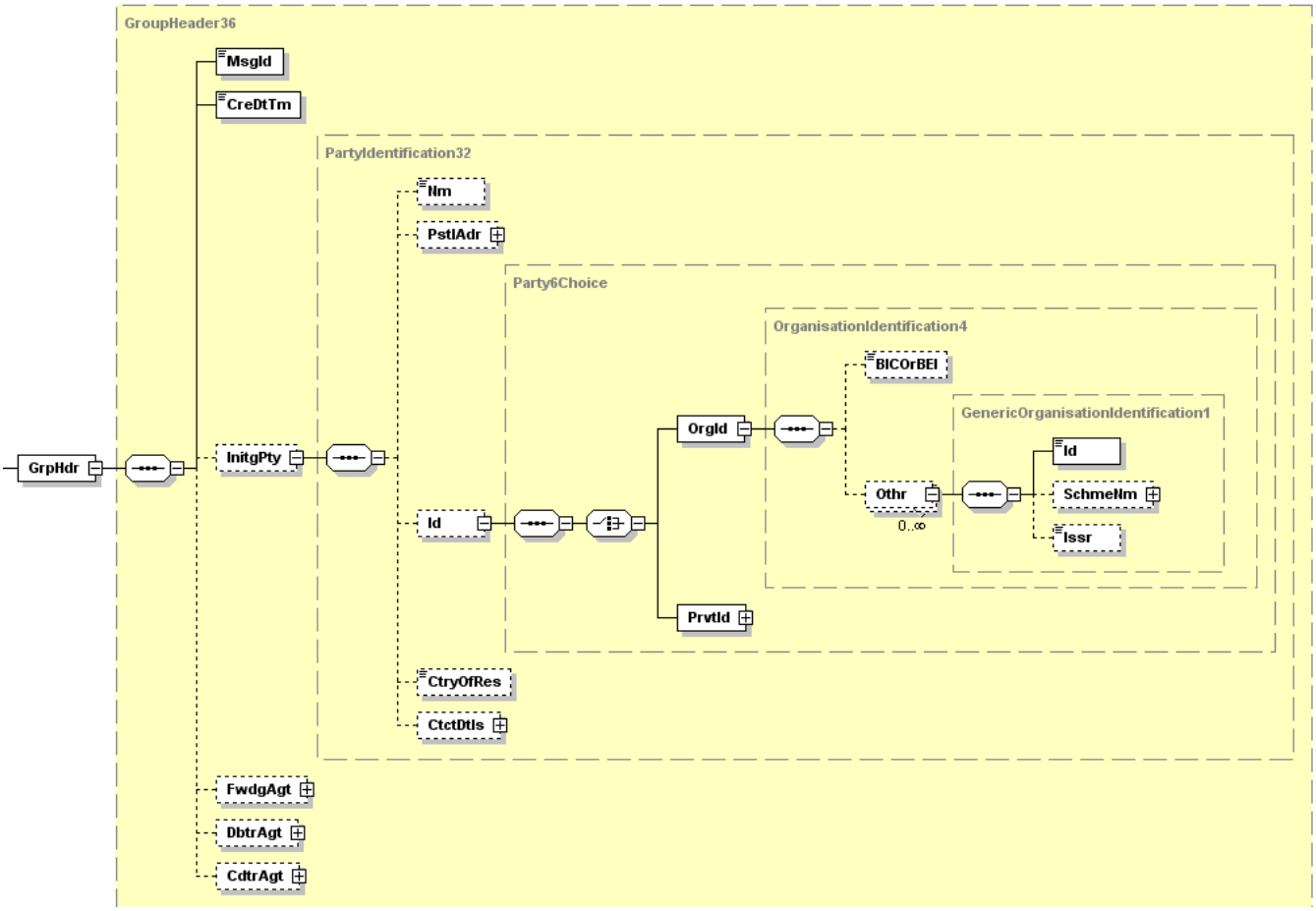
10.2 Allocation of pain.002 for HAC

Complete message – general overview:



10.2.1 Allocation of the element group Group Header

Element group <GrpHdr> - overview:



Stipulations for the allocation:
This element group occurs exactly once.
All elements in <GrpHdr>, which are not mentioned in the list, will never be used in HAC!

Name	XML Tag	Rules for HAC
MessageIdentification	<MsgId>	Mandatory in the ISO schema (and in HAC as well)
CreationDateTime	<CreDtTm>	Mandatory in the ISO schema (and in HAC as well): Creation date/Time of the pain.002 message. Allocation rule: The representation of <CreDtTm> must be the same as specified for EBICS-schema (see EBICS specification chapter 2.3). Example: 2015-05-13T10:00:00Z
InitiatingParty	<InitPty><Id><OrgId>	Element group for the transfer of the HostID (optional in the ISO schema; but mandatory in HAC) HostId (technical ID for the EBICS bank server) to be allocated in <Othr>.

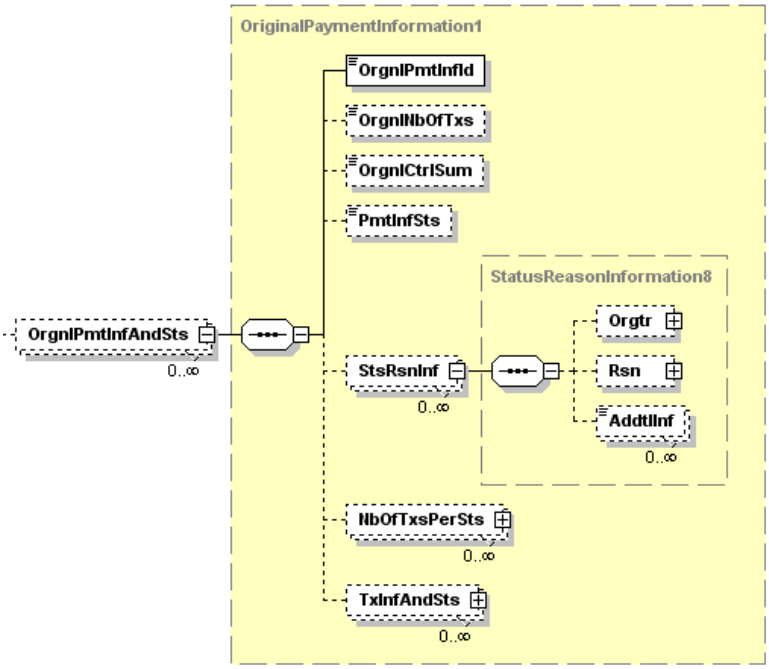
10.2.2 Allocation of the element group Original Group Information and Status

As to ISO this element group occurs exactly once.
There are two mandatory elements: <OrgnlMsgId> and <OrgnlMsgNmId> (both type Max35Text). They state grouping information concerning the original message. By reason that HAC states information on a collection of different orders/actions of a PartnerID, these elements are allocated with the constant „EBICS“.

10.2.3 Allocation of the element group Original Payment Information and Status

This element group is optional in ISO but for HAC it must occur at least once!

Element group <OrgnlPmtInfAndSts> - Overview:



Stipulations for the allocation:
All elements in <OrgnlPmtInfAndSts>, which are not mentioned in the list, will never be used in HAC!

Name	XML Tag	Rules for HAC
OriginalPaymentInforamtionIdentification	<OrgnlPmtInfId>	Mandatory in the ISO schema (and in HAC as well). Information on the type of action; see chapter 10.2.3.1
StatusReasonInformation	<StsRsnInf>	[0..unbounded] in ISO schema, occurrence exactly one time in HAC Information on the order (including all involved users and the timestamp), the result of the action and data for the display file; see chapter 10.2.3.2

10.2.3.1 Type of action

The type of action is allocated in the element <OrgnlPmtInfId>.
The following range of values is defined for that:

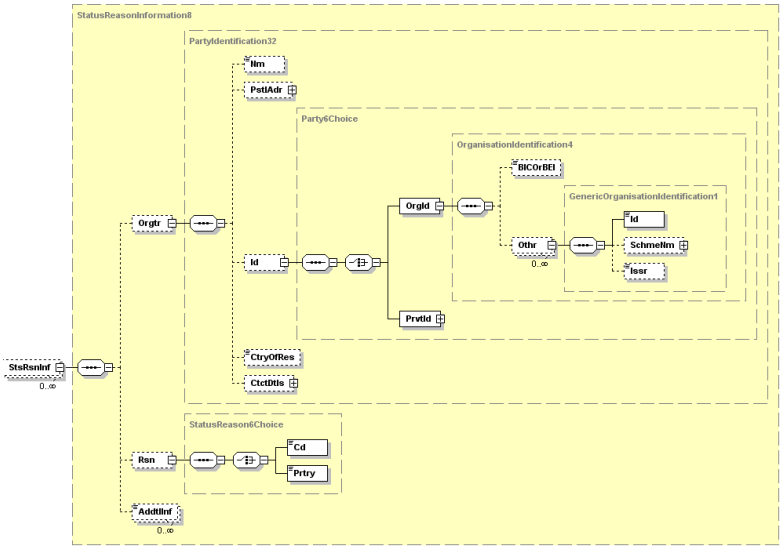
HAC – types of action	Meaning	Range of values (Max35Text); codes defined for EBICS
„type of action“ *transmission*		
File submitted to the bank	Each kind of file upload except the upload of an ES file	FILE_UPLOAD
File downloaded from the bank	Each kind of file download except the download of an ES file	FILE_DOWNLOAD
Electronic signature submitted to the bank	Upload of an ES file using the EDS process (administrative order type HVE)	ES_UPLOAD
Electronic signature downloaded from the bank	Download of an ES file (reserved for later versions)	ES_DOWNLOAD
„type of action“ *postprocessing (EDS etc.)*		
Signature verification	Bank server verifies the transmitted ES	ES_VERIFICATION Code when **no** EDS is used.
Forwarding to EDS	File is stored in the EDS process waiting for the necessary ES's	VEU_FORWARDING
EDS signature verification	Bank server verifies the transmitted ES within the EDS process	VEU_VERIFICATION Code when EDS is used.
End of EDS signature verification	The verification process in the EDS is finished, because the last ES necessary for authorisation of a payment within the EDS was verified successfully. In case of an incorrect last VEU the type of action VEU_VERIFICATION with a result of the action (ISO error code) allowed for this case must be used (see chapter 10.4).	VEU_VERIFICATION_END
Cancellation of EDS order	Order is cancelled by a authorised user within the EDS process	VEU_CANCEL_ORDER
„type of action“ *additional information*	Provision of additional information from bank to customer using HAC (in <AddtlInf>)	ADDITIONAL

One of the “final indication” types of action has to be provided to indicate that no further protocol steps are due for the corresponding order. For this type of action no reason code (in result of action) is allowed. It is simply a label.

„type of action“ *final indication*		
HAC end of order (positive)	With reference to an orderID this label serves as the end label (no protocol step follows) for the orderID. The order is positively completed at EBICS level, which means that either all signatures required for approval are correct and the order can therefore go into further processing **or** that the order has been cancelled by an authorised user (administrative order type HVS).	ORDER_HAC_FINAL_POS
HAC end of order (negative)	With reference to an orderID this label serves as the end label (no protocol step follows) for the orderID. The order could not be processed on EBICS level completely for several reasons (see codes of results of action in the previous HAC step(s) of this order, e.g either due to incorrect signatures or due to a technical error.)	ORDER_HAC_FINAL_NEG

10.2.3.2 Result of action

The result of action is allocated in the element group Status Reason Information.
Element group <StsRsnInf> - Overview:



Stipulations for the allocation:
All elements in <StsRsnInf>, which are not mentioned in the list, will never be used in HAC!

Name	XML Tag	Rules for HAC
Originator / Name	<Orgtr><Nm>	Name of the customer
Originator / Identification / OrganisationIdentification /Other	<Orgtr><Id><OrgId><Othr> 1) <Id> 2) <SchmeNm><Prtry> Note: lenght of Id name = Mmax35text	Element group (0..N occurrences) for different identification codes with the following meaning:Identification ID The code in <SchmeNm><Prtry> identifies the kind of ID in the element <Id> (Most of the ID names are already defined in EBICS; in this case they are provided in EBICS notation): ➢ UserID (Identification of the user) ➢ PartnerID (Identification of the client) ➢ SystemID (Identification of the technical user) ➢ OrderID (order number) ➢ AdminOrderType ➢ ServiceName ➢ Scope ➢ ServiceOption ➢ ContainerType ➢ MsgName ➢ OrderIDRef (If the action refers to another order, the OrderID is allocated here) ➢ AdminOrderTypeRef (If the action refers to another order, the administrative order type is allocated here) and the service identifiers as well: ➢ ServiceNameRef ➢ ScopeRef ➢ ServiceOptionRef ➢ ContainerTypeRef ➢ MsgNameRef ➢ PartnerIDRef (If the action refers to another order AND another Partner, the PartnerID is allocated here) ➢ TimeStamp (Timestamp of the action provided in ISO 8601 format, analogous to chapter 2.3 in the EBICS specification) ➢ DataDigest (Hash value) Note 1: All IDs noted above have to be provided in HAC in case they are available on the bank server. Note 2: If IDs for service elements “Ref” are present (e.g. ServiceNameRef) “the (current) AdminOrderType”can only be a H* order type. Therefore the identifier “ServiceName” to “MsgName” cannot be present.

Name	XML Tag	Rules for HAC
Reason	<Rsn>	Result of the action, table see in chapter 10.3. All types of results are defined as ISO reason codes (to be allocated in <Cd>). This element is mandatory for HAC except: The type of action <OrgnlPmtInfAndSts><OrgnlPmtInfId> contains 1) one of the two “final indication” labels (see chapter 10.2.3.1). In this case it is not permitted to provide <Cd>. 2) “ADDITIONAL”. In this case the allocation is optional.
AdditionalInformation	<AddtlInf>	In case a file is displayed (display of an extraction of the content of a file) there are 1..n occurrences (see chapter 0) Further free text (Max105Text) is always permitted. The use is optional and repeatable at will (it may be used for free text information for customers) and the content of the element <AdditionalOrderInfo> (information regarding the order on the part of the client) – as this information may be up to 255 characters, more than one occurrence is possible)

10.2.3.3 Display file (Use in Germany)

The content for the display file (Information about the file content) is provided in <StsRsnInf><AddtlInf>.
The display file will be delivered with the final indication label (ORDER_HAC_FINAL, refer to chapter 10.2.3.1).
Note: This also applies to files with missing SignatureFlag meaning that the file is not authorised by ES but by accompanying note signed by hand.
The existing character of the display file can be reused: Each line is one occurrence of <AddtlInf>.

Name	XML Tag	Rules for HAC
StatusReasonInformation / AdditionalInformation	<StsRsnInf><AddtlInf>	Only used for specific data in free text: 1. Examples for commonly used formats see chapters 0, 10.2.3.3.2 and 10.2.3.3.3 2. Hash value (only needed for SEPA container file) 3. display file for files without specific format

10.2.3.3.1 Example for SEPA

HAC resulting from a pain.001 message with 3 Payment Information Blocks

<StsRsnInf>
<AddtlInf>G U T S C H R I F T E N</AddtlInf>
<AddtlInf>Datei-ID: 4782647268346</AddtlInf>
<AddtlInf>Datum/Zeit: 28.11.2010/09:30:47</AddtlInf>
<AddtlInf>----------</AddtlInf>
<AddtlInf>Sammlerreferenz : 46573264784</AddtlInf>
<AddtlInf>Bank-Code : WELADEDD</AddtlInf>
<AddtlInf>Kontonummer : DE44300500000054627452</AddtlInf>
<AddtlInf>Auftraggeberdaten : XXX</AddtlInf>
<AddtlInf>Anzahl der Zahlungssaetze : 187</AddtlInf>
<AddtlInf>Summe der Betraege (EUR) : 68.672,00</AddtlInf>
<AddtlInf>Ausfuehrungstermin : 01.12.2010</AddtlInf>
<AddtlInf>----------</AddtlInf>
<AddtlInf>Sammlerreferenz : 46573264783</AddtlInf>
<AddtlInf>Bank-Code : WELADEDD</AddtlInf>
<AddtlInf>Kontonummer : DE44300500000054627452</AddtlInf>
<AddtlInf>Auftraggeberdaten : YYY</AddtlInf>
<AddtlInf>Anzahl der Zahlungssaetze : 165</AddtlInf>
<AddtlInf>Summe der Betraege (EUR) : 354.378,00</AddtlInf>
<AddtlInf>Ausfuehrungstermin : 03.12.2010</AddtlInf>
<AddtlInf>----------</AddtlInf>
<AddtlInf>Sammlerreferenz : 46573264782</AddtlInf>
<AddtlInf>Bank-Code : WELADEDD</AddtlInf>
<AddtlInf>Kontonummer : DE30300500000035351767</AddtlInf>
<AddtlInf>Auftraggeberdaten : XXX</AddtlInf>
<AddtlInf>Anzahl der Zahlungssaetze : 34</AddtlInf>
<AddtlInf>Summe der Betraege (EUR) : 45.100,20</AddtlInf>
<AddtlInf>Ausfuehrungstermin : 01.12.2010</AddtlInf>
</StsRsnInf>
….

10.2.3.3.2 Example for SEPA container

HAC resulting from a container with 2 pain.001 messages

<StsRsnInf>
<AddtlInf>G U T S C H R I F T E N</AddtlInf>
<AddtlInf>Datei-ID: 4782647268346</AddtlInf>
<AddtlInf>Datum/Zeit: 28.11.2010/09:30:47</AddtlInf>
<AddtlInf>----------</AddtlInf>
<AddtlInf>Sammlerreferenz : 46573264784</AddtlInf>
<AddtlInf>Bank-Code : WELADEDD</AddtlInf>
<AddtlInf>Kontonummer : DE44300500000054627452</AddtlInf>
<AddtlInf>Auftraggeberdaten : XXX</AddtlInf>
<AddtlInf>Anzahl der Zahlungssaetze : 187</AddtlInf>

<AddtlInf>Summe der Betraege (EUR) : 68.672,00</AddtlInf>
<AddtlInf>Ausfuehrungstermin : 01.12.2010</AddtlInf>
<AddtlInf>Hash-Wert : 24 AE 87 34 FE BA 22 12</AddtlInf>
<AddtlInf> 34 E4 5A 34 54 33 43 23</AddtlInf>
<AddtlInf> 15 34 55 78 FA F1 33 11</AddtlInf>
<AddtlInf> 93 67 30 03 19 67 BE FA</AddtlInf>
<AddtlInf>G U T S C H R I F T E N</AddtlInf>
<AddtlInf>Datei-ID: 4782647268347</AddtlInf>
<AddtlInf>Datum/Zeit: 28.11.2010/09:30:47</AddtlInf>
<AddtlInf>----------</AddtlInf>
<AddtlInf>Sammlerreferenz : 46573264785</AddtlInf>
<AddtlInf>Bank-Code : WELADEDD</AddtlInf>
<AddtlInf>Kontonummer : DE30300500000035351767</AddtlInf>
<AddtlInf>Auftraggeberdaten : YYY</AddtlInf>
<AddtlInf>Anzahl der Zahlungssaetze : 23</AddtlInf>
<AddtlInf>Summe der Betraege (EUR) : 14.256,00</AddtlInf>
<AddtlInf>Ausfuehrungstermin : 01.12.2010</AddtlInf>
<AddtlInf>Hash-Wert : 29 AE 87 34 FE BA 22 12</AddtlInf>
<AddtlInf> 34 E4 5A 34 54 33 43 23</AddtlInf>
<AddtlInf> 15 34 55 78 FA F1 33 11</AddtlInf>
<AddtlInf> 93 67 30 03 19 67 BE BB</AddtlInf>
</StsRsnInf>
….

10.2.3.3.3 Example for DTAZV (German format used for international payments)

….
<StsRsnInf>
<AddtlInf>G U T S C H R I F T E N</AddtlInf>
<AddtlInf>Bank-Code : 30040000</AddtlInf>
<AddtlInf>Kundennummer : 0000000001</AddtlInf>
<AddtlInf>Auftraggeberdaten : KARL MUSTERMANN</AddtlInf>
<AddtlInf> MUSTERSTR. 1</AddtlInf>
<AddtlInf> 50825 KOELN</AddtlInf>
<AddtlInf>Erstellungsdatum : 10.05.00</AddtlInf>
<AddtlInf>Auftragswaehrung : USD</AddtlInf>
<AddtlInf>Bank-Code : 30040000</AddtlInf>
<AddtlInf>Kontowaehrung : EUR</AddtlInf>
<AddtlInf>Kontonummer : 1234567890</AddtlInf>
<AddtlInf>Ausfuehrungstermin : 10.11.00</AddtlInf>
<AddtlInf>Betrag : 20.000,000</AddtlInf>
<AddtlInf>Anzahl der Datensaetze T : 000000000000001</AddtlInf>
<AddtlInf>Summe der Betraege : 000000000020000</AddtlInf>
</StsRsnInf>
….

10.3 Annex for HAC: External reason codes (result of action)

The following results of action have to be protocolled in the element <Rsn><Cd>.
They are part of the external ISO code list “ExternalStatusReason1Code”:

ISO code	ISO Name	Definition
AM05	Duplication	The data digest of the transmitted order data is already known on the bank server (order data with the same data digest was transmitted recently)
AM21	LimitExceeded	Transaction amount exceeds limits agreed between bank and client
DS01	ElectronicSignaturesCorrect	The Electronic Signature(s) are correct
DS02	OrderCancelled	An authorized user has cancelled the order
DS03	OrderNotCancelled	The user’s attempt to cancel the order was not successful
DS04	OrderRejected	The order was rejected by the bank side (for reasons concerning content)
DS05	OrderForwardedForPostprocessing	The order was correct and could be forwarded for postprocessing
DS06	TransferOrder	The order was transferred to EDS
DS07	ProcessingOK	All actions concerning the order could be done by the EBICS bank server
DS08	DecompressionError	The decompression of the file was not successful
DS09	DecryptionError	The decryption of the file was not successful
DS10	Signer1CertificateRevoked	The certificate is revoked for the first signer.
DS11	Signer1CertificateNotValid	The certificate is not valid (revoked or not active) for the first signer
DS12	IncorrectSigner1Certificate	The certificate is not present for the first signer
DS13	SignerCertificationAuthority Signer1NotValid	The authority of signer certification sending the certificate is unknown for the first signer
DS14	UserDoesNotExist	The user is unknown on the server
DS15	IdenticalSignatureFound	The same signature has already been sent to the bank
DS16	PublicKeyVersionIncorrect	The public key version is not correct. This code is returned when a customer sends signature files to the financial institution after conversion from an older program version (old ES format) to a new program version (new ES format) without having carried out re-initialisation with regard to a public key change.
DS17	DifferentOrderDataInSignatures	Order data and signatures don’t match
DS18	RepeatOrder	File cannot be tested, the complete order has to be repeated. This code is returned in the event of a malfunction during the signature check, e.g. not enough storage space.
DS19	ElectronicSignatureRightsInsufficient	The user’s rights (concerning his signature) are insufficient to execute the order
DS20	Signer2CertificateRevoked	The certificate is revoked for the second signer

ISO code	ISO Name	Definition
DS21	Signer2CertificateNotValid	The certificate is not valid (revoked or not active) for the second signer
DS22	IncorrectSigner2Certificate	The certificate is not present for the second signer
DS23	SignerCertificationAuthority Signer2NotValid	The authority of signer certification sending the certificate is unknown for the second signer
DS24	WaitingTimeExpired	Waiting time expired due to incomplete order
DS25	OrderFileDeleted	The order file was deleted by the bank server (for multiple reasons)
DS26	UserSignedMultipleTimes	The same user has signed multiple times
DS27	UserNotYetActivated	The user is not yet activated (technically)
DS0A	DataSignRequested	Data signature is required *In EBICS this means that the Electronic Signature(s) have not been sent to the bank server yet or that the number of signatures is insufficient*
DS0B	UnknownDataSignFormat	Data signature for the format is not available or invalid. *In EBICS this means that the Electronic signature(s) are incorrect*
DS0C	SignerCertificateRevoked	The signer certificate is revoked *In EBICS this also means that the user is locked*
DS0D	SignerCertificateNotValid	The signer certificate is not valid (revoked or not active). *In EBICS this means that the public key has not been activated yet or certificate is not valid*
DS0E	IncorrectSignerCertificate	The signer certificate is not present. *In EBICS this means that the public key does not exist or certificate is not present*
DS0F	SignerCertificationAuthority SignerNotValid	The authority of the signer certification sending the certificate is unknown
DS0G	NotAllowedPayment	Signer is not allowed to sign this operation type *In EBICS this means that the user has no authorisation rights*
DS0H	NotAllowedAccount	Signer is not allowed to sign for this account
ID01	CorrespondingOriginalFileStillNotSent	Signature file was sent to the bank but the corresponding original file has not been sent yet.
TA01	TransmissonAborted	The transmission of the file was not successful – it had to be aborted (for technical reasons)
TD01	NoDataAvailable	There is no data available (for download)
TD02	FileNonReadable	The file cannot be read (e.g. unknown format)
TD03	IncorrectFileStructure	The file format is incomplete or invalid
TS01	TransmissionSuccessful	The (technical) transmission of the file was successful.
TS04	TransferToSignByHand	The order was transferred to pass by accompanying note signed by hand

10.4 Annex for HAC: Type/result of action (permitted pairs)

If more than one reason code is suitable the most precise should be chosen.

<OrgnlPmtInfId> (Type of action)	Possible/Permitted values for <Rsn><Cd> (Result of action)	Description of the code (shortened, more details see chapter 10.3)
FILE_UPLOAD	AM05 TS01 TA01 DS0C DS08 DS09	Upload aborted Upload successful Upload aborted User locked/certificate revoked Decompression error Decryption error
FILE_DOWNLOAD	TS01 TA01 DS0C DS08 DS09 TD01	Download successful Download aborted User locked/certificate revoked Decompression error Decryption error Not data available for download
ES_UPLOAD	TS01 TA01 DS0C DS08 DS09 ID01	Upload (of ES) successful Upload (of ES) aborted User locked/certificate revoked Decompression error Decryption error Original order file has not been sent before
ES_DOWNLOAD		*still not in use*

<OrgnlPmtInfId> (Type of action)	Possible/Permitted values for <Rsn><Cd> (Result of action)	Description of the code (shortened, more details see chapter 10.3)
ES_VERIFICATION	AM21 TD02 TD03 TS04 DS01 DS0A DS0B DS0C DS0D DS0E DS0F DS0G DS0H DS10 (DS11; DS12) DS20 (DS21; DS22) DS13/ DS23 DS14 DS15DS16 DS17 DS18 DS19 DS24 DS25 DS26 DS27 DS08 DS09	Amount exeeds limit File cannot be read The file format is invalid Not ES-signed file (no SignatureFlag) ES(s) are correct Number of ES(s) insufficient ES(s) are not correct Certificate is revoked / user is locked Certificate is not valid /public key not activated Certificate not present / public key doesn't exist CA for certificate is unknown Signer not allowed to sign this operation Signer not allowed to sign this account Certificate revoked (not valid; not present) for first signer Certificate revoked (not valid; not present) for second signer CA unknown for first/second signer User (means signer) is unknown on the server The same ES already has been sent to bankPublic kexy version not correct order data and ES(s) don't match Repeat order (file not testable) Signer's ES rights are unsufficient Waiting time expired and file deleted by bank File deleted by bank (multiple reasons) Same user signed multiple times User (means signer) not yet activated Decompression error Decryption error
VEU_FORWARDING	DS06	Order transferred to the EDS

<OrgnlPmtInfId> (Type of action)	Possible/Permitted values for <Rsn><Cd> (Result of action)	Description of the code (shortened, more details see chapter 10.3)
VEU_VERIFICATION	AM21 TD02 TD03 DS01 DS0B DS0C DS0D DS0E DS0F DS0G DS0H DS10 (DS11; DS12) DS20 (DS21; DS22) DS13/ DS23 DS14 DS15DS16 DS17 DS18 DS19 DS24 DS25 DS26 DS27	Amount exeeds limit File cannot be read The file format is invalid ES(s) are correct ES(s) are not correct Certificate is revoked / user is locked Certificate is not valid /public key not activated Certificate not present / public key doesn't exist CA for certificate is unknown Signer not allowed to sign this operation Signer not allowed to sign this account Certificate revoked (not valid; not present) for first signer Certificate revoked (not valid; not present) for second signer CA unknown for first/second signer User is unknown on the server The same ES already has been sent to bankPublic kexy version not correct order data and ES(s) don't match Repeat order (file not testable) Signer's ES rights are unsufficient Waiting time expired and file deleted by bank File deleted by bank (multiple reasons) Same user signed multiple times User not yet activated
VEU_VERIFICATION_END	DS05	Order was correct, forwarded for postprocessing
VEU_CANCEL_ORDER	DS02 DS03	Order cancelled Order not cancelled
ADDITIONAL	*Optional*	Note: This is not in the scope of EBICS
ORDER_HAC_FINAL	---	

11 Appendix: Cryptographic processes

11.1 Identification and authentication signature

11.1.1 Process

Identification and authentication signatures are based on the RSA signature process. The following parameters determine the identification and authentication signature process: Length of the (secret) RSA key, hash algorithm, padding process, canonisation process.

For the identification and authentication process, EBICS defines the **process "X002"** with the following parameters:

Parameter	Value
Key length in Kbit	>=2Kbit (2048 bit) and <=16Kbit
Hash algorithm	SHA-256
Padding process	PKCS#1
Canonisation process	http://www.w3.org/TR/2001/REC-xml-c14n-20010315

The optional XML signature fields "KeyInfo" and "Object" remain unfilled.
The transaction is cancelled with return code EBICS_INVALID_REQUEST_CONTENT if X001 is still used in a request.

11.1.2 Format

Identification and authentication signatures are represented in EBICS messages in accordance with the W3C recommendation "Signature Syntax and Processing" ((http://www.w3.org/TR/xmldsig-core/). Hence identifiers of the algorithms for forming the hash value, the signature and the indicator of the canonisation process are components of the identification and authentication signature. Therefore it is not necessary to change the XML interface when a new version of "X00n" is defined with altered parameters. This especially applies for versions that utilise SHA-224, SHA-256, SHA-384 or SHA-512 as a hash function.
When placing the identification and authentication signature in the element `SignatureValue`, it is principally not filled up to the full length of the modulo of the RSA key for generating this signature. .

11.2 Electronic signatures

11.2.1 Process

Electronic signatures are based on the RSA signature process. The processes for generating/verifying electronic signatures are defined in the Appendix (Chapter 14).
EBICS must support Version “A005” or “A006” of the bank-technical electronic signature.

11.2.2 Format

The schema file „ebics_signature_S002.xsd“ contains the element `UserSignatureData` for the signature of the subscriber in EBICS messages. To this end, an instance document is created for “ebics_signature_S002.xsd” that contains `UserSignatureData` for subscriber ES’s as top-level elements. `UserSignatureData` contains a list of elements `OrderSignatureData` for one or more subscriber ES’s (see also Diagram 4).
The XML schema definition file “ebics_orders_H005.xsd” contains the definition of the global elements `BankSignatureData` for embedding the financial institution's electronic signature (As this is an intended feature, the structure is not usable yet, especially `BankSignatureData` still contains an element `OrderSignature` to receive a bank ES in base64 coding (see Diagram 4).

11.2.3 EBICS authorisation schemata for signature classes

EBICS specifies the authorisation schemata for orders that require one or two bank-technical ES’s. Authorisation schemata for orders that require more than two bank-technical ES’s are not described in this standard, although it is not forbidden to transmit more than two ES’s.

E = single signature, A = first signature, B = second signature, T = transport signature (not bank-technical).

Authorisation schema for orders with a minimum ES quantity = 0:

The minimum quantity ES = 0 applies to orders that are authorised via accompanying notes (no SignatureFlag) or for key management orders which require only a transport signature.

Authorisation schema for orders with a minimum ES quantity = 1:

- Authorisation via a single bank-technical ES:
  Authorisation of the order with a single ES can be effected with a single signature.

E	A	B	T
☑	☐	☐	☐

- Authorisation with two bank-technical ES's:
  Authorisation of the order can also take place with 2 ES's of class E, A or B if at least one of these two is a first or a single signature.

first ES → / second ES ↓	E	A	B	T
E	☑	☑	☑	☐
A	☑	☑	☑	☐
B	☑	☑	☐	☐
T	☐	☐	☐	☐

Authorisation schema for orders with a minimum ES quantity = 2:

- With the exception of the combination of two second signatures, authorisation of the order is possible with any combination of two bank-technical ES's.

first ES → / second ES ↓	E	A	B	T
E	☑	☑	☑	☐
A	☑	☑	☑	☐
B	☑	☑	☐	☐
T	☐	☐	☐	☐

In general the following applies:

- There is no maximum ES quantity defined, but in the case of more than two ES the transmitted signatures have to comply with the rules of the authorisation schemas above.
- Individual signatures are fundamentally admissible for authorisation, but are only sufficient in the case of orders where the minimum ES requirement = 0 or ES requirement = 1
- A transport signature never authorises the execution of an order, it only allows the order to be submitted
- The order in which signatures are submitted is irrelevant
- The bank-technical ES's of an order MUST be supplied by different subscribers (if necessary, also different customers).

11.3 Encryption

11.3.1 Encryption at TLS level

11.3.1.1 Process

The customer system and the bank system MUST agree on the use of one of the following procedures (so-called "ciphersuites", see RfCs 2246 and 3268) within the framework of the TLS handshake (details see current EBICS Annex "Transport Layer Security")

11.3.2 Encryption at application level

11.3.2.1 Process

The process for encrypting the order data and ES's of an order is a hybrid process based on the symmetrical encryption process 2-key triple DES and the asymmetrical encryption process RSA.

The order data and ES's of an EBICS transaction are symmetrically encrypted. For each EBICS transaction, a random symmetrical key (transaction key) is generated by the sender of order data and/or ES's that is used for encryption of both the order data and the ES's. The symmetrical key is transmitted to the recipient asymmetrically-encoded.

Based on standard encryption procedures, EBICS defines the encryption procedure "**E002**" with the following **characteristics:**

- Symmetrical encryption algorithm
    - Generation of the transaction key
    - AES-128 (key length 128 bit) in CBC mode
    - ICV (Initial Chaining Value) = 0
    - Padding process in accordance with ANSI X9.23 / ISO 10126-2.
- RSA encryption of the transaction key, key length >= 2Kbit (2048 bits) and <=16Kbit
- Padding process for the RSA encryption: PKCS#1

The **process for asymmetrical encryption** of the transaction key must be adapted for EBICS as follows:

- Minimum length of the (secret) RSA key is 2048
- The padding process conforms with PKCS#1.

Concretely, these adaptations mean:

- The length of PDEK is equal to the length of the RSA key that is used (>= 2048)
- PDEK is generated from DEK via PKCS#1 padding
- EDEK is the result of the RSA encryption of PDEK.

Analogously, the **process for decryption** of the transaction key must also be adapted for EBICS:

- PDEK is the result of the RSA decryption of EDEK
- The 128 lowest-value bits of PDEK form the secret key DEK.

In the context of "E002", the process SHA-256 is used to **form this hash value of the public RSA key**.

11.3.2.2 Formats

The compressed and encrypted ES's and order data segments are embedded in the EBICS messages as base64-coded binary data.

Within the EBICS messages, transmission of the asymmetrically-encrypted transaction key takes place within an XML element of type `DataEncryptionInfoType`. This type is defined in the XML schema definition file ebics_types_H005.xsd and its graphical representation is contained in Diagram 99.

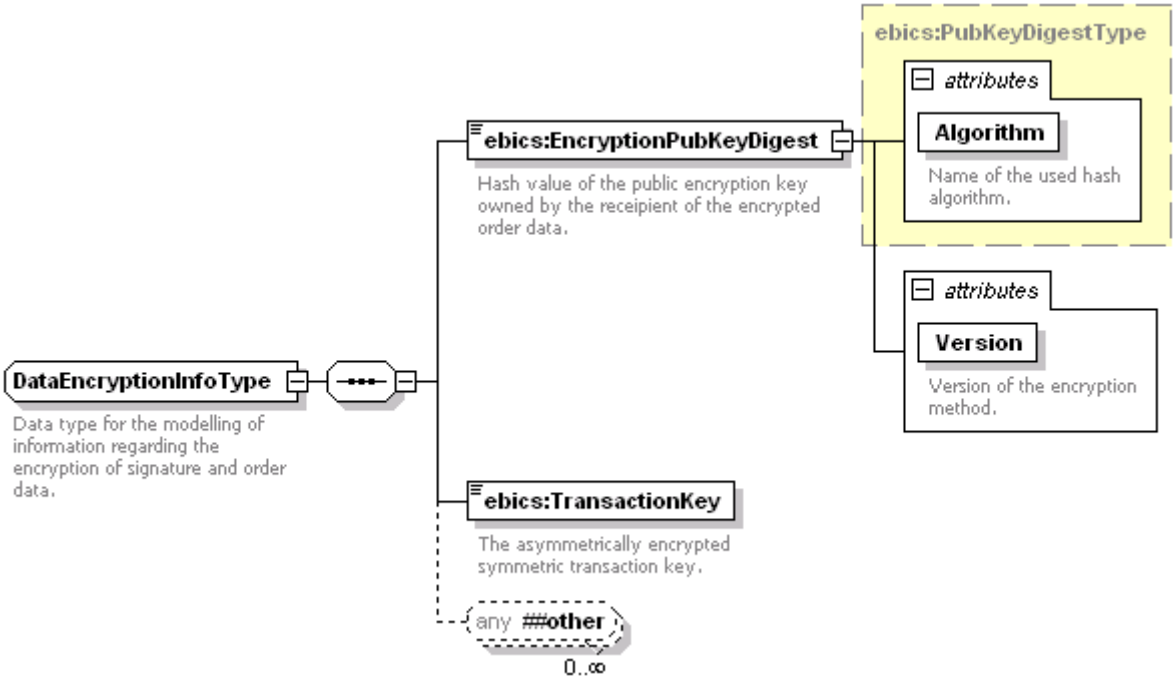


Diagram 99: Definition of the XML schema type DataEncryptionInfoType

The element `ebicsRequest/body/DataTransfer/DataEncryptionInfo` or `ebicsReponse/body/DataTransfer/DataEncryptionInfo`, respectively, of type `DataEncryptionInfoType` is a part of the first EBICS request of an upload transaction (cf. ebics_request_H005.xsd) or the first EBICS response of a download transaction (cf. ebics_response_H005.xsd).

In contrast to the resolution, `DataEncryptionInfoType` does not contain any subscriber details. This is not necessary, since the sender/recipient of the order data is always the initiating party. The subscriber / customer ID of the initiating party is already a component of the control data of the first EBICS request of every EBICS transaction and is firmly assigned to the EBICS transaction.

In addition to the SHA-256 hash value of the certificate in DER binary format, the element `EncryptionPubKeyDigest` also contains the version of the encryption process that is used and the identifier of the hash algorithm used.

Therefore it is not necessary to change `DataEncryptionInfoType` when a new version "E00n" is defined with altered parameters. This especially applies for versions that allow SHA-224, SHA-256, SHA-384 or SHA-512 as one or more of the hash functions.

Please note: In the case where no certificate has been used in V 2.x and the key will be further used in V 3.0 the hash value is only calculated over the public key, i.e. in this case there is no change in the calculation of the hash value compared to the previous EBICS version.

When placing the encrypted transaction key in the element `TransactionKey` it is principally not filled up to the full length of the modulo of the RSA key for the encryption.

11.4 Replay avoidance via Nonce and Timestamp

11.4.1 Process description

The first EBICS request that serves for initialisation of an EBICS transaction contains the elements "**Nonce**" and "**Timestamp"** that are together intended to prevent replaying of this request.

"Nonce" and "Timestamp" form a functional unit for the avoidance of replay:

1. The customer system generates a random "Nonce" and sets a "Timestamp" at the current point in time that the message is sent.
2. The bank system compares the received "Nonce" with a locally-stored list of previously-received "Nonce" values. In addition, it verifies the deviation between the "Timestamp" and the current time. If the "Nonce" that has just been received is present in the stored list or if the deviation of the "Timestamp" is greater than a tolerance period specified by the financial institution, the request is answered with the technical error code EBICS_TX_MESSAGE_REPLAY.
3. If the "Nonce" and "Timestamp" verification was carried out without errors, the bank system stores the "Nonce" and "Timestamp" pair in the local list and continues with the further processing of the message.

The bank system can delete "Nonce"/"Timestamp" pairs whose time stamps lie outside the tolerance period from its list: Messages that contained such a pair would have already been rejected due to the excessive deviation of the "Timestamp". Therefore the fixed tolerance period applies equally to the verification of new pairs as well as the deletion process of stored pairs.

With the elements "Nonce" and "Timestamp", this process guarantees that the first EBICS request of a transaction is unambiguous. This prevents the bank from initialising new EBICS transactions on the basis of old, replayed messages. At the same time, "Timestamp" restricts the chronological necessity of the storage of "Nonce" values by the bank.

11.4.2 Actions of the customer system

11.4.2.1 Generation of “Nonce” and “Timestamp”

The customer system MUST fill out the following fields in the transaction phase “Initialisation”:

- `ebicsRequest/header/static/Nonce` with a cryptographically-strong random number of length 128 bits
- `ebicsRequest/header/static/Timestamp` with the current time stamp for transmission of the EBICS request (date and time in accordance with ISO 8601).

An example of syntactically-correct setting of the values “Nonce” and “Timestamp” is shown in the following XML excerpt:

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
 xmlns="urn:org:ebics:H005"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H005 ebics_request_H005.xsd"
 Version="H005" Revision="1">
 <header authenticate="true">
   <HostID>EBIXHOST</HostID>
   <Nonce>01A56FF768B3B36C5120E9904A7FB035</Nonce>
   <Timestamp>2005-06-22T17:07:34.123+02:00</Timestamp>
   […]
 </header>
 […]
</ebicsRequest>
```

Further information on correct setting of the two XML schema elements can be found under http://www.w3.org/TR/xmlschema-2/#hexBinary (hexBinary) and http://www.w3.org/TR/xmlschema-2/#dateTime (dateTime).

11.4.2.2 Behaviour in the event of error response EBICS_TX_MESSAGE_REPLAY

The bank system uses the technical error code EBICS_TX_MESSAGE_REPLAY to signal that the EBICS message that has just been sent by the client contains a “Nonce” value that corresponds with that stored in the bank system, or that the “Timestamp” lies outside the tolerance period.

When using cryptographically-strong random numbers as “Nonce” and when the financial institution has selected sensible tolerance periods (guideline: a few hours), the likelihood of an accidental collision can be disregarded due to the miniscule possibility of its occurrence.

Therefore after receipt of the report EBICS_TX_MESSAGE_REPLAY, the customer system must take into account the possibility of a replay attack, an intolerably-imprecise clock setting

at the customer's or the bank's end, or an error in its own transaction management in the assignment of "Nonce" values.

If the subscriber would nevertheless like to successfully transmit the EBICS message in question, they must first regenerate the fields `ebicsRequest/header/static/Nonce` and `ebicsRequest/header/static/Timestamp` in accordance with Chapter 11.4.2.1. The remaining contents can be left unchanged.

11.4.3 Actions of the bank system

11.4.3.1 Verification of "Nonce" and "Timestamp"

When the bank system receives an initial EBICS message from a subscriber, it MUST carry out the following actions to verify for message replay. If these verifications are all passed, there is no message replay.

1. **Matching of received "Timestamp" and local time stamp:** Normalised to UTC, the received "Timestamp" must be within the tolerance period that is stretched around the current time stamp of the bank system. This tolerance period will compensate for differences in precision between the clocks involved in the systems and possibly also early/late changeover to summer/wintertime. At the same time, the tolerance period determines when the bank system can delete stored "Nonce"/"Timestamp" pairs. Messages arriving with a "Timestamp" outside of the tolerance period will not be accepted. "Nonce"/"Timestamp" pairs that have been stored in the past and are now outside of the tolerance period can therefore be deleted.
   The tolerance period must be set as a one-off occurrence by the bank system. Here, large values (= large tolerance periods) increase the storage requirements for valid "Nonce"/"Timestamp" pairs whilst low values (= smaller tolerance periods) increase the risk of rejected EBICS messages as a result of excessive clock differences between customer & bank systems.
   If the received "Timestamp" is not within the tolerance period there is a risk of message replay. Therefore the bank system MUST reply with the technical error code EBICS_TX_MESSAGE_REPLAY.
2. **Comparison of the received "Nonce" with the locally-stored "Nonce" values:** All "Nonce"/"Timestamp" pairs that originate from valid EBICS requests within the tolerance period are stored at the bank's end. If the received "Nonce" corresponds with a stored "Nonce" the bank system MUST reply with the technical error code EBICS_TX_MESSAGE_REPLAY.

11.5 Initialisation letters

Initialisation letters for INI contain the public bank-technical subscriber certificate, initialisation letters for HIA contain the subscriber's public identification and authentication certificate and the subscriber's public encryption certificate.

11.5.1 Initialisation letter for INI (example with version A006 of the ES)

User Name	Frank Sample
Date	TT.MM.JJJJ
Time	HH:MM:SS
Host ID	BANKXXXX
Bank	« name of the bank »
User ID	Xxxxxxxx
Partner ID	Yyyyyyyy
Version	*Name of signature version (i.e. A***)*

Signature certificate

Certificate issued to: name-surname or identifier *(only present in case of CA issued certificates)*
Certificate issued by: name of CA *(only present in case of CA issued certificates)*

-----BEGIN CERTIFICATE-----
MIIC6DCCAdCgAwIBAgIIW+dFLrrgUj0wDQYJKoZIhvcNAQELBQAwKjEoMCYGA1UEAwwfRUJJQ1NQQVJUTkVSSURfRUJJQ1NVU0VSSUQ7Qz1GUjAeFw0xNjExMjMxMTAwMDhaFw0yMTExMjQxMTAwMDhaMCoxKDAmBgNVBAMMH0VCSUNTUEFSVE5FUklEX0VCSUNTVVNFUklEO0M9RIIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCgDAuAyZ5Qh18LOcu9H+w9fA+FjgIJoK3WkGF5zsyqWDyH9dlsHo7fp3FQXYaRLGi4VyVrSwgdDOF4gxopOnVO6nYevepqiriBD129YB3rzMxgh/zwuQB60rRyEkr/5mvdddrWpj6RWErRTvQL5CZpeNZ9G/z96sFa7Rzi2W7K2oHr+piiC5moB2cqP54lnzOIx2Z5V5E9w/Fxq8rIQP6XnXu8iZv9bZbF2jy9iED3umEav+9H0Gn67GFxy2i9OkKOGvcmLy9wwiDPF756t2xSrpNhVEFCek8pJPnDkQji93XqtTSfZXezKuT2L59MhuICH4IMVOOD2xaOvsmHYml5AgMBAAGjEjAQMA4GA1UdDwEB/wQEAwIGQDANBgkqhkiG9w0BAQsFAAOCAQEAWONno+PpSXFBciiqS76x0NkYiYTSk3rOUjK/Q/sC+FQX60TgBEbybLXvbGB2fkkUeoQopCuqWkVamJ01tnD+sUPRducSMMh1YERLyTRk/15YjtoeeiEGJNKldGbaR9W6KaTMdY6SOIloyAm/t6HDDhpgL83rN8d5C1uiIpkrPbmqGJ5iOlkJBWzRBxOXAxIGa9OZ4r/RolF7wAwSbfr2cB7rfySrJOUvdZYafwlKZVw8PtSft2JCOnrT5iz0+wGBCIboLiaBOSIO4o6Y+qAC15hbkfdEC5JiK0++vxsDHqHdggtTLU9DP36+KwEPyf8HjW2tCt8F4eucxT8GrfqZcw==
-----END CERTIFICATE-----

Hash of the signature certificate (SHA-256):

72 46 B6 32 85 DF 35 B8
1B 08 66 51 6D E0 FC 1D
DF 35 C7 DD F9 90 EA 6E
A2 C8 A0 ED A9 AB 32 49

I hereby confirm the above public keys for my electronic signature.

Date: **Signature:**

11.5.2 Initialisation letter for HIA (example)

User Name	Frank Sample
Date	TT.MM.JJJJ
Time	HH:MM:SS
Host ID	BANKXXXX
Bank	« name of the bank »
User ID	Xxxxxxxx
Partner ID	Yyyyyyyy
Version	*Name of authentification signature (i.e. X***)* *and name of encryption (i.e. E***)*

Authentification Certificate - Type X***

-----BEGIN CERTIFICATE-----
MIIC6DCCAdCgAwIBAgIIMQ36MhxHh34wDQYJKoZIhvcNAQELBQAwKjEoMCYGA1UEAwwfRUJJQ1NQQVJUTkVSSURfR
UJJQ1NVU0VSSUQ7Qz1GUjAeFw0xNjExMjMxMTAwMDdaFw0yMTExMjQxMTAwMDdaMCoxKDAmBgNVBAMMH0VCSU
NTUEFSVE5FUklEX0VCSUNTVVNFUklEO0M9RIIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDLoPa1O8X
J8L066vGQ9yzXm1NRyvjGxO4c/2GTNxnzAN9egbawaouw/OUyMZ0Pof6zRfcSm4NhxqnxkE18FIpBUzxZDEiDy0CITSHDm
knI7xJibk+zCdnHYE3QS5Kg7CeGXnZm30Gwl4UnOvfhKONgPK8/DRXdZDdzrSkaOt+Xqhi1i8qUerGoAEt7HNrs2gWfnirEBk
sj3Mj0OrTdIwdgWAuLEuk7CnA4gweqsRjha/EaXrQbUB4KJHOS2NsJczh4HKoaoHdEiyFq8Asm4mFhqQmarpv59zzsnnsep+
ho0z59+7ETBaW5KgQZsRE3dbZVHhPPONsPkQs6OLKacn5pAgMBAAGjEjAQMA4GA1UdDwEB/wQEAwIHgDANBgkqhki
G9w0BAQsFAAOCAQEAGw2nxKLtmzpEnPmA89tnSO1lFE49y/aGPYpabWjveR+P1VMglGNp7RWK+NVoZBBJuSBRqnmY
EbieQK2Mo/hShFVpMiEyRB5mdkoRu58PHl4pvpVVUtpVLHjjw8SVKXm8nw6l+8laYwRAuUn63pOI7de3Hy0DsrBYDwcxMpr
8RSC5l4ZZmeKlGwW1GqPzCUu74M+8eqZhlOD2TFruhlTsGO3zeQKeUZ/uy8Y9PMfrjPrtwrrGgx105agdyKHSuY3FkslsVrT
VNOHRQWcMP84zdNL1F2PuZofnJ1zc+unctpq3flHaZ663fUKDMKleKMOoXfQ13Vugg4cHXS/DiaE9A==
-----END CERTIFICATE-----

Hash of the authentification certificate (SHA-256):

```
8E  98  E6  46  FC  E4  E9  5E
3E  50  5B  A2  DF  AA  8D  89
81  38  AE  17  B1  51  D3  12
87  96  F4  C1  FF  BA  E7  82
```

Encryption Certificate - Type E***

-----BEGIN CERTIFICATE-----
MIIC6DCCAdCgAwIBAgIIbZ163qos4DQwDQYJKoZIhvcNAQELBQAwKjEoMCYGA1UEAwwfRUJJQ1NQQVJUTkVSSURfR
UJJQ1NVU0VSSUQ7Qz1GUjAeFw0xNjExMjMxMTAwMDhaFw0yMTExMjQxMTAwMDhaMCoxKDAmBgNVBAMMH0VCSU
NTUEFSVE5FUklEX0VCSUNTVVNFUklEO0M9RIIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC3K5Lgva6
kZizCJVgFTIMjwjLvPl6wfWWP4ei/eqABeslZ6Zv9z/EauplDZzK0ulyIuCwtyO7V36EiCLZ0VS7V20izpbIllwVyYVi950/Q9Pznvz0
p9KvwquheXLFHTwdUATuAEKHT8wc2347j5vRfYCxjxk1Xgk8sgbnyXBwJxy/XkaaALxEfY/60jUz7ip1jIlB4AH03I+qn5lsI1d7E
ZFCTNwGIHXRgivLiNJ7sF5Q+MTqwZ2kYgReBzY6rDD7SMaOlcfopqDDkbyayohKnJxaSXUCpLTKjHfi2ZPhCxwyZRTKG16
y1jscbb4Asocmcsrg/SeHY/G1YERe6Xfa/AgMBAAGjEjAQMA4GA1UdDwEB/wQEAwIFIDANBgkqhkiG9w0BAQsFAAOCAQE
AfINilMSoo1O4ms3qQTTEH6KJLrC9UYRJMzTO0YpTXROOB4n3NHG/q1ElZU41UB8VcCrpWKWBroqx98oRNrFyOD1wGj
B+ine5bxT71ncALEk7ZneUSE3anZKaQV6mZbaJWRq/HSNTQ3G6Ml1LZ8/ZFy5Bt+VnYIXG/tASv5U/jW0+67ceNs/j94zzrH9
auvL7h6PP6260znqGKKgxuX6+XMT21ff7jyG3h+BwGWrtCwU1qmbZGW3wRYTR6x9kCoiV+WI5gqWSpOoai7Oi7nBGklrLb
eOL3UFSvrfjEhpS9az45vn0ldFs+C8eYSgs+ZsBWNcuu8UaLt9S57UmlaR8Gg==
-----END CERTIFICATE-----

Hash of the encryption certificate (SHA-256):

```
E3  FA  11  A3  A4  40  CF  29
6D  25  1B  09  F4  1A  38  F7
33  E5  3A  96  FF  DF  6C  5F
30  DF  B2  9D  72  40  3E  D7
```

I hereby confirm the above public keys for my electronic signature.

Date: **Signature:**

11.6 Generation of the transaction IDs

Transaction IDs are cryptographically-strong random numbers with a length of 128 bits. This means that the likelihood of any two bank systems using the same transaction ID at the same time is sufficiently small.

Transaction IDs are generated by cryptographic pseudo-random number generators (PRNG) that have been initialised with a real random number (seed). The entropy of the seed should be at least 100 bits.

12 Appendix: Overview of selected EBICS details

12.1 Optional EBICS features

With EBICS, not all functions are defined as mandatory. Financial institutions that implement the EBICS standard are free to support some administrative order types or functions within a transaction sequence.

12.1.1 Optional administrative order types

The following EBICS order types CAN be supported by a financial institution (i.e. they are optional):

- HAA (download retrievable services / BTF)
- HKD (download customer's customer and subscriber data)
- HTD (download subscriber's customer and subscriber data)

12.1.2 Optional functionalities in the course of the transaction

A financial institution or a customer product CAN support the following EBICS functionalities (i.e. they are optional for both sides):

- Preliminary verification (see Chapters 3.6 and 5.3)
- Recovery (see Chapters 3.4 and 5.4).

12.2 EBICS bank parameters

With EBICS administrative order type HPD (see also Chapter 9.2), the subscriber can receive information relating to the financial institution's specific access (`AccessParams`) and protocol parameters (`ProtocolParams`).

Access parameters (`AccessParams`):

Parameter name	#	Meaning	Example
`URL`	1..∞	URL or IP address for electronic access to the financial institution It is possible to specify several URLs. Every URL with a valid_from-date that has been reached (or if the corresponding field is empty) is valid. If a URL cannot be reached the customer may use another valid address.	"www.die-bank.de"
`URL@valid_from`	0..1	Commencement of validity of URL/IP. If not specified, the entry is valid with immediate effect	"2005-01-30T15:30:45.123Z"
`Institute`	1	Designation of the financial institution	"Die Bank"
`HostID`	0..1	ID of the EBICS bank system	"bank01"

Protocol parameters (`ProtocolParams`):

Parameter name	#	Meaning	coll. admin. order types
`Version`	1	Permitted versions (listed in each case) for EBICS protocol (`Protocol`), encryption (`Encryption`) signature (`Signature`) and identification and authentication (`Authentication`)	all
`Recovery`	0..1	Support for the recovery of transactions	all
`PreValidation`	0..1	Support for preliminary verification. If this parameter is set, the financial institution thereby ensures that it checks at least a part of the data that is transmitted by the subscriber within the framework of preliminary verification. However, the financial institution is not obliged to comprehensively verify the data	uploads
`ClientDataDownload`	0..1	Support of administrative order types HKD (download customer data, Chapter 9.3) and HTD (download subscriber data, Chapter 9.4).	HKD, HTD
`DownloadableOrder» Data`	0..1	Support of administrative order type HAA (download retrievable BTF, Chapter 9.1).	HAA

12.3 Security media of bank-technical keys

EBICS defines the following value categories for specification of the security medium of (secret) bank-technical keys:

Security medium	Setting
No specification	0000
Diskette	01dd
Chipcard	02dd
Other removable storage medium	03dd
Non-removable storage medium	04dd

In the above table, “dd” represents any number combination that is specified individually by each institution.

12.4 Patterns for subscriber IDs, customer IDs, order IDs, hostIDs

The following table specifies the patterns of different IDs that are permitted in EBICS. In addition, for each ID all of the XML types that are used in EBICS are listed to record corresponding IDs.

ID	Subscriber ID / ID of the technical subscriber	Customer ID	Order ID	Host ID
Pattern	[a-zA-Z0-	[a-zA-Z0-	[A-Z]{1}[A-Z0-	---

	9,=]{1,35}	9,=]{1,35}	9]{3}	
XML type **defined in ebics_types_H005.xsd**	Both of the type `UserIDType`	`PartnerIDType`	`OrderIDType`	`HostIDType`

13 Appendix: Complete List of Administrative Order Type Identifiers

The administrative order types in the following tables are explained in detail in Chapters:

4 Key management
8 Distributed electronic signature
10 HAC and
9 Other administrative order types).

Identification	Direction of trans-mission	Text	Optional/ mandatory support by EBICS bank server solutions
BTD	D	Download of a file identified by a BTF structure	Mandatory
BTU	U	Upload of a file identified by a BTF structure	Mandatory
HAA	D	Download retrievable order types	Optional
HAC	D	Download customer acknowledgement (XML-format)	Mandatory
HCA	U	Send amendment of the subscriber key for identification and authentication and encryption	Mandatory
HCS	U	Transmission of the subscriber key for ES, identification and authentication and encryption	Mandatory
HEV	D	Download supported EBICS versions	Mandatory
HIA	U	Transmission of the subscriber key for identification and authentication and encryption within the framework of subscriber initialisation	Mandatory
HKD	D	Download customer's customer and subscriber data	Optional
HPB	D	Transfer the public bank key (download)	Mandatory
HPD	D	Download bank parameters	Mandatory
HTD	D	Download subscriber's customer and subscriber data	Optional
HVD	D	Retrieve EDS state	Mandatory
HVE	U	Add EDSsignature	Mandatory
HVS	U	Cancellation of orders in the EDS	Mandatory
HVT	D	Retrieve EDS transaction details	Mandatory
HVU	D	Download EDS overview	Mandatory
HVZ	D	Download EDS overview with additional informations	Mandatory
H3K	U	Transmission of all public keys (subscriber key, key for identification and authentication and key for encryption) for initialisation in case of CA-issued certificates	Optional

The concrete use of the administrative order types necessary for the electronic distributed signature has to be agreed in the contract betweeen customer and his bank.

Further administrative order types for the key management:

Identification	Direction of transmission	Text	Format
INI	U	Send password initialisation	Customer's public key for the ES (see Appendix Chapter 14)
PUB	U	Send public key for signature verification	Customer's public key for the ES (see Appendix Chapter 14)
SPR	U	Suspension of access authorisation	Transmission of an ES file with a signature for a dummy file that only contains a space (mandatory)

14 Appendix: Signature process for the electronic signature

The utilised security processes must provide the electronic signature for the data that is to be transmitted. In doing this, the following requirements profile is to be fulfilled:

- The signature may only be provided by the signatory so that the signatory cannot deny the signature and so that it can be verified that the origin of any misuse can only be the responsibility of the signatory.
- All potential recipients must be able to verify the correctness of the signature, wherein it must be additionally guaranteed that this verification is also possible at a later point in time (e.g. by legal entities).
- The signature must be in direct connection to the signed data contents so that it simultaneously authenicates the corresponding data contents, allowing any potential recipient (especially legal entities, even at a later point in time) to also verify the data contents by means of the signature (data integrity verification).
- The signature solution must be applicable to any contents.
- From a performance viewpoint, the signature process must be useable on less-powerful PCs with passable computing performance.
- The administration requirement for necessary storage of the data required for generation of the signature, and especially verification of the signature (identifications) must be as low as possible (simple key management).
- The concrete technical solution must be compatible with common operating systems that may be used by the signatory and the recipient.
- The characters restricted to the operating system (CR, LF and Ctrl-Z) are not included in the calculation of hash values of the A005/A006 ES.

This requirements profile can only be fulfilled by the use of asymmetrical cryptographic processes.

Use of the electronic signature is strongly recommended for all data transmissions that do not serve purely for information acquisition, insofar as an alternative is not agreed in the special arrangements for individual processes.

A detailed description of the mathematical processes and data structures used must be published free of charge for each security process that is used. This description must be sufficient to allow a functionally-compatible product to be created by any manufacturer. Furthermore, a positive certificate of conformity for the process as a whole and in particular the mathematical procedures utilised therein must be provided by an accreditation agency specified by the German banking sector.

With due consideration for these requirements, it is mandatory that the electronic signature process described in the following text is supported by the bank from 1st April 2002.

14.1 Version A005/A006 of the electronic signature

With due consideration for the requirements in chapter 14, it is mandatory that the electronic signature process described in the following text is supported by the bank.
For the signature processes A005 and A006 an interval of 2048 bit (minimum) and 4096 bit (maximum) is defined for the key length.

14.1.1 Preliminary remarks and introduction

The following sub-chapters of chapter 14.1 contain the description of two new signature mechanisms. The two signature mechanisms are both based on the signature schemes of [PKCS1] and the usage of SHA-256 as algorithm for the hashing, but differentiated by the usage of different methods of [PKCS1] for padding.
Since the completion of [A005] the naming for the signature mechanisms has been changed. In contrast to [A005], where the two new signature mechanisms still have been named A005_V1.5 and A005_PSS, the mechanisms will be called A005 and A006 in future. The following table shows the relationship between future names, the old names of [A005] and the names used in [PKCS1]:

future name	name in [A005]	[PKCS1]
A005	A005_V1.5	EMSA-PKCS1-v1_5 with SHA-256
A006	A005_PSS (with SHA-256 hash value as input)	EMSA-PSS with SHA-256 (with SHA-256 hash value as input)

The following description of the two new signature mechanisms is based on the corresponding paragraphs of the specification of SECCOS 6 [SECCOS6]. Both signature mechanisms will be supported by a ZKA signature card, which is based on SECCOS 6 and which contains the ZKA signature application [ZKASigAnw].
For the calculation of an electronic signature the ZKA signature application [ZKASigAnw] offers two different keys, the so called AUT-key and the so called DS-key. Since banking applications will in future use for the calculations of electronic signatures the AUT-key as well as the DS-key, the following special conditions of SECCOS 6 for the usage of these keys must be taken into account:

- For the AUT-key the signature will be calculated using the command INTERNAL AUTHENTICATE. If used with the PSS padding of [PKSC1], the SECCOS smart card will always calculate a hash value over the input data within the execution of the command INTERNAL AUTHENTICATE. Since the application usually also calculates a hash value over the actual message M before calling INTERNAL AUTHENTICATE, this procedure will result in calculating the hash value twice, i. e. the value hash(hash(M)) will be calculated.

- For this reason A006 will be defined in such a way that a prior calculated hash value over the message M will be used as input for the signature mechanism rather than the message M itself.

The asymmetric cryptographic algorithms supported by the SECCOS ICC are based on the RSA algorithm with odd public key exponent ([RSA]).
In chapter 14.1.2 of this document, the principle of construction and the **key components** of the public and private RSA keys according to annex F of [EMV CA], and [PKCS1] for odd public exponents are explained.
A ***signature algorithm*** consists of an algorithm for signature generation and an inverse algorithm for message recovery. The standard signature algorithm supported by the ZKA SECCOS ICC is described in chapter 14.1.3 of this document.
The described signature algorithm based on the RSA algorithm is used by the ZKA SECCOS ICC only in the context of signature mechanisms. A ***signature mechanism*** defines, in which way a message M is transformed into a byte sequence which serves as input for the signature generation by a signature algorithm. The byte sequence generated by a signature mechanism is referred to as ***Digital Signature Input*** (***DSI***).
The ZKA SECCOS ICC supports several signature mechanisms. In chapter 14.1.4 of this document, the new so called A005 and A006 mechanisms are described which are both based on PKCS #1 padding and the usage of SHA-256 as hash algorithm.

14.1.2 RSA

An RSA key pair consists of

- a **public key P_K** and
- a **private key S_K**.

The public and private key consist of **key components**. RSA keys are also called **asymmetric keys**.
For the generation of an RSA key pair with an **odd public key exponent e**, two different **primes p and q (prime factors)** are used. e must be coprime to (p-1) and (q-1).
The corresponding **private exponent d** is defined by

$$e*d \equiv 1 \bmod kgV(p-1, q-1).$$

The primes p and q as well as the private exponent d have to be kept secret.
The product of the primes n = p*q is called **modulus**.
The **public key P_K** of the RSA key pair consists of the components

- **modulus n** and
- **public exponent e**.

The **private key S_K** of the RSA key pair may be represented by components in two ways (see [PKCS1]):

1. Representation of S_K by the components:
   - **modulus n** and
   - **private exponent d**,
2. Representation of S_K by the components:

- **prime factor p,**
- **prime factor q,**
- **d_p = d mod (p-1),**
- **d_q = d mod (q-1) and**
- **qInv = q^{-1} mod p.**

Of the first representation, only the component d has to be kept secret. The components of the second representation are called **Chinese Remainder Theorem-Parameters** (**CRT parameters**). All CRT parameters have to be kept secret.
The SECCOS ICC shall support the RSA algorithm with any odd public key exponent. In most cases one of the odd public key exponents 3 or $F_4 = 2^{16}+1$ is used.

In this document the following notation is used:
k denotes the bit length of the modulus n of an RSA key pair.
k is defined unambiguously by the equation $2^{k-1} <= n < 2^k$.
n is represented by a bit sequence:

$n = b_k\ b_{k-1} \ldots b_1$, with $b_k <> 0$.

The integer value of n is defined by the leftmost bit b_k being the most significant bit and the rightmost bit b_1 being the least significant bit of the binary representation of n.
For k there exist unique digits N >= 1 and 8 >= r >=1 with k = 8*(N-1) + r such that n may also be represented by the bit sequence:

$n = b_r\ b_{r-1} \ldots b_1\ b_{8*(N-1)} \ldots b_{8*(N-2)+1} \ldots b_8 \ldots b_1$.

If r = 8, n may be represented as a sequence of N byte:

$n = B_N\ B_{N-1} \ldots B_1$, with $B_N <>$ '00'.

If r < 8, the bit sequence $b_r\ b_{r-1} \ldots b_1\ b_{8*(N-1)} \ldots b_{8*(N-2)+1} \ldots b_8 \ldots b_1$ 8-r leading binary 0's are added:

$n = 0 \ldots 0\ b_r\ b_{r-1} \ldots b_1\ b_{8*(N-1)} \ldots b_{8*(N-2)+1} \ldots b_8 \ldots b_1$.

In this way n may be represented as a byte sequence

$n = B_N\ B_{N-1} \ldots B_1$, with $B_N <>$ '00'.

The integer value of n is not changed by the introduction of leading 0's in the binary representation of n. Therefore the integer value of n is the same, whether n is represented by a sequence of N byte or by a sequence of k bit.
N is the byte length of n.
N is defined unambiguously by the equation $2^{8*(N-1)} <= n < 2^{8*N}$

14.1.3 Standard digital signature algorithm

14.1.3.1 Standard signing function

Let S_K be a private RSA key consisting of the modulus n and the private key exponent d or consisting of CRT parameters. The associated public RSA key P_K consists of the modulus n and the public key exponent e.
Then a binary coded byte sequence x, its integer value between 0 and n-1 resulting from the binary representation of x, may be signed with S_K. Then x may be represented as a byte sequence with a length of N byte and as a bit sequence with a length of k bit. The k-th bit of the representing byte or bit sequence may have the value 1, but does not have to. If existent, the bit b_{8*N} ... b_{k+1} of the representing byte sequence have the value 0.
The following notation is used for the generation of a signature with the private key S_K consisting of n and d:

$$sign(S_K)[x] = x^d \bmod n$$

If the private key S_K is represented by CRT parameters, $sign(S_K)[x] = x^d \bmod n$ shall be computed as follows:

$$sign(S_K)[x] = s_2 + h*q$$

where s_2 and h shall be computed as follows:

$$s_1 = x^{dp} \bmod p,$$

$$s_2 = x^{dq} \bmod q,$$

$$h = qInv*(s_1 - s_2) \bmod p.$$

The exponentiations $x^d \bmod n$, $x^{dp} \bmod p$ and $x^{dq} \bmod q$ shall be performed with the integer value resulting from the binary representation of x.
The result of the signature generation is a byte sequence s resulting from the binary representation of the integer value of the exponentiation $x^d \bmod n$ or from the binary representation of the integer value of $s_2 + h*q$. The integer value is between 0 and n-1. Then s may be represented as a byte sequence with a length of N byte and as a bit sequence with a length of k bit. The k-th bit of the representing byte or bit sequence may have the value 1, but does not have to. If existent, the bit b_{8*N} ... b_{k+1} of the representing byte sequence have the value 0.

14.1.3.2 Standard recovery function

Let P_K be a public RSA key consisting of the modulus n and the public key exponent e.
The plaintext may be recovered using P_K from a binary coded byte sequence s, if the integer value, resulting from the binary representation of s, is between 0 and n-1. Then s may be represented as a byte sequence with a length of N byte and as a bit sequence with a length of k bit. The k^{-th} bit of the representing byte or bit sequence may have the value 1, but does not have to. If existent, the bit b_{8*N} ... b_{k+1} of the representing byte sequence have the value 0.
The following notation is used for the plaintext recovery:

$$recover(P_K)[s] = s^e \bmod n$$

The exponentiation s^e mod n shall be performed with the integer value resulting from the binary representation of s.
The result of the plaintext recovery is an integer value between 0 and n-1. It may therefore be represented as a byte sequence with a length of N byte and as a bit sequence with a length of k bit. The k^{-th} bit of the representing byte or bit sequence may have the value 1, but does not have to. If existent, the bit b_{8*N} ... b_{k+1} of the representing byte sequence have the value 0.
It is valid for a RSA key pair P_K and S_K:

recover(P_K)[sign(S_K)[x]] = x

14.1.4 Signature Mechanisms A005 and A006

The digital signature mechanisms A005 and A006 are both based on the industry standard [PKCS1] using the hash algorithm SHA-256. They are both signature mechanisms without message recovery.
A **hash algorithm** maps bit sequences of arbitrary length (**input bit sequences**) to byte sequences of a fixed length, determined by the Hash algorithm. The result of the execution of a Hash algorithm to a bit sequence is defined as **hash value**.
The hash algorithm SHA-256 is specified in [FIPS H2]. SHA-256 maps input bit sequences of arbitrary length to byte sequences of 32 byte length. The padding of input bit sequences to a length being a multiple of 64 byte is part of the hash algorithm. The padding even is applied if the input bit sequence already has a length that is a multiple of 64 byte.
SHA-256 processes the input bit sequences in blocks of 64 byte length.
The hash value of a bit sequence x under the hash algorithm SHA-256 is referred to as follows:

SHA-256(x)

For building the value of the **Digital Signature Input (DSI)** out of the hash value [PKCS1] defines two different encoding methods, called EMSA-PKCS1-v1_5 and EMSA-PSS. Therefore two different digital signature mechanisms will be defined based on these two encoding methods. The different mechanisms will be denoted A005 and A006.

14.1.4.1 Signature Mechanism A005

For the computation and verification of a digital signature with the signature mechanism described in [PKCS1] using the encoding method EMSA-PKCS1-v1_5, the following points have to be indicated:

- the hash algorithm HASH to be used,
- the byte length H of the generated hash values,
- the signature algorithm to be used and
- the maximal byte length N of the generated DSI to be allowed as input for the signature algorithm.

The digital signature mechanism A005 is identical to EMSA-PKCS1-v1_5 using the hash algorithm SHA-256. The byte length H of the hash value is 32.
Within ZKA smart cards RSA is used as signature algorithm. Therefore N is the byte length of the modulus n of the applied RSA key.
In the following, digital signature generation and verification on the basis of the digital signature mechanism A005 are described. The used abbreviations are defined in chapter 14.1.2.

14.1.4.1.1 Digital signature generation

According [PKCS1] (using the method EMSA-PKCS1-v1_5) the following steps shall be performed for the computation of a signature for message M with bit length m.

1. The hash value HASH(M) of the byte length H shall be computed. In the case of A005 SHA-256(M) with a length of 32 bytes.

2. The DSI for the signature algorithm shall be generated.

   The DSI is a sequence of N-1 byte constructed as follows:

Denotation	Byte length	Value
Block type	1	'01'
Padding field	N-3-D	'FF..FF'
Separator	1	'00'
Digest-Info	D	BER-TLV coded data object with OID and parameters of the hash algorithm and with the hash value HASH(M)

   Using SHA-256 the Digest-Info is structured as follows:

Tag	Length (in byte)	Value	Description
'30'	'31'		Tag and length of SEQUENCE
'30'	'0D'		Tag and length of SEQUENCE
'06'	'09'	'60 86 48 01 65 03 04 02 01'	OID of the SHA-256 (2 16 840 1 101 3 4 2 1)
'05'	'00'	-	TLV coding of ZERO
'04'	'20'	'XX..XX'	hash value

   The byte length D of the Digest-info has the value 51. The padding field has a length of N-54 byte. Since N has at least the value 128 (for the minimal key length of 1024 bits), it must be padded at least with 74 byte 'FF'.

3. A signature shall be computed using the DSI with the standard algorithm for the signature generation described in section 14.1.3.1.

Since the DSI is a byte sequence of length N-1, the integer value resulting from the binary representation of the DSI is always less than the value of the modulus n.

The signature may be represented as a byte sequence with the byte length N. In the representation of the modulus n as a byte sequence the bit b_k has the value 1 and the bit b_{8*N} b_{8*N-1} ... b_{k+1} have, if existent, the value 0. In the representation of the signature as a byte sequence the bit b_{8*N} b_{8*N-1} ... b_{k+1} therefore also shall have the value 0.

14.1.4.1.2 Digital signature verification

According to [PKCS1] (using the method EMSA-PKCS1-v1_5) the following steps shall be performed for the verification of a signature. The signature to be verified and the message M' require to be available as byte sequences.

1. The signature must be represented as a byte sequence with the byte length N. In the representation of the signature as a byte sequence the bit b_{8*N} b_{8*N-1} ... b_{k+1} ,if existent, shall have the value 0. If this is not the case, the signature shall be rejected.

   The integer value resulting from the binary representation of the signature shall be less than n. If this is not the case, the signature shall be rejected.

2. The standard algorithm for plaintext recovery described in section 14.1.3.1 shall be applied to the signature. The result has to be represented as a byte sequence of N-1 byte length. If this is not the case, the signature shall be rejected.

3. A DSI' with a length of N-1 byte shall be generated from the message M' as described in steps 1. and 2. of section 14.1.4.1.1.

The DSI' shall be compared with the plaintext recovered in step 2. If the values match, the verification of the signature was successful. Otherwise the signature shall be rejected.

14.1.4.1.3 Notation

The following notation is used for the computation of a signature for the message M with the signature mechanism A005 and the private RSA key S_K:

$s = sign_{A005}(S_K)[M]$.

The following notation is used for the verification of a signature s for the message M with the signature mechanism A005 and the public RSA key P_K:

$verify_{A005}(P_K)[s,M]$.

14.1.4.2 Signature mechanism A006

For the computation and verification of a digital signature with the signature mechanism described in [PKCS1] using the encoding method EMSA-PSS, he following points have to be indicated:

- the hash algorithm HASH to be used,
- the byte length H of the generated hash values,

- the byte length S of the salt to be used,
- the mask generation function to used,
- the signature algorithm to be used,
- the maximal bit length k of the generated DSI to be allowed as input for the signature algorithm and
- the maximal byte length N of the generated DSI to be allowed as input for the signature algorithm.

The digital signature mechanism A006 is based on EMSA-PSS using the hash algorithm SHA-256. The byte length H of the hash value is 32.
The length S of the salt is defined by the used hash algorithm, i.e. the length S of the salt shall be the byte length H of the hash value.
For A006 only the mask generation function MGF1 as described in [PKCS1] will be used.
Notation: k is length of the modulus n (in bits) of the applied RSA key. The length of the DSI (in bits) is k – 1 and will be denoted as emBits. The length of the modulus n (in bytes) is denoted as N. The length of the DSI (in bytes) is denoted as emLen.

14.1.4.2.1 Mask generation function MGF1

The mechanism described in [PKCS1], sections 8.1 and 9.1 uses a mask generation function described in [PKCS1], section B.2.
MGF1 is a mask generation function based on a hash algorithm HASH, which calculates hash values with the byte length H. MGF1 creates a byte sequence of a given length **maskLen** from a given input value (seed) **mgfSeed** as described in the following:

1. Let T be an empty byte sequence.
2. For a counter from 0 to $\lceil maskLen / H \rceil - 1$, do the following:
   a. Convert the counter to a byte sequence C with the length of 4 bytes.
   b. Calculate the hash value HASH (mgfSeed | C) and concatenate this to the byte sequence T:

$$T = T \mid HASH(mgfSeed \mid C)$$

3. The result MGF1(mfgSeed, maskLen) will be the leftmost maskLen bytes of the byte sequence T.

Note that $\lceil maskLen / H \rceil$ defines the smallest integer larger than or equal to (maskLen / H).

14.1.4.2.2 Digital signature generation according to EMSA-PSS

According to [PKCS1] (using the method EMSA-PSS), sections 8.1.1 and 9.1.1 the following steps shall be performed for the computation of a signature for message M with bit length m.

1. The hash value HASH(M) of the byte length H shall be computed. If EMSA-PSS will be used as basis for the signature mechanism A006, the hash value SHA-256(M) with the length of 32 bytes will be calculated.

2. The input value DSI for the signature algorithm shall be generated as follows:

   Generate a random number of S bytes to be used as salt.

   Build the message M' as follows:

   M' = '00 00 00 00 00 00 00 00' | HASH(M) | salt

   Compute over M' the hash value HASH(M') of the byte length H.

   Build a padding string PS with a length of emLen – H – S – 2 bytes consisting of '00' bytes.

   Let DB = PS | '01' | salt; DB is a byte sequence of the length emLen – H – 1.

   Let dbMask = MGF(HASH(M'), emLen – H – 1) the result of the mask generation function. If EMSA-PSS is used as basis for A006, the function MGF1 as described in 14.1.4.2.1 will be used.

   Let maskedDB = DB $\oplus$ dbMask.

   Set the leftmost 8*emLen – emBits bits of the leftmost byte in maskedDB to zero.

   Let DSI = maskedDB | HASH(M') | 'BC'.

3. A signature shall be computed using the byte sequence DSI as input to the standard signing function described in 14.1.3.1.

   It has to be regarded, that the DSI is represented as a sequence of emLen byte. The integer value resulting from the binary representation of the DSI is always less than the value of the modulus n, since the bit length emBits of the DSI is less than the bit length of the modulus.

   The signature may be represented as a byte sequence with the byte length N. In the representation of the modulus n as a byte sequence the bit b_k has the value 1 and the bit b_{8*N} b_{8*N-1} ... b_{k+1} have, if present, the value 0. In the representation of the signature as a byte sequence the bit b_{8*N} b_{8*N-1} ... b_{k+1} therefore also shall have the value 0.

14.1.4.2.3 Digital signatur verification according to EMSA-PSS

According to [PKCS1] (using the method EMSA-PSS), sections 8.1.2 and 9.1.2, the following steps shall be performed for the verification of a signature. The signature to be verified and the message M must to be available as byte sequences.

1. The signature must be represented as a byte sequence with the byte length N. In the representation of the signature as a byte sequence the bit b_{8*N} b_{8*N-1} ... b_{k+1} ,if present, shall have the value 0. If this is not the case, the signature shall be rejected.

   The integer value resulting from the binary representation of the signature shall be less than n. If this is not the case, the signature shall be rejected.

2. The standard function for plaintext recovery shall be applied as described in 14.1.3.2 to the signature. The result has to be represented as a byte sequence of emLen byte length. If this is not the case, the signature shall be rejected.

3. The recovered plaintext shall be checked as follows:

   The hash value HASH(M) of the byte length H shall be computed.

   The least significant byte of the recovered plaintext shall have the value 'BC'. If this is not the case, the signature shall be rejected.

   Let maskedDB be the leftmost emLen – H – 1 bytes of the recovered plaintext and let HM' be the next H bytes of the recovered plaintext.

   If the leftmost 8*emLen – emBits bits of the most significant byte of maskedDB are not all equal to zero, the signature shall be rejected.

   Let dbMask = MGF (HM', emLen – H – 1), using the function MGF1.

   Let DB = maskedDB ⊕ dbMask.

   Set the leftmost 8*emLen – emBits bits of the leftmost byte in DB to zero.

   If the emLen – H – S – 2 leftmost bytes of DB are not all equal to '00' or if the byte at the position emLen – H – S – 1 does not have the value '01', the signature shall be rejected.

   Let salt be the rightmost S bytes of DB.

   Let

   M' = '00 00 00 00 00 00 00 00' | HASH(M) | salt

   and compute the hash value HASH(M') of the byte length H.

If HM' = HASH(M') the verification of the signature was successful. Otherwise the signature shall be rejected.

14.1.4.2.4 Notation for EMSA-PSS

The following notation is used for the computation of a signature for the message M with the signature mechanism according to [PKCS1] using EMSA-PSS and the private RSA key S_K:

$s = sign_{EMSA\text{-}PSS}(S_K)[M]$.

The following notation is used for the verification of a signature s for the message M with the signature mechanism according to [PKCS1] using EMSA-PSS and the public RSA key P_K:

$verify_{EMSA\text{-}PSS}(P_K)[s,M]$.

14.1.4.2.5 Digital signature generation according to A006

As already mentioned banking applications will also use the AUT-key for the generation of a signature, which was formerly intended only for authentication purposes. Using the command INTERNAL AUTHENTICATE with the AUT-key and the signature mechanism EMSA-PSS the SECCOS smart card will always calculate internally a hash value over the input data of the command. Since banking applications have to calculate signatures over messages which are usually quite long, these messages cannot be given directly as input data with the command INTERNAL AUTHENTICATE to the SECCOS smart card. For this reason the banking application will also calculate a hash value over the message. This hash value will be the input data of the command INTERNAL AUTHENTICATE. Hence, using the AUT-key and EMSA-PSS, the hash value will be calculated twice. For this reason the signature mechanism A006 will be defined as follows.
To calculate a signature s over a message M with the private key S_K using the signature mechanism A006 the following steps have to be performed:

- calculate the hash value HM = SHA-256(M).
- then calculate the signature $s = sign_{EMSA\text{-}PSS}(S_K)[HM]$.

14.1.4.2.6 Digital signature verification according to A006

To verify a signature s over a message M with the public key P_K using the signature mechanism A006 the following steps have to be performed:

- calculate the hash value HM = SHA-256(M).
- then verify the signature using $verify_{EMSA\text{-}PSS}(P_K)[s,HM]$.

14.1.4.2.7 Notation for A006

The following notation is used for the computation of a signature for the message M with the signature mechanism A006 and the private RSA key S_K:

$s = sign_{A006}(S_K)[M]$.

The following notation is used for the verification of a signature s for the message M with the signature mechanism A006 and the public RSA key P_K::

$verify_{A006}(P_K)[s,M]$.

14.1.5 References

[EMV CA] Europay International, MasterCard International and Visa International, Integrated Circuit Card Specifications for Payment Systems, Annexes, Version 3.1.1, 31.05.1998

[FIPS H2] FIPS 180-2, Secure Hash Signature Standard, Federal Information Processing Standards Publication 180-2, U. S. Department of Commerce / N.I.S.T., National Technical Information Service, August 2002

[PKCS1] PKCS #1: RSA Encryption Standard, Version 2.1, 14.06.2002

[RSA] R. L. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public key cryptosystems, Communications of the ACM, vol. 21, n. 2, 1978, 120-126

[SECCOS6] Interface Specifications for the SECCOS ICC, Secure Chip Card Operating System (SECCOS), Version 6.1, 19.05.2006 (with revisions as on October 16th, 2006)

[ZKASigAnw] Interface Specifications for the SECCOS ICC, Digital Signature Application for SECCOS 6, Version 1.1, 25.05.2007

14.1.6 XML structure of signature versions A005/A006

The following diagram illustrates the structure of the bank-technical electronic signature (ES) in structured form:

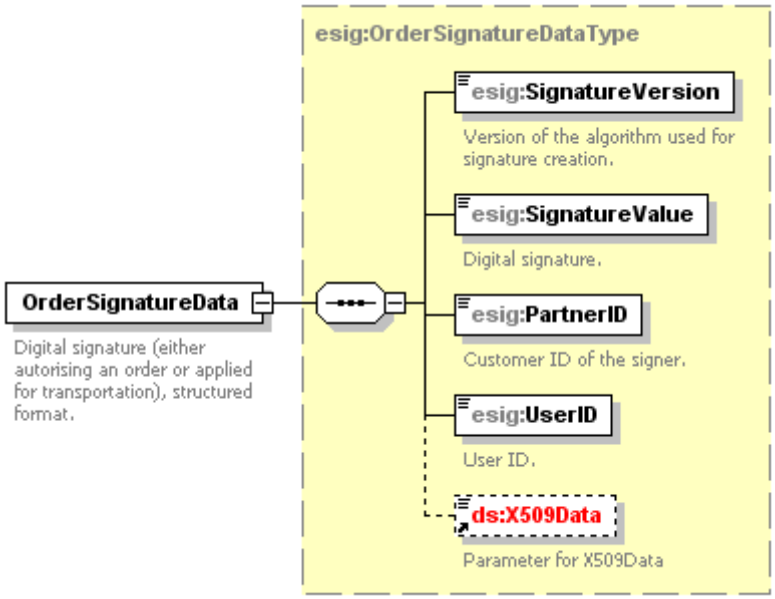


Diagram 100: OrderSignatureData – structured electronic signature

`OderSignatureData` may only be transmitted as part of an XML document with root element `UserSignatureData`. Detailed information and illustrations see chapter 3.5.3 .

With the intention to utilize the ES in structured form outside of EBICS, all necessary data structures have been defined in an independent XSD file (ebics_signature_S002.xsd) which can be downloaded from https://www.ebics.org/en/ebics-schema.
For the transport of the public signature key the format `SignaturePubKeyInfoType` is used (see chapter 4.2).

15 Appendix: Standards and references

The EBICS detailed concept refers to a number of processes, algorithms and format stipulations.

The associated standard document identifications and links to the referenced documents are listed in the following section.

Standard	Characteristics	Standard identification	Reference
EBICS	Multi-bank capable interface for Internet-based communication	H005	https://www.ebics.org/en/ebics-schema (XML-Schema)
ZIP	Universal compression algorithm	RFC 1950, RFC 1951	http://www.ietf.org/rfc/rfc1950.txt http://www.ietf.org/rfc/rfc1951.txt
base64	Coding format for textual byte code transport	RFC 1421, RFC 2045	http://www.ietf.org/rfc/rfc1421.txt http://www.ietf.org/rfc/rfc2045.txt
UTF-8	Coding format for Unicode characters	RFC 3629 (ISO 10646)	http://www.ietf.org/rfc/rfc3629.txt
HTTP 1.1	Internet application protocol	RFC 2616	http://www.ietf.org/rfc/rfc2616.txt
TLS	Transport layer encryption	RFC 2246, RFC 3268 (+AES), RFC 2818 (HTTP via TLS)	http://www.ietf.org/rfc/rfc2246.txt http://www.ietf.org/rfc/rfc3268.txt http://www.ietf.org/rfc/rfc2818.txt
TCP	Internet transmission protocol	RFC 793	http://www.ietf.org/rfc/rfc793.txt
IP(v4)	Internet network protocol	RFC 791	http://www.ietf.org/rfc/rfc791.txt
XML	Hierarchical documentation language	(W3C-Rec.)	http://www.w3.org/TR/REC-xml/
XML signature	Process for digital signature of XML documents	RFC 3275	http://www.ietf.org/rfc/rfc3275.txt http://www.w3.org/TR/xmldsig-core/

X.509v3	Format and profile for PKI certification data	RFC 5280	http://www.ietf.org/rfc/rfc5280.txt
Country codes	Format for country abbreviations	RFC 1766, ISO 639	http://www.ietf.org/rfc/rfc1766.txt
Time stamp	Format for date & time stamp	ISO 8601 (2004)	http://www.iso.org/iso/en/CatalogueDetail» Page.CatalogueDetail?CSNUMBER=40874
SHA-256	Hash algorithm	RFC 6234, FIPS 180-4 (SHA gen.)	https://tools.ietf.org/html/rfc6234 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
AES	Symmetrical encryption algorithm	FIPS 197	http://csrc.nist.gov/publications/fips/fips197/» fips-197.pdf

16 Appendix: Glossary

Administrative Order type	Three-figure alphanumeric code that identifies a type of administrative EBICS order.
AES	“Advanced Encryption Standard”: a symmetrical encryption algorithm that is intended to replace DES. In the EBICS context, AES is used for TLS as well as for the encryption of bank-technical order data (in accordance with RFC 3268).
Bank system	Components within the responsibility sphere of the financial institution that are involved in the implementation of an EBICS transaction. This includes both the bank-technical target system and the HTTP server(s) that receive the EBICS message and forward it to the bank-technical target system.
Bank-technical electronic signature	Subscriber’s ES of signature class “E”, “A” or “B”, via which the processing of an order is authorised.
Bank-technical key (public/private)	RSA key pair whose private key is used for configuring the bank-technical electronic signature and whose public key is used for its verification.
Bank-technical order data	Data that is required for the processing of an order. The format of this data depends on the BTF odentifier and/or administrative order type. The majority of the data formats that are used in EBICS have already been defined. The data formats of the administrative order types that have been newly defined for EBICS (such as e.g. Distributed Electronic Signature order types) are defined in EBICS by means of an XML schema. The order data of an order is transparently embedded (in compressed, encrypted form) in EBICS messages.
Bank-technical target system	Component within the responsibility sphere of the financial institution that is responsible for the administration of customers/subscribers and the processing of bank-technical orders. Within the framework of the EBICS specification, the bank-technical target system can be viewed as a “secure black box”.
base64	Coding algorithm and format in accordance with RFCs 1421 & 2045. The result of a base64 coding run can be completely represented in ASCII.
BTF identifier	Set of information which defines the kind of order (primarily defined by the XML structure “service”).
CA	Abbreviation for certificate authority
Certificate	If the term “certificate” is used in the EBICS specification documents the public key format x.509 is meant. Keys certified by a CA are called CA-issued certificates. In the EBICS documents the **hash values of keys** are understood as the composition of the **hash value of the certificate**.
Client	Communications unit that sends EBICS requests and receives EBICS

	responses. See also “Customer system”.
Control data	Data in an EBICS message that is required for controlling the flow of an EBICS transaction. This is data for authentication of the subscriber by the bank system, data for identification of the next transaction step that is to be carried out, or technical return codes, or order parameters, or data for preliminary verification or bank-technical return codes.
Customer	Organisational unit (company or private person) that concludes a contract with the financial institution. In this contract it will be agreed as to which business transactions the customer will conduct with the financial institution, which accounts are concerned, which of the customer’s subscribers work with the system and the authorisations that these subscribers will possess.
Customer system	Components that are used by subscribers to upload orders to the financial institution and to obtain information on orders or subscriber accounts from the financial institution.
Distributed bank-technical signature	See “Electronic Distributed Signature”.
Download transaction	EBICS transaction for transmission of a download order. The transaction phases of a download transaction are: transaction initialisation, data transfer, acknowledgement of the download data.
EBICS message	EBICS request from a subscriber or EBICS response from the financial institution. EBICS messages are mainly composed of control data, the identification and authentication signature and bank-technical data.
EBICS request	Request from a subscriber in XML format that has been defined in EBICS.
EBICS response	Response from the financial institution in XML format that has been defined in EBICS.
EBICS transaction	Sequential flow of EBICS transaction phases that are necessary to transmit an order to the bank-technical target system. EBICS transactions can be upload or download transactions.
EBICS transaction administration	Bank system component that is responsible for the administration of EBICS transactions.
EBICS transaction phase	Sequence of connected EBICS transaction steps. A differentiation is drawn in EBICS between the following transaction phases: Transaction initialisation (“initialisation”), data transfer (“transfer”) and acknowledgement (“receipt”) .
EBICS transaction step	Pair comprising an EBICS request and the associated EBICS response. An EBICS request is always initiated by the customer system.
EDS	See “Electronic Distributed Signature”.
Electronic Distributed Signature	A process where in bank-technical electronic signatures can be supplied for a particular order, irrespective of time or place. See Chapter 8 for details. The abbreviation is “EDS”. In HAC-Codes the previous abbreviation VEU remains.
Electronic	Voluntary signature of the order data in an upload order by a

signature (ES)	subscriber with which the corresponding order can be submitted or authorised, or also a financial institution signature for download data. In EBICS, ES's are used in accordance with the Appendix (Chapter 14) that are at least configured in accordance with process A005
Encryption key (public/private)	RSA key pair whose public key is used by the communications partners for encryption of the symmetrical transaction key and whose private key is used by owners for decrypting the same transaction key.
ES	See "Electronic signature".
ES signature key	See "Bank-technical key (public/private)".
Host ID	EBICS host ID for the identification of the EBICS bank computer system in every request message of the customer system..The financial institution communicates the EBICS host ID together with the URL for the bank access to the customer.
Identification and authentication key (public/private)	RSA key pair whose private key is used for configuring the identification and authentication signature and whose public key is used for its verification.
Identification and authentication signature	Digital signature to ensure the authenticity of the control data in an EBICS message. XML Signature is used as a signature format.
Key management	Component of the bank system that is responsible for the assignment of public keys to subscribers and that controls access to the keys it administrates.
Order	Bank-technical or system-related business transaction whose type is identified via BTF identifiers
Order data	See "Bank-technical order data"
Order parameters	Additional order parameters that the client transmits to the server in the first transaction step. See Chapter 3.11.
Order ID	Unambiguous order ID assigned by the bank server and submitted to the client system in the response of an upload transaction. It especially serves the synchronizing of order data and electronic signatures in a second upload. The application is to ensure the allocation of unambiguous order IDs per each customer ID and per BTF identifiers. Structure of a 4-digit order ID: 1st position: Alphabetic character (A–Z), selectable freely 2nd to 4th position: Alphanumerical characters (A–Z or 0–9) in ascending order
OrderData	See "Order type" and "BTF identifier".
Partner	See "Customer"
Segmentation	Division of the data volume of the order data after compression, encryption and base64-coding into segments with a size of max. 1 MB. See also Chapter 7.
Server	Communications unit that receives EBICS requests and sends EBICS responses. See also "Bank system".
Signature class	Relates to subscriber's ES's. EBICS defines the following signature classes: Individual signature (type "E"), First signature (type "A"), Second signature (type "B"), Transport signature (type "T"). See Chapter 3.5.1 for details.

Subscriber	Human users (“non-technical subscribers”) or a technical system (“technical subscriber”) that is/are assigned to a customer. Is identified by the combination of subscriber ID and customer ID. The technical subscriber serves for the data exchange between customer and financial institution. It must not be put on the same level as a technical ID for a service provider.
Subscriber initialisation	A process according to which the public subscriber keys are transmitted to the financial institution and are then activated by the financial institution. After successful execution of subscriber initialisation, subscriber are set in the bank system to the state “Ready”.
TLS	“Transport Layer Security”: Protocol in accordance with RFCs 2246 & 3268 for the cryptographic security of messages that use TCP/IP as a transmission protocol. In the EBICS context, TLS is used for the transport encryption of HTTP messages (HTTPS).
Transaction	See “EBICS transaction”.
Transaction key	Symmetrical key that is used within the EBICS transaction for the encryption of bank-technical data.
Transaction management	See “EBICS transaction management”.
Transaction phase	See “EBICS transaction phase”.
Transaction step	See “EBICS transaction step”.
Transport signature	Subscriber’s ES of signature class “T” via which the order is submitted (but its processing is not authorised).
Trust anchor	In the context of certification verification, a trust anchor (point of trust) is a certificate that is considered trustworthy. This is usually a certificate from a CA (Certification Authority).
Upload transaction	EBICS transaction for transmission of an upload order. The transaction phases of an upload transaction are: Transaction initialisation, data transfer.
UTF-8	“Unicode Transformation Format“, a character encoding standard according to RFC 3629.
ZIP	Loss-free compression algorithm according to RFCs 1950 and 1951.

17 Table of diagrams