Z E N T R A L E R K R E D I T A U S S C H U S S

MITGLIEDER: BUNDESVERBAND DER DEUTSCHEN VOLKSBANKEN UND RAIFFEISENBANKEN E.V. BERLIN • BUNDESVERBAND DEUTSCHER BANKEN E. V. BERLIN • BUNDESVERBAND ÖFFENTLICHER BANKEN DEUTSCHLANDS E. V. BERLIN • DEUTSCHER SPARKASSEN- UND GIROVERBAND E. V. BERLIN-BONN • VERBAND DEUTSCHER HYPOTHEKENBANKEN E. V. BERLIN

Anlage 1

der Schnittstellenspezifikation für die Datenfernübertragung zwischen Kunde und Kreditinstitut gemäß DFÜ-Abkommen

„Spezifikation für die EBICS-Anbindung“

Version 2.5

Final Version, 16. Mai 2011

Diese Spezifikation ist ab dem 1. Juli 2012 gültig.

Hinweis zur deutschen Fassung der EBICS-Spezifikation:
Diese Übersetzung wurde mit der größtmöglichen Sorgfalt erstellt.
Im Zweifelsfall ist die von der EBICS SCRL herausgegebene englischsprachige Fassung unter www.ebics.org (Rubrik „Specification“) maßgeblich.

Änderungsverfolgung

Die Tabelle enthält eine Übersicht der wesentlichen Änderungen von Version 2.4.2 auf Version 2.5.

Kapitel	Art*	Beschreibung	Inkrafttreten
3.1.4 und verschiedene XML-Beispiele im Dokument	Ä	Bis zur Schemaversion H003 (d.h. bis zur EBICS-Version 2.4.x) war der target namespace eine http-Adresse. Üblicherweise werden xsd's in lokalen Verzeichissen abgelegt, daher ist es nur notwendig, das Schema für einen initialen Download auf einer Webseite zur Verfügung zu stellen (www.ebics.org mit deutschsprachigen Erläuterungen auch unter www.ebics.de). Ab Schemaversion H004 (EBICS 2.5) ist der target namespace als urn-Adresse definiert ("urn:org:ebics:H004") um zu betonen, dass es sich um einen symbolischen Namen handelt. Die xsd-Datein wurden umbenannt, sie erhalten den Namenszusatz _H004 (daher mussten auch die include-Definitionen aktualisiert werden). Hinweise: es erfolgte keine Umbenenungen bzw. Änderungen für die Schemadateien signature.xsd (S001) und hev.xsd (H000)	2.5
Verschiedene Abschnitte; insbesondere in Kapitel 4, 5, 8 und 9	Ä	Um eine eindeutige Auftragsnummer sicherzustellen wird diese (OrderID) pro Auftrag ab EBICS-Version 2.5 durch den Bankserver zugewiesen (an Stelle der bisherigen kundensystemseitigen Nummernvergabe). Die Regeln zur Prüfung von Auftragsnummern und Auftragsattributen wurde daher auch überarbeitet (Kapitel 5.5.1.2.1, I.e.c)	2.5
Verschiedene Abschnitte; insbesondere in Kapitel 4.4.1.1; 4.4.1.3 (neues Kapitel)	E	Spezifikation einer neuen Aufragsart H3K: Alle öffentlichen Schlüssel können in einem Schritt gesendet werden (H3K-Request), für den EU-Schlüssel ist ein Zertifikat zu verwenden, das von einer Zertifizierungsinstanz (CA) ausgegeben wurde. INI- und HIA-Brief sind in diesem Fall nicht notwendig.	2.5
4.4.2.1	K	Ergänzung eines Hinweises, dass der Kunde die Gültigkeit des Zertifikates, das er beabsichtigt zu verwenden, zu überprüfen hat.	2.5
8.3.1	E	Integration des optionales Attributs "isCredit" in die HVZ-Response (analog zu HVT)	2.5
8, 9.3	E	Die HKD- und HTD-Response enthält die Auftragsart. Im Falle von FUL/FDL wird nun zusätzlich eine Information über das Dateiformat aufgenommen. Aus Konsistenzgründen wird das optionale Element "FileFormat" auch in die Elementgruppen HV*OrderParams (für * = D, T; E und S) und HV*ResponseOrderData (für * = U und Z) integriert.	2.5

* F = Fehler; Ä = Änderung; K = Klarstellung; E = Erweiterung; L = Löschung

Kapitel	Art*	Beschreibung	Inkrafttreten
10, 13	Ä	Die Spezifikation des Kundenprotokolls PTK, das nur in Deutschland verwendet wird, wird durch die Spezifikation eines XML-basierten Kundenprotokolls (HAC) ersetzt.	2.5
Anhang 1	E / Ä	Definition eines neuen Return Codes EBICS_CERTIFICATES_VALIDATION_ERROR (Nummer 091219) für den H3K-Request Aktualisierung von verschiedenen Return Code-Beschreibungen (bedingt durch die nun serverseitige Vergabe der Auftragsnummer)	2.5
Schemadateien innerhalb von H004		1. Der Target namespace für H004 wird als urn-Adresse definiert: urn:org:ebics:H004 2. Neue Datei- (xsd-) Namen für alle Schemadateien innerhalb von H004 (z.B. ebics_request_H004) 3. HVZResponseOrderData: Definition eines optionalen Attributs “isCredit” für TotalAmount 4. UserPermissionType (genutzt in HKD und HTD): Definition eines optionalen elements “FileFormat” 5. Definition einer Elementgruppe H3KResponseOrderData (in ebics_orders_H004.xsd) und verschiedener neuer Datentypen “…CertificateInfoType” (in ebics_types_H004.xsd) 6. Integration des Elements “OrderID” in ebics_keymgmt_response_H004.xsd (als Pflichtelement definiert) und ebics_response (als optional definiert) 7. Definition von bisher optionalen Attributen (mit Default) als verpflichtend, die in der Kanonisierung (X002) einbezogen werden: a. In ebicsRequest/header/mutable und ebicsResponse/header/mutable: 1+2) /SegmentNumber/lastSegment b. In ebicsRequest/header/static/OrderDetails/HVTOrderParams 3)/OrderFlags/completeOrderData 4)/OrderFlags/fetchLimit 5)/OrderFlags/fetchOffset c. ebicsRequest/header/static/OrderDetails/*OrderParams 6)/Parameter/Value/Type 8. Aus Konsistenzgründen erhalten die folgenden Strukturen abenfalls das optionale Element “FileFormat” HV*OrderParams (für * = D, T; E und S) HV*ResponseOrderData (für * = U und Z)	2.5 = Schema H004

Inhalt

Die XML-Schemata (H004, S001, H000) finden sich unter
http://www.ebics.org (siehe „Specification“) oder mit deutschsprachigen Erläuterungen unter
http://www.ebics.de (siehe „Spezifikation“)

1 Übersicht und Zielsetzungen für EBICS

1.1 Zielsetzung der Kooperation

Die deutsche Kreditwirtschaft, vertreten durch den Zentralen Kreditausschuss (ZKA), und die französische Kreditwirtschaft, vertreten durch das Comité Français d'Organisation et de Normalisation Bancaires (CFONB), gründeten eine Gesellschaft (EBICS SCRL) zur gemeinsamen Nutzung von EBICS.
EBICS wurde ursprünglich von der deutschen Kreditwirtschaft entwickelt. Es ermöglicht Firmenkunden eine flexible, sichere und effektive Durchführung ihrer Bankgeschäfte und erlaubt ihnen, den für ihre individuellen Bedürfnisse geeigneten Dienstleister auszuwählen. Außerdem ist EBICS „multibankfähig". Dies bedeutet, dass in Zukunft Firmenkunden in beiden Ländern mit Hilfe derselben Software mit jeder beliebigen Bank in Deutschland und Frankreich Verbindung aufnehmen können.
Prinzipiell ist die Spezifikation für beide Länder gleichermaßen gültig. In Ausnahmefällen jedoch kann eine für ein Land geltende Vorschrift eine individuelle Anpassung der Spezifikation erforderlich machen.
Als optional definierte Funktionalitäten können in dem einen Land unterstützt (und verpflichtend vorgeschrieben) und gleichzeitig im anderen Land nicht unterstützt werden.
Ein gemeinsamer Implementation Guide, der die die Unterstützung von optionalen Funktionalitäten näher erläutert, wurde erstellt (siehe Kapitel 3 in diesem Guide).

1.2 Grundsätzliche Zielsetzung von EBICS

Die vorliegende EBICS-Feinspezifikation („Electronic Banking Internet Communication Standard") erweitert das bestehende DFÜ-Abkommen vom 15.03.1995 um die Funktionalität multibankfähiger, sicherer Kommunikation über das Internet und bildet unter dem Titel „Spezifikation für die EBICS-Anbindung" nunmehr die Anlage 1 des DFÜ-Abkommens. Das im Rahmen des DFÜ-Abkommens bislang allein gültige FTAM-Verfahren wird nach der Praxiseinführung von EBICS noch bis zum 31.12.2010 parallel zu EBICS unterstützt und bildet in der derzeit aktuellen Version 2.0 vom 03.11.2005 unter dem Titel „Spezifikation für die FTAM-Anbindung" die Anlage 2 des DFÜ-Abkommens. Die für alle Kommunikationsverfahren im Rahmen des DFÜ-Abkommens verbindlichen bankfachlichen Datenformate bilden unter dem Titel „Spezifikation der Datenformate" die Anlage 3 des DFÜ-Abkommens. Die jeweils aktuelle Version der Anlage 3 sowie die o.g. Anlage 2 sind unter www.ebics.de („Spezifikationen") downloadbar.

Die anwendungsbezogenen Elemente des FTAM-Verfahrens, d.h. die Multibankfähigkeit, die Übertragung von Dateien in bankfachlichen Formaten unter Nutzung von Auftragsarten sowie die definierten Sicherheitsverfahren für die Elektronische Unterschrift (EU) bleiben dabei für EBICS vollständig erhalten. Es werden die im Anhang (Kapitel 13) und im Dokument „EBICS Anhang 2 Auftragsartenkennungen" definierten Auftragsarten unterstützt. Die in EBICS gegenüber dem FTAM-Verfahren nicht mehr unterstützten Auftragsarten sind in Kapitel 3.10 aufgeführt.

Die Elektronische Unterschrift wird ab Version 2.4 in der Version A004 und höher unterstützt.

EBICS stellt keine speziellen Anforderungen an die konkrete Architektur der Kundensysteme; es können eigenständige Desktop-Applikationen genauso angebunden werden wie etwa Client/Server-Anwendungen oder Applet-Lösungen.

Das Verfahren „DFÜ mit Kunden“ wird auf der Anwendungsebene ergänzt um das Konzept der Verteilten Elektronischen Unterschrift (VEU), das eine zeitlich und räumlich unabhängige Autorisierung von Aufträgen kundenübergreifend ermöglicht.

Die grundsätzlichen Merkmale des EBICS-Standards sind:

- Übertragung der fachlichen Daten (Geschäftsvorfälle) über Auftragsarten in den etablierten bankfachlichen Formaten
- Erweiterung des DFÜ-Abkommens um die Möglichkeit der „Verteilten Elektronischen Unterschrift (VEU)“
- Spezifikation der EBICS-spezifischen Protokollelemente in XML
- Versand der Nachrichten über HTTP („internetbasiert“); Nutzung von TLS als Basis-Transportsicherung zwischen dem Kunden- und dem Banksystem unter Verwendung der TLS-Server-Authentisierung
- kryptographische Absicherung jedes einzelnen Schritts einer Transaktion mittels Verschlüsselung und digitaler Signaturen auf Anwendungsebene.

Die EBICS-Feinspezifikation ist die Grundlage für die Entwicklung von Kunden-und Banksystemen, die über das EBICS-Protokoll kommunizieren. Als solche enthält sie herstellerunabhängige Verfahrensbeschreibungen und gewährleistet so die Interaktion zwischen Kunden– und Banksystemen unterschiedlicher Hersteller.

Diese Feinspezifikation umfasst die EBICS-Protokollbeschreibung samt Details zu Schlüsselmanagement, VEU und den XML-Schemata für die Auftragsdaten der neuen EBICS-Auftragsarten. Die kompletten XML-Schemata sind als eigenständige HTML-Dokumentation hinterlegt.

Die Feinspezifikation schränkt die Implementierungsfreiheit der Kunden- und Banksysteme durch Vorgaben und Festlegungen nur da ein, wo Sicherheitsüberlegungen oder Abläufe jenseits der EBICS-Kommunikation dies erforderlich machen. In Abgrenzung zum EBICS Implementation Guide werden in der Feinspezifikation keine Implementierungsalternativen aufgezeigt

Es ist zu beachten, dass alle Beschreibungen, die FTAM betreffen, für die französische Implementation nicht gelten.

2 Definitionen

2.1 Begriffe

Folgende Begriffe in Kapitälchenschrift haben in der Protokolldefinition eine spezielle Bedeutung:

- **MUSS/MÜSSEN:** beschreibt eine zwingende Anforderung; nur Implementierungen, die diese Anforderung erfüllen, sind EBICS-konform
- **SOLL/SOLLEN/SOLLTE/SOLLTEN:** benennt Anforderungen, die normalerweise zu befolgen sind; technisch oder fachlich begründete Ausnahmen sind im Einzelfall aber möglich
- **KANN/KÖNNEN:** bezeichnet unverbindliche Empfehlungen oder optionale Features.

Funktionalitäten oder Merkmale des EBICS-Protokolls werden als **optional** gekennzeichnet, wenn sie vom Kreditinstitut nicht unterstützt werden müssen. Kunden haben gegenüber den Kreditinstituten keinen rechtlichen Anspruch auf die entsprechende Funktionalität.

Funktionalitäten oder Merkmale des EBICS-Protokolls werden in einer bestimmten Version als **vorgesehen** gekennzeichnet, wenn sie für nachfolgende Versionen vorbereitet sind, jedoch in der gegebenen Version noch nicht verwendet werden dürfen.

2.2 Notationen

2.2.1 XML

2.2.1.1 XML-Schema

Folgende Symbolik zur grafischen Darstellung von XML-Schema wird verwendet:

- Elemente werden in Rechtecke gesetzt
- Attribute werden ebenfalls in Rechtecke gesetzt und von einem Kasten „attributes“ umgeben
- Elemente, Attribute und weitere Deklarationen, die zu einem komplexen Typ gehören, werden von einem gestrichelten, gelb hinterlegten Kasten umgeben
- Eine „Verzweigung” (entspricht `choice` bei XML-Schema) wird mit einem Achteck dargestellt, das ein Schaltsymbol für drei mögliche Schaltpositionen enthält. Rechts vom Symbol verzweigen die Verbindungslinien zu den möglichen Alternativen
- Eine „Sequenz“ (entspricht `sequence` bei XML-Schema) wird mit einem Achteck dargestellt, das ein Liniensymbol mit drei Punkten darauf enthält. Rechts vom Symbol verzweigen die Verbindungslinien zu den einzelnen Sequenzelementen

- Symbole mit durchgehender Umrandung deuten auf obligatorische Verwendung hin und entsprechen bei XML-Schema dem Attribut `minOccurs="1"` für Elemente bzw. `use="required"` für Attribute
- gestrichelte Symbole deuten auf optionale Verwendung hin und entsprechen bei XML-Schema dem Attribut `minOccurs="0"` für Elemente bzw. `use="optional"` für Attribute
- durchgestrichene Symbole deuten auf vorgesehene Verwendung hin und entsprechen bei XML-Schema der Attributkombination `minOccurs="0" maxOccurs="0"` für Elemente bzw. `use="prohibited"` für Attribute
- „m..n“ an der rechten unteren Ecke eines Elementsymbols begrenzt die Verwendung des Elements auf m- bis n-faches Vorkommen und entspricht `minOccurs="`*m*`" maxOccurs="`*n*`"` bei XML-Schema; bei „m..∞“ entsprechend `minOccurs="`*m*`" maxOccurs="unbounded"`
- Elementgruppen werden durch Achtecke dargestellt und entsprechen der `group`-Deklaration in XML-Schema
- Attributgruppen werden von Kästen mit dem jeweiligen Gruppennamen umgeben und entsprechen der `attributeGroup`-Deklaration in XML-Schema.

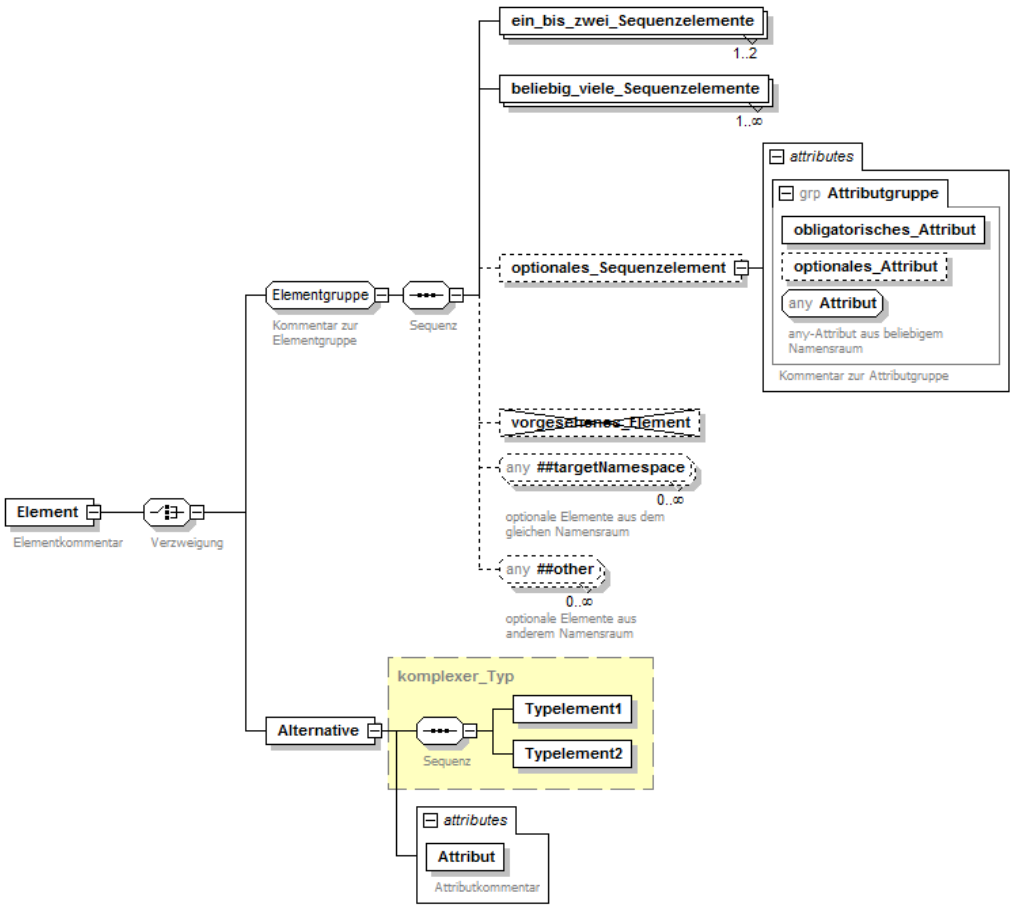


Abbildung 1: Symbolik für XML-Schema

2.2.1.2 XML-Dokumente

Einzelne Codeteile werden mit der Schriftart Courier versehen. Falls ein Elementname oder –typ nicht komplett in eine Zeile passt, wird mit »
in die nächste Zeile weitergeleitet.

```
Komplette Codebeispiele werden in Courier 8pt und mit Rahmen gesetzt.
```

2.2.2 Ablaufdiagramme

Abläufe werden mit Hilfe von UML 2.0-Aktivitäten dargestellt. Im vorliegenden Dokument enthalten sie einen Start- und einen Endknoten. Ein Startknoten ist der Startpunkt eines Ablaufs, der Endknoten markiert das Ende eines gesamten Ablaufs.

Für die bessere Übersichtlichkeit werden Aktivitäten ineinander verschachtelt. Aktionen, die wiederum eine Aktivität enthalten, werden durch ein Gabelsymbol gekennzeichnet. Die Aktivität A in Abbildung 2 besteht aus drei Ablaufschritten (Aktionen). Schritt_2 ist selbst wiederum eine Aktivität aus 2 Ablaufschritten. Die Aktivität Schritt_2 wird somit innerhalb der Aktivität A aufgerufen, d.h vom Startknoten von Schritt_2 bis zum Endknoten von Schritt_2 durchlaufen.

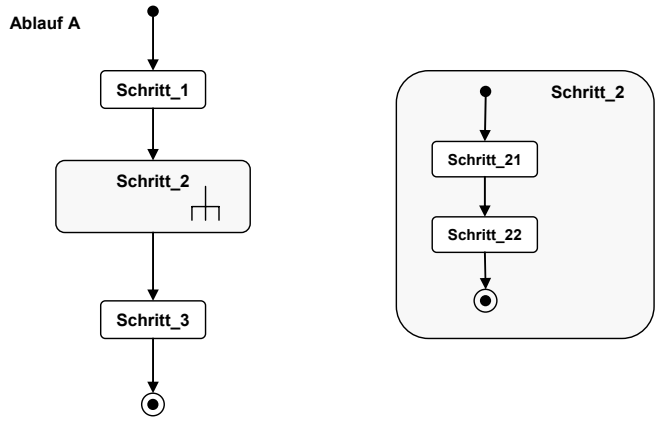


Abbildung 2 Verschachtelung von Aktivitäten

2.2.3 Sonstige Notationen

Bei Nennung neuer Auftragsarten wird mit dem Anhängsel „[verpflichtend]“ kenntlich gemacht, dass das Kreditinstitut diese Auftragsart unterstützen MUSS. Mit dem Anhängsel „[optional]“ ist demgegenüber gemeint, dass das Kreditinstitut diese Auftragsart unterstützen KANN.

Analog dazu wird vorgesehenen Merkmalen bzw. Funktionalitäten das Kennzeichen „[vorgesehen]“ angehängt.

2.3 Datentypen

XML-Schema definiert einen Satz von primitiven und abgeleiteten Datentypen, die zum Aufbau eigener Datentypen benutzt werden können.

Im Zusammenhang mit EBICS finden hauptsächlich die folgenden primitiven Datentypen Verwendung:

- **string**: Zeichenketten unbeschränkter Länge und beliebigen Aufbaus
- **boolean**: boolscher Wahrheitswert mit den Ausprägungen „true“ (=1) oder „false“ (=0)

- **decimal**: Dezimalzahlen beliebiger Genauigkeit
- **dateTime**: Zeitstempel mit Datum und Uhrzeit gemäß ISO 8601
  Der Aufbau ist wie folgt: YYYY-MM-DDTHH:MM:SS.sss**Z,**
  wobei Z wiedergibt, dass ein Umrechnung auf UTC stattgefunden hat.
  Sollte der Datumsstring nicht diesem Aufbau entsprechen ist die EBICS-Nachricht abzulehnen bzw. die EU-Prüfung als negativ zu bewerten.
- **date**: Datumsangabe gemäß ISO 8601
- **hexBinary**: Hexadezimalwert beliebiger Länge
- **base64Binary**: Datentyp zur Aufnahme von base64-kodierten Binärdaten
- **anyURI**: Uniform Resource Locator (z.B. URL, IP-Adresse).

Die folgenden vordefinierten Datentypen werden von primitiven Datentypen abgeleitet und finden im EBICS-Standard Verwendung:

- **normalizedString**: Zeichenkette, die am Anfang und Ende von Leerraum (blanks) bereinigt ist
- **token**: ein normalizedString, der keine Zeilenvorschübe und keine mehrfachen Leerräume hintereinander enthält
- **language**: Länderkennzeichen gemäß RFC 1766
- **nonNegativeInteger**: ganzzahlige, nichtnegative Werte
- **positiveInteger**: ganzzahlige, positive Werte.

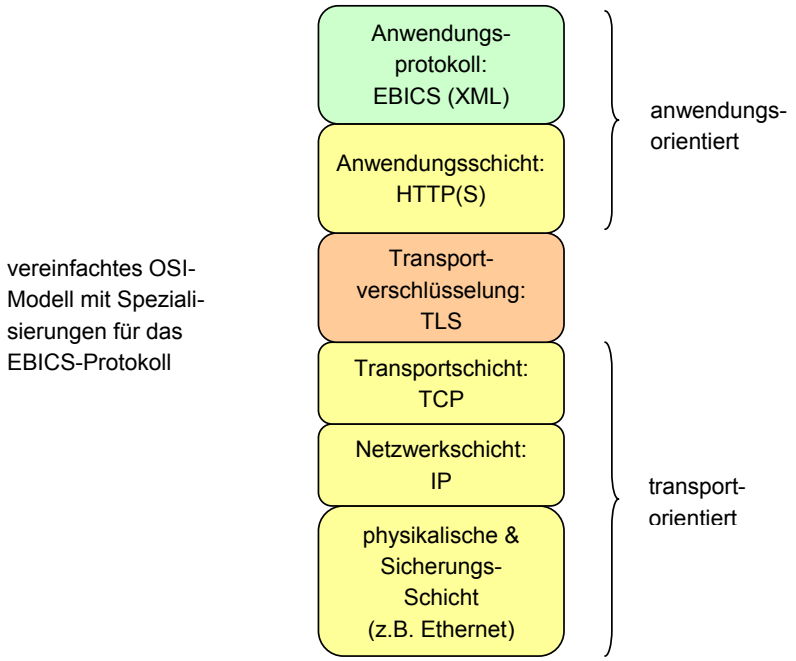
Mit Hilfe der genannten Datentypen werden in den EBICS-Schemata neue Datentypen definiert:

- **einfache (simple) Datentypen** definieren lediglich einschränkende oder erweiternde Merkmale bezüglich des Wertebereichs eines vorhandenen primitiven oder abgeleiteten Datentyps, d.h. sie leiten von einem bestehenden Datentyp ab
- **komplexe (complex) Datentypen** definieren neue Strukturen, die sich aus Feldern und Attributen unterschiedlicher (einfacher oder komplexer) Datentypen zusammensetzen.

3 Designentscheidungen

Dieses Kapitel verdeutlicht Entscheidungen, die ausschlaggebenden Einfluss auf das Design des EBICS-Protokolls hatten. Darunter fallen sowohl netzwerkspezifische Details als auch Vorgaben fachlicher und technischer Natur.

3.1 OSI-Modell aus EBICS-Sicht



3.1.1 TCP/IP als paketorientierte Übertragungsschicht

Als Transportprotokoll wird TCP/IP verwendet. Mit IP (Internet Protocol) werden die auszutauschenden Daten als Pakete übertragen. Diese Paketübertragung wird durch TCP (Transmission Control Protocol) als Übertragungskontrollprotokoll überwacht.

Der Kommunikationsaufbau erfolgt unter Verwendung einer URL (Uniform Resource Locator). Alternativ kann auch eine IP-Adresse des jeweiligen Kreditinstituts benutzt werden.Die URL oder die IP-Adresse ist zusammen mit der EBICS Host-ID für den Verbindungsaufbau zum Bankrechner erforderlich und wird dem Kunden bei Vertragsabschluss von dem Kreditinstitut mitgeteilt.

3.1.2 TLS als Transportverschlüsselung

TLS wurde von der Arbeitsgruppe Transport Layer Security WG im IETF (Security-Area) entwickelt. Es stellt einen ursprünglich von Netscape (zunächst unter dem Namen SSL) entwickelten, offenen Standard zur gesicherten Übertragung paketorientierter Daten dar. Das Ziel von TLS ist es, die Datensicherheit auf Schichten oberhalb von TCP/IP zu gewährleisten. Das Protokoll ermöglicht Datenverschlüsselung, Echtheitsbestätigung von Servern und Nachrichtenintegrität für TCP/IP-Kommunikation.

Es vereinigt folgende Grundeigenschaften:

1. Die TLS-Verbindung ist vertraulich: Beim TLS-Handshake wird unter Verwendung asymmetrischer Verschlüsselung (bei EBICS: RSA) ein gemeinsamer, geheimer Schlüssel vereinbart, der in der weiteren TLS-Sitzung als symmetrischer Schlüssel (bei EBICS: AES oder 3DES) dient.
2. Die Integrität der TLS-Verbindung ist gesichert: Der Nachrichtentransport beinhaltet eine Nachrichtenintegritätsüberprüfung mittels sogenannter „Message Authentication Codes“ (MACs). Sichere Hash-Funktionen (bei EBICS: SHA-1) kommen für die MAC-Berechnungen zum Einsatz.
3. Die Identität des Kreditinstituts wird durch die Verwendung von Serverzertifikaten und elektronischer Signaturen beglaubigt; mittels dieser TLS-Serverauthentisierung sind die Nachrichten des Kreditinstituts authentisiert.
4. TLS enthält Schutzmechanismen gegen Man-in-the-middle-Attacken auf der TLS-Verbindungsstrecke zwischen Kunden- und Banksystem. Dazu nutzt es interne Zähler und „verteilte Geheimnisse“ („shared secrets“) und sichert zudem den Handshake mit signierten Zusammenfassungen der bis dahin ausgetauschten Daten gegen einen solchen Angriff ab.

Für die Übertragung der EBICS-Nachrichten zwischen Kundensystem und Banksystem wird eine TLS-Verbindung zwischen beiden Systemen aufgebaut.

Eingesetzt wird TLS 1.0 mit X.509v3-Serverzertifikaten, d.h. der Server MUSS sich mittels Zertifikat authentisieren. Die Art des Zertifikats MUSS für den Schlüsseltauschalgorithmus der gewählten Schlüssel geeignet sein.

Zugunsten einer besseren Marktakzeptanz verzichtet EBICS in der Version H004 auf die TLS-Client-Authentisierung. Eine spätere Erweiterung um die Fähigkeit zur TLS-Client-Authentisierung (und der damit verbundenen Ausgabe von X.509v3-Clientzertifikaten für TLS an Kundensysteme) wird nicht ausgeschlossen.

Die Begriffe Client und Server werden in den nächsten beiden Unterkapiteln synonym zu TLS-Client bzw. TLS-Server verwendet. Das Kundensystem übernimmt dabei die Rolle des TLS-Clients, das Banksystem die des TLS-Servers.

3.1.2.1 Vorabverteilung und Prüfung der Vertrauensanker

Die Herausgeber der TLS-Serverzertifikate können sowohl öffentliche als auch nicht-öffentliche (z.B. bankinternen) CAs sein. Es liegt in der Verantwortung des Kreditinstituts, eine öffentliche CA als Herausgeber ihres TLS-Serverzertifikats nur dann zuzulassen, wenn diese CA eine hinreichende Prüfung der Identität der Zertifikatsinhaber garantiert.

Es ist Aufgabe des Kreditinstituts, Teilnehmern, die mit dem Kreditinstitut über EBICS kommunizieren möchten, ein vertrauenswürdiges CA-Zertifikat vom Zertifizierungpfad des TLS-Serverzertifikats zukommen zu lassen. Dafür gibt es die beiden folgenden Möglichkeiten:

- Auslieferung des CA-Zertifikats an den Kunden/ Teilnehmer:

  Das gilt sowohl für öffentliche wie für nichtöffentliche CAs.

- Auslieferung eines CA-Zertifikats (Bridge-CA), mit dessen Hilfe eine Liste von vertrauenswürdigen nichtöffentlichen CA-Zertifikaten unterzeichnet wurde:

  Das CA-Zertifikat des Kreditinstituts ist Bestandteil dieser Liste. Der Vorteil einer solchen signierten Liste kommt für den Teilnehmer dann zum Tragen, wenn die Liste die CA-Zertifikate mehrerer Kreditinstitute enthält, welchen der Teilnehmer bankfachliche Aufträge über EBICS erteilen möchte.

Die Auslieferung der CA-Zertifikate an den Kunden/ Teilnehmer SOLL auf elektronischem Wege passieren. Denkbar sind Auslieferungen über e-Mail oder Auslieferungen als Bestandteil der Kundensoftware bzw. als Updates der Kundensoftware.

Zusätzlich zum CA-Zertifikat KANN das Kreditinstitut auch das TLS-Serverzertifikat selbst an den Kunden/ Teilnehmer ausliefern. Während der Serverauthentisierung beim TLS-Verbindungsaufbau kann in diesem Fall das TLS-Serverzertifikat als Vertrauensanker verwendet werden.

Neben der Auslieferung von Zertifikaten MUSS das Kreditinstitut sicherstellen, dass der Teilnehmer die erhaltenen Zertifikate über einen zweiten, unabhängigen Kommunikationsweg überprüfen kann. Das kann beispielsweise über die Veröffentlichung dieser Zertifikate und ihrer Fingerprints im Internet erfolgen.

Im Gegenzug ist der Teilnehmer dafür verantwortlich, den Abgleich der über unterschiedliche Kommunikationswege eingegangenen Zertifikate vorzunehmen.

3.1.2.2 Serverauthentisierung

Im Rahmen des TLS-Handshakes überträgt der Server sein Zertifikat an den Client. Voraussetzung für den Aufbau einer TLS-Verbindung zwischen Client und Server ist die erfolgreiche Prüfung des Serverzertifikats durch den Client.

Dabei handelt es sich um ein X.509v3-Zertifikat. Die Überprüfung von X.509v3-Zertifikaten im Allgemeinen ist in RFC 3280 definiert, die Besonderheiten der Überprüfung der TLS-Serverzertifikate im Rahmen des HTTPS-Handshakes sind in RFC 2818 beschrieben.

Der Teilnehmer ist dafür verantwortlich, Kundensoftware einzusetzen, die die Überprüfung des TLS-Serverzertifikats gemäß dieser Spezifikationen durchführt. Ebenso ist er dafür verantwortlich, Kundensoftware einzusetzen, die das vom Kreditinstitut vorab erhaltene CA-Zertifikat bzw. TLS-Serverzertifikat bei dieser Prüfung als Vertrauensanker einzusetzt.

3.1.3 HTTP(S) als technisches Basisprotokoll

Das Hypertext Transfer Protocol (HTTP) ist ein zustandsloses Datenaustausch-Protokoll zur Übertragung von Daten. HTTP wird im „World Wide Web“ (WWW) hauptsächlich zur Übertragung von Webseiten verwendet.

Die Kombination von HTTP und TLS als Transportverschlüsselung wird auch als „HTTPS“ (HTTP Secure) bezeichnet. Hierfür ist der Port 443 (SSL) reserviert, welcher bei den meisten Firewall-Konfigurationen uneingeschränkt nutzbar ist.

Die Zustandslosigkeit von HTTP erzwingt für das EBICS-Protokoll die Verwendung eigener Sitzungsparameter, die mehrere Kommunikationsschritte logisch zu einer Transaktion zusammenfassen.

Die Kommunikation zwischen Kunde und Kreditinstitut erfolgt klassisch via Client/Server-Rollen. Das Kreditinstitut übernimmt wie bisher auch die (passive) Serverrolle und der Kunde die (aktive) Clientrolle. Bei diesem Kommunikationsschema sendet der Client mittels HTTP-Request eine Anfrage an den Server; dieser antwortet mit einer HTTP-Response. Der Request kann generell als GET- (zusätzliche Daten werden in die URL kodiert) oder POST-Anfrage (zusätzliche Daten werden an den HTTP-Header angehängt) gestellt werden; im EBICS-Kontext findet ausschließlich POST Verwendung.

Bei EBICS MUSS sowohl vom Client als auch vom Server HTTP 1.1 eingesetzt werden.

3.1.4 XML als Anwendungsprotokollsprache

Das Anwendungsprotokoll EBICS setzt auf dem technischen Basisprotokoll HTTP(S) auf. Als Protokollsprache auf Anwendungsebene wurde XML (Extensible Markup Language) gewählt. Für diese Entscheidung sprechen folgende Gründe:

1. XML verwendet lesbare Marken („tags“). Deren Namen/Attribute können derart gewählt werden, dass sich die Bedeutung der „tags“ auch ohne Dokumentation erschließen lässt.
2. Kostenlose XML-Parser sind für die gängigen Betriebssysteme und Programmiersprachen erhältlich.

3. XML-Nachrichten können einfach um zusätzliche Elemente und Attribute erweitert werden. Hierbei ist es für die Erhaltung der syntaktischen Korrektheit („well-formedness“) der Gesamtnachricht nicht erforderlich, Anpassungen an den bestehenden Nachrichtenabschnitten vorzunehmen.

4. Für die Validierung von XML-Nachrichten steht XML-Schema als standardisierte Definitionssprache zur Verfügung.

Als Zeichenkodierung innerhalb der EBICS-XML-Nachricht MUSS UTF-8 verwendet werden. UTF-8 wird von jedem XML-Parser unterstützt und kodiert abwärtskompatibel zu ASCII.

Die Syntax der XML-Nachrichten wird mit Hilfe sogenannter XSD-Dateien (XML Schema Definition) fixiert. Für EBICS sind die folgenden XSD-Dateien definiert worden und unter http://www.ebics.de (siehe “Spezifikation”) abrufbar:

- „ebics_request.xsd_H004“ enthält das XML-Schema für die Anfragen des Kundensystems
- „ebics_response.xsd_H004“ definiert das XML-Schema für die Antworten des Banksystems
- „ebics_orders.xsd_H004“ enthält auftragsspezifische Datenstrukturen
- „ebics_types.xsd_H004“ führt einfache EBICS-Typdeklarationen auf.

Neben diesen Hauptschemata sind an gleicher Stelle noch die folgenden spezifischen Varianten für Transaktionen zu finden, welche speziell das Schlüsselmanagement betreffen:

- „ebics_keymgmt_request.xsd_H004“ definiert das XML-Schema für Anfragen des Kundensystems im Rahmen des Schlüsselmanagements
- „ebics_keymgmt_response.xsd_H004“ enthält das XML-Schema für die Antworten des Banksystems im Rahmen des Schlüsselmanagements.

Der target namespace ist als “urn:org:ebics:H004” definiert.

Für die Übermittlung der EU in strukturierter Form wurde ein Schema „ebics_signature.xsd“ definiert, es hat die Schema Target Location http://www.ebics.de.org/S001, ist abrufbar unter http://www.ebics.de (siehe “Spezifikation”):

- Dieses Schema wurde als eigenständig definiert, um auch außerhalb EBICS verwendet werden zu können. Es ist auch der Import des o.g. Namespaces zur Nutzung der EU in EBICS erforderlich. Es hat das Prefix „esig“.
- Von „ebics_signature“ (in dieser Datei wurden für EBICS 2.5 keine Aktualisierungen/Änderungen vorgenommen) aus wird auf Strukturen des W3C-Standards "XML-Signature" referenziert mit den Prefix „ds“ (siehe dazu Kapitel 3.8). Dieses Schema ist an gleicher Stelle unter dem Namen „xmldsig-core-schema.xsd“ hinterlegt.

Jede der vier XSD-Dateien mit der Endung „_request_H004“ oder „_response_H004“ definiert eine oder mehrere Arten von EBICS XML-Nachrichten, die jeweils ein anderes XML-Wurzelelement („root element“) mit einem eindeutigen Namen haben.

Für Standard-EBICS-Nachrichten definiert „ebics_request.xsd_H004“ das Wurzelelement `ebicsRequest` für Anfragen des Kundensystem und „ebics_response_H004.xsd“ definiert das Wurzelelement `ebicsResponse` für Antworten des Banksystems. Für Transaktionen des Schlüsselmanagements enthält „ebics_keymgmt_request.xsd“ drei zusätzliche XML-Nachrichten für Anfragen des Kundensystems mit den Wurzelelementen `ebicsUnsecuredRequest, ebicsUnsignedRequest` und `ebicsNoPubKeyDigestsRequest.` „ebics_keymgmt response_H004.xsd“ definiert für das Schlüsselmanagement das Wurzelelement `ebicsKeyManagementResponse` für Antworten des Banksystems.

„ebics.xsd_H004“ inkludiert diese vier XML-Schema Dateien und enthält daher die Gesamtmenge aller Definitionen der EBICS-Schemaversion H004. Dies kann ebenfalls aus http://www.ebics.de (siehe “Spezifikation”). Sein target namespace ist ebenfalls “urn:org:ebics:H004”. Mit Hilfe dieser Datei kann verifiziert werden, dass alle globalen Definitionen im EBICS Namensraum (Elemente und Typen) eindeutige Namen haben. Diese Eigenschaft des EBICS XML-Protokolls vereinfacht die Verarbeitung von EBICS XML-Nachrichten durch Standard-XML-Werkzeuge, weil bereits der Name des XML-Wurzelelements zusammen mit der Angabe des EBICS Namensraums ausreicht, um das zulässige Format für die komplette XML-Nachricht zu bestimmen. Ein Standard-XML-Parser kann z.B. bereits anhand dieses XML-Fragments erkennen, gegen welche Definition in den EBICS XSD-Dateien das gesamte XML-Dokument validiert werden muss:

```
<ebicsRequest xmlns="urn:org:ebics:H004" Version="H004">
```

Die Schema Location besteht aus einem Referenz-Paar, getrennt durch ein Leerzeichen. Die erste Referenz ist der Name des Namespaces, die zweite Referenz ist ein Hinweis, wo das entsprechende Schema zu diesem Namespace zu finden ist, z.B. der lokale Dateiname ebics_request_H004.xsd:

```
<ebicsRequest xmlns=" urn:org:ebics:H004" Version="H004"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=""urn:org:ebics:H004 ebics_request_H004.xsd"">
```

Die Referenzierung von EBICS-XML-Elementen und -Attributen wird an folgendem Beispiel aus der XML-Schema-Datei „ebics_request.xsd_H004“ für die EBICS-Wurzelstruktur für Standardanfragen des Kundensystems („root element“ `ebicsRequest` und dessen Unterelemente) erläutert:

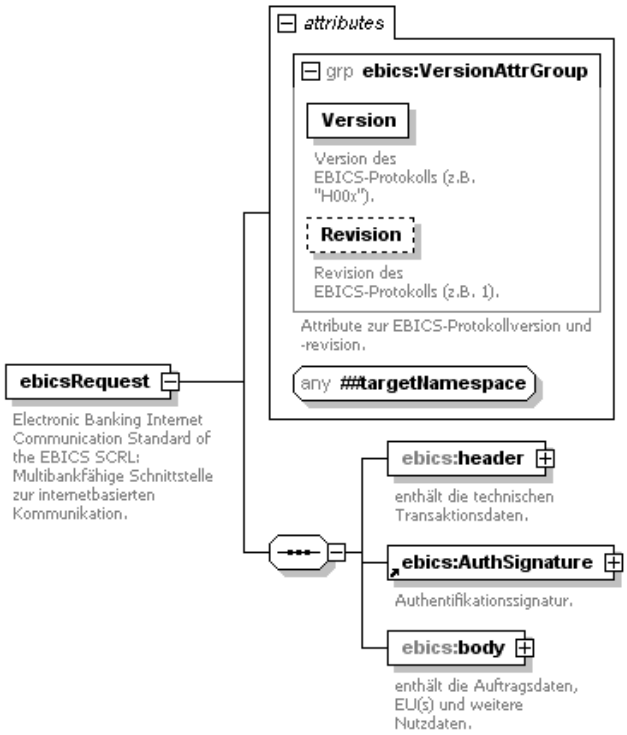


Abbildung 3: Wurzelstruktur des EBICS-Protokolls

Das XML-Wurzelelement für Standard-EBICS-Nachrichten mit Requests des Kundensystems heißt `ebicsRequest`. Es enthält einige Attribute mit fundamentalen Informationen, die zum Parsen der Nachricht als Ganzes benötigt werden (Attributgruppe `VersionAttrGroup`):

- `Version` für die EBICS-Protokollversion (z.B. „H004“)
- `Revision` für die EBICS-Protokollrevision: Dieses Attribut SOLLTE ebenfalls mitgeschickt werden, um mehrere (kompatible) Revisionen der gleichen Protokollversion fachlich unterscheiden zu können.

Die direkte Unterstruktur zu `ebicsRequest` bilden die folgenden Elemente:

- `header`: Das XML-Tag enthält technische Informationen (sog. „technische Steuerdaten“) in den Subtags:
  - `static` für diejenigen technischen Steuerdaten, die über die gesamte Transaktion hinweg konstant bleiben
  - `HostID` für die EBICS Host-ID zur Identifizierung des EBICS Bankrechners. Das Element `HostID` ist in allen EBICS Request-Nachrichten des Kundensystems

enthalten (sowohl für Standard-Transaktionen als auch für systembedingte Transaktionen). Die EBICS Host-ID muss nicht identisch sein zu der Host-ID für das FTAM-Verfahren und wird dem Kunden vom Kreditinstitut mitgeteilt.

- `mutable` für die veränderlichen technischen Steuerdaten.

Beide Subtags von `header` MÜSSEN in der obigen Reihenfolge erscheinen.

- `AuthSignature`: In diesem Element ist die Authentifikationssignatur gemäß dem Standard „XML-Signature“ untergebracht. Das XML-Tag MUSS in allen Nachrichten auftauchen mit Ausnahme der Auftragsarten INI, HIA, HSA und HPB (für diese Auftragsarten werden eigene XML-Schemata benutzt; siehe hierzu Kapitel 4.4).
- `body` enthält die eigentlichen Auftragsdaten, Unterschriften (EUs) und weitere Daten, die in direktem Bezug zum Auftrag stehen oder zu dessen Auswertung benötigt werden.

Die XML-Anfragen der Teilnehmer an die Kreditinstitute werden als EBICS-Requests bezeichnet, die XML-Antworten der Kreditinstitute als EBICS-Responses. Das HTTP-Binding eines EBICS-Requests und der zugehörigen EBICS-Response ist: der EBICS-Request wird in einen HTTP-POST-Request eingebettet, die EBICS-Response in die entsprechende HTTP-Response.

Ein typischer HTTP-Request sieht in EBICS wie folgt aus (Ausschnitt):

```
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=" urn:org:ebics:H004"
       xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="urn:org:ebics:H004 ebics_request_H004.xsd "
       Version="H004" Revision="1">
  <header authenticate="true">
    <static>
      <HostID>EBIXHOST</HostID>
      …
    </static>
    <mutable>
      …
    </mutable>
  </header>
  <AuthSignature>
    …
  </AuthSignature>
  <body>
    …
  </body>
</ebicsRequest>
```

Eine entsprechende mögliche HTTP-Response beschreibt der folgende Ausschnitt:

```
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:H004"
       xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation=" urn:org:ebics:H004 ebics_response_H004.xsd"
       Version="H004" Revision="1">
  <header authenticate="true">
    <static>
      …
    </static>
    <mutable>
      …
    </mutable>
  </header>
  <AuthSignature>
    …
  </AuthSignature>
  <body>
    …
  </body>
</ebicsResponse>
```

Weitere Details zu der Struktur von EBICS-Protokollnachrichten und Transaktionsdetails sind in Kapitel 5 aufgeführt. Die Formate der XML-Nachrichten für Standardantworten des Banksystems und die systembedingten Nachrichten des KeyManagements verwenden andere XML-Wurzelelemente, die weitgehend analog zum Standard-Request aufgebaut sind. Die kompletten XML-Schemata sind in der separaten HTML-Schemadokumentation zu finden.

Das Schema ebics_hev.xsd (ebenfalls für EBICS 2.5 nicht umbenannt oder aktualisiert)zur Abfrage der von der Bank unterstützten EBICS-Versionen steht unter http://www.ebics.org (siehe „Spezifikation“) zur Verfügung (siehe auch Kapitel 9.5)

3.2 Komprimierung, Verschlüsselung und Kodierung der Auftragsdaten

EBICS behandelt bankfachliche Nutzdaten transparent. Das bedeutet: Auftragsdaten werden unabhängig von der spezifischen Datenstruktur unterschiedlicher Auftragsarten als binärer Block behandelt und in die XML-Struktur eingebettet. Hierbei MÜSSEN diese Auftragsdaten vor dem Versand stets zunächst ZIP-komprimiert, dann hybrid verschlüsselt (gemäß Verfahren E002) und das Ergebnis schließlich base64-kodiert werden. Ausnahmen: Bei den Schlüsselmanagement-Auftragsarten INI, HIA, H3K und HSA erfolgt der Versand

unverschlüsselt (siehe hierzu Kapitel 4.4.1.2.5.1 für INI, HIA und H3K bzw. Kapitel 4.8.2 für HSA). Die Standards, die den in EBICS gültigen ZIP-Algorithmus sowie das base64-Format definieren, sind im Anhang (Kapitel 16) spezifiziert.

Die derart erzeugte Datenrepräsentation ist dann ohne weitere Zeichenkonvertierung oder Strukturierung z.B. in das XML-Element `ebicsRequest/body/DataTransfer/OrderData` einzustellen.

Die ZIP-Komprimierung dient zur Verringerung des Datenvolumens, das übertragen werden muss.

Bei der hybriden Verschlüsselung werden die eigentlichen Daten symmetrisch verschlüsselt. Der hierzu verwendete Transaktionsschlüssel wird wiederum asymmetrisch verschlüsselt und z.B. in Form des XML-Elements `ebicsRequest/body/DataTransfer/DataEncryptionInfo/TransactionKey` beigefügt (siehe hierzu Kapitel 6.2).

Die Verschlüsselung der Auftragsdaten erfolgt zusätzlich zur TLS-Transportverschlüsselung. Dadurch ist gewährleistet, dass die Auftragsdaten sowohl auf dem Weg über öffentliche Netze (ergänzend zu TLS) als auch jenseits der TLS-geschützten Verbindungsstrecke vor unautorisiertem Lesezugriff geschützt sind.

base64 verwendet zur Kodierung des Binärstroms nur druckbare ASCII-Zeichen und stellt somit sicher, dass die Auftragsdaten unverfälscht ihr Ziel erreichen und dort authentisch ausgewertet werden können.

3.3 Segmentierung der Auftragsdaten

Unter Segmentierung wird die Aufteilung von großen Datenvolumina in kleinere, einzelne Übertragungssegmente verstanden.

Die Segmentierung der Auftragsdaten findet bei EBICS auf Anwendungsprotokollebene statt. Auftragsdaten dürfen nur dann in einer einzelnen EBICS-Nachricht übermittelt werden, wenn sie in komprimierter, verschlüsselter und base64-kodierter Form die vorgegebene feste Größe von 1 MB nicht überschreiten. Das gilt für Sende- und Abholaufträge gleichermaßen. Wird die Grenze von 1MB überschritten, MÜSSEN die komprimierten, verschlüsselten und base64-kodierten Auftragsdaten in Segmente aufgeteilt werden, deren Größe jeweils die festgelegte Segmentgröße von 1 MB nicht überschreitet. Die Segmente werden dann in fortlaufender Reihenfolge in einzelnen EBICS-Nachrichten übertragen.

Weitere Details zur Segmentierung der Auftragsdaten sind Kapitel 7 zu entnehmen.

3.4 Wiederaufsetzen der Übertragung von Aufträgen (Recovery) [optional]

Recovery ermöglicht es, die Übertragung eines Auftrags nach einem Transport- oder Verarbeitungsfehler fortzusetzen, ohne alle bereits erfolgreich versandten Auftragsdatensegmente erneut übertragen zu müssen.

Basierend auf dem Ablauf der Übertragung von Aufträgen in mehreren fest vorgegebenen Schritten definiert EBICS ein Recovery-Verfahren auf der XML-Anwendungsprotokollebene. Es ist ein optimistisches Recovery-Verfahren, das auf einen separaten Synchronisationsschritt verzichtet, da das Kundensystem in der Regel weiss, mit welchem Schritt die Übertragung des betroffenen Auftrags fortzusetzen ist.

Details zur Recovery sind Kapitel 5.4 zu entnehmen.

3.5 Elektronische Unterschrift (EU) der Auftragsdaten

Die „**E**lektronische **U**nterschrift“ (EU) der Auftragsdaten stellt die Authentizität der Auftragsdaten sicher, jenseits der TLS-Übertragungsstrecke und unabhägig von der Komprimierung, Verschlüsselung, Kodierung sowie Segmentierung der Auftragsdaten.

Bei Sendeaufträgen handelt es sich um die willentliche Signatur eines Teilnehmers, welche die inhaltliche Zustimmung (content commitment) des Teilnehmers dokumentiert, bei Abholaufträgen ist es die Signatur des Kreditinstituts.

EUs werden gemäß Anhang (Kapitel 14) erzeugt, in der EBICS-Version „H004“ muss mindestens die EU-Version „A004“ unterstützt werden (siehe Anhang (Kap 11.2)).

3.5.1 EU der Teilnehmer

Die Auftragsdaten der Sendeaufträge MÜSSEN vor dem Versand signiert, d.h. mit mindestens einer EU versehen werden. Ausnahmen bilden die Schlüsselmanagement-Auftragsarten INI und HIA, die nicht bankfachlich signiert werden.

Aus einer EU eines Teilnehmers kann das Banksystem mit Hilfe des öffentlichen Signaturschlüssels des Unterzeichners gemäß dem verwendeten und von der Bank bzgl. EBICS unterstützten Signaturverfahren den Hashwert der signierten Auftragsdaten extrahieren.

Für die EUs von Teilnehmern sind die folgenden **Unterschriftklassen** definiert, die hier in der Reihenfolge abfallender Stärke aufgelistet werden („E“ ist die stärkste, „T“ die schwächste Unterschriftsklasse):

- **Einzelunterschrift (Typ „E”)**
- **Erstunterschrift (Typ „A“)**.
- **Zweitunterschrift (Typ „B“)**

- **Transportunterschrift (Typ „T“).**

Durch die Zuordnung von Unterschriftsklassen zu Teilnehmern wird innerhalb des Kreditinstituts ein Berechtigungsmodell für EUs definiert. Beispielsweise haben Teilnehmer mit der Unterschriftsklasse A die Berechtigung zur Erstunterschrift von Aufträgen. Institutsindividuell KÖNNEN detailliertere Berechtigungsmodelle definiert werden, bei denen die Unterschriftsberechtigung eines Teilnehmers bezüglich der Auftragsart und/ oder des Betragslimits und/ oder des benutzten Kontos parametrisierbar ist.

Die Unterschriftsklasse einer gegebenen EU eines Teilnehmers ist die stärkste Klasse, die dieser EU im Berechtigungsmodell des entsprechenden Kreditinstituts zugeordnet werden kann.

Unterschriftsberechtigungen des Typs „T“ werden Teilnehmern pauschal zugewiesen oder (in detaillierteren Berechtigungsmodellen) Teilnehmern in Kombination mit bestimmten Auftragsarten. Sie sind jedoch nicht abhängig von Konten oder Betragslimits. Transportunterschriften werden nicht zur bankfachlichen Autorisation von Aufträgen verwendet, sondern lediglich zu deren (autorisierten) Einreichung an das Banksystem.

Als **bankfachliche EUs** bezeichnet man EUs vom Typ „E“, „A“, oder „B“. Bankfachliche EUs dienen der Autorisierung von Aufträgen. Aufträge können mehrere bankfachlichen EUs benötigen, die dann von unterschiedlichen Teilnehmern geleistet werden MÜSSEN. Die Unterzeichner eines Auftrags können zudem Teilnehmer unterschiedlicher Kunden sein.

Für jede unterstützte Auftragsart wird zwischen Kreditinstitut und Kunde eine **Mindestanzahl erforderlicher bankfachlicher EUs** vereinbart.

Details zum Format der EU und ihrer Anwendung zur Auftragsautorisation werden im Anhang (Kapitel 11.2) erläutert.

3.5.2 EU der Kreditinstitute [vorgesehen]

Die EU der Kreditinstitute ist eine *vorgesehene* Funktionalität von EBICS. Der Grund ist die fehlende abschließende rechtliche Betrachtung dieser EU.

Sowohl im vorliegenden Feinkonzept als auch in den XML-Schemadateien von EBICS wurden Vorbereitungen getroffen, die die Umsetzung folgender Festlegungen in zukünftigen EBICS-Versionen erleichtern:

- verpflichtende EU des Hashwertes und der Displaydatei des zu unterzeichnenden Auftrags im Falle der Auftragsart HVD
  Siehe Kapitel 8.3.2.2.
- optionale EU der Abholdaten bei Abholaufträgen
  Es ist je Teilnehmer und Auftragsart konfigurierbar, ob der Teilnehmer die entsprechen-

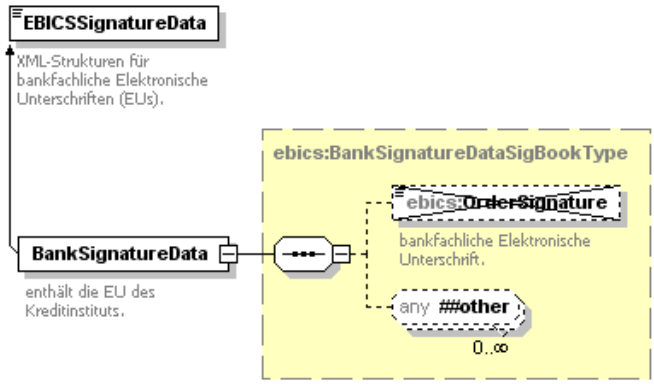
den Abholdaten mit oder ohne EU des Kreditinstituts anfordern muss.
Siehe Kapitel 3.12, Kapitel 5.5.1.2.1 (Verarbeitungsschritt “Überprüfung Dateiattribute”), sowie Kapitel 5.6.1.1.1.

- Abholung des öffentlichen bankfachlichen Schlüssels des Kreditinstituts über die Auftragsart HPB
  Siehe Kapitel 4.4.2.2
- Überprüfung des Hashwerts des öffentlichen bankfachlichen Schlüssels des Kreditinstituts im Rahmen der Transaktionsinitialisierung
  Bestandteil jeder Transaktionsinitialisierung ist die Überprüfung der öffentlichen Schlüssel des Kreditinstituts, über die der Teilnehmer verfügt. Ausgenommen sind die Auftragsarten des Schlüsselmanagement INI, HIA, HPB, HSA sowie die Auftragsart HEV zur Abfrage der bankseitig unterstützen EBICS-Versionen.
  Siehe Kapitel 4.6.2 und Kapitel 5.5.1.2.1.
- Binärformat für die EU der Kreditinstitute analog zum Binärformat der Teilnehmer-EU
  Siehe Anhang (Kapitel 11.2.2).

3.5.3 Darstellung der EUs in EBICS-Nachrichten

Die EUs eines Auftrags werden mit Hilfe der XML-Elemente `BankSignatureData` (für die Bank-EU; vorgesehenes Feature, das in der Schemadatei „ebics_orders_H004.xsd“ definiert ist) und `UserSignatureData` (für die Teilnehmer-EU, das in der Schemadatei „ebics_signature.xsd“ definiert ist) dargestellt. Sie substituieren jeweils das abstrakte Element `EBICSSignatureData` . Abbildung 4 enthält die graphische Darstellung von `EBICSSignatureData`: die einzelnen EUs sind im Signaturverfahren A004 jeweils in einem Element `OrderSignature` enthalten. Ab den Verfahren A005/A006 sind sie im Element `OrderSignatureData` enthalten. EUs werden gemäß Anhang (Kapitel 14) gebildet. `OrderSignature` enthält die base64-Kodierung der entsprechenden EU-Datei. Da EU-Dateien im Signaturverfahren A004 keine Kunden-IDs enthalten, MUSS bei den EUs von Teilnehmern zusätzlich das Attribut `PartnerID` mit der Kunden-ID des Unterzeichners (=Einreichers) befüllt werden. In der strukturierten Signaturinformation `UserSignatureData` für Signaturverfahren ab A005/006 ist die Kunden-ID bereits im Element `PartnerID` enthalten. Die Angabe einer abweichenden Kunden-ID für die kundenübergreifende EU ist ausschließlich mit der Auftragsart HVE über spezielle Auftragsparameter möglich.

Die bankfachliche EU des Kreditinstituts wird analog zur bekannten bankfachlichen Teilnehmer-EU gebildet (siehe Anhang (Kapitel 11.2.2) im Vergleich zu Anhang (Kapitel 14)), wobei in diesem Fall das Attribut `PartnerID` entfällt.



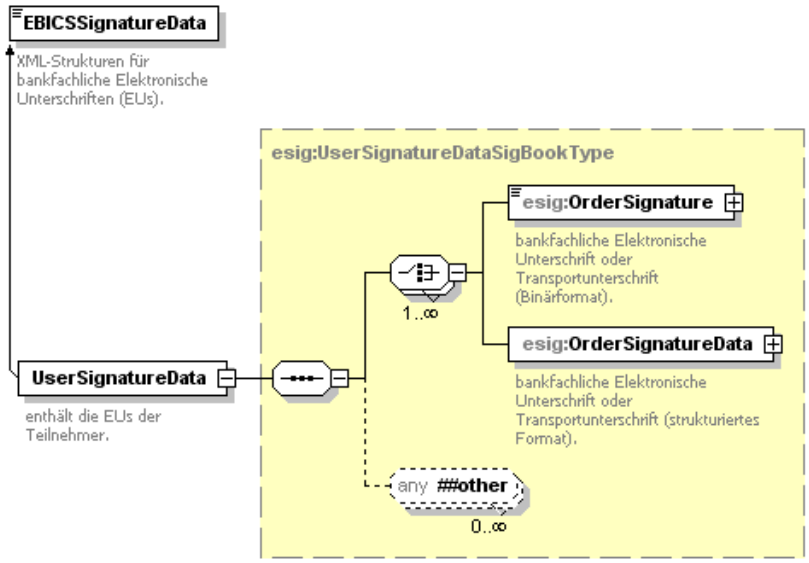


Abbildung 4: XML-Strukturen BankSignatureData und UserSignatureData für die EUs eines Auftrags in binärer und strukturierter Form

Um die EUs eines Auftrags in EBICS-Nachrichten einzubetten, sind die folgenden Schritte notwendig:

- Erstellung eines Instanzdokuments zu ebics_orders_H004.xsd bzw. ebics_signature.xsd, welches nur aus dem Element `BankSignatureData` (Bank-EU) bzw. `UserSignatureData` (Teilnehmer-EU) besteht.
- ZIP-Komprimierung, Verschlüsselung, base64-Kodierung des Instanzdokuments
  Die Verschlüsselung erfolgt mit dem Transaktionsschlüssel `TransactionKey` aus dem XML-Zweig `ebicsRequest/body/DataTransfer/DataEncryptionInfo` (siehe Kapitel 6.2).
- Das Resultat wird in das Element `SignatureData` im Zweig `DataTransfer` des EBICS-Rumpfs eingestellt (siehe Kapitel 3.1.4).

3.6 Vorabprüfung [optional]

Der Teilnehmer KANN bei Upload-Aufträgen im ersten Transaktionsschritt Informationen versenden, die das Banksystem – sofern es diese Funktionalität unterstützt – zur Vorabprüfung des Auftrags nutzen KANN. Die Vorabprüfung kann eine oder mehrere der folgenden Prüfungen umfassen: Kontoberechtigungsprüfung, Limitprüfung, EU-Verifikation. Treten bei der Vorabprüfung (fachliche) Fehler auf, so ist die Fortsetzung der Übertragung des Auftrags sinnlos, zumal der Auftrag nicht durchgeführt werden kann.

Ob ein Kreditinstitut die Vorabprüfung generell unterstützt, können Teilnehmer über die Bankparameterabfrage (Auftragsart HPD, zurückgelieferte XML-Struktur `HPDResponseOrderData`, Attribut `ProtocolParams/PreValidation@supported`) erfahren. Mitgelieferte Parameter für Vorabprüfungen, die ein Kreditinstitut nicht unterstützt, werden von dem Kreditinstitut ignoriert.

Mehr zur Auftragsart HPD ist in Kapitel 9.2 zu finden. Für Details zur Vorabprüfung siehe Kapitel 5.3.

3.7 Technische Teilnehmer

EBICS-Kundensysteme können selbst wieder als Client-Server-Systeme, sogenannte MultiUser-Systeme, realisiert werden. In diesem Fall übernimmt der Server die Rolle des EBICS-Clients in der Kommunikation mit dem Banksystem und ist als solcher verantwortlich für die Übermittlung von Aufträgen gemäß der EBICS Spezifikation.

Dieser kundenseitige Server stellt sich gegenüber dem Banksystem als „technischer Teilnehmer“ dar, der im Wesentlichen wie ein (menschlicher) Teilnehmer auf dem Banksystem administriert wird.

Die EBICS-Requests eines technischen Teilnehmers und eines menschlichen Teilnehmers unterscheiden sich nur dadurch, dass der technische Teilnehmer in allen EBICS-Requests das Feld SystemID mit seiner Teilnehmerkennung belegt und die Authentifikationssignatur für den EBICS-Request erstellt.

EBICS-Responses für den technischen Teilnehmer werden immer mit dem öffentlichen Verschlüsselungsschlüssel des technischen Teilnehmers verschlüsselt.

Für den technischen Teilnehmer gilt folgendes:

- Im EBICS-Request wird grundsätzlich (zusätzlich zu den Feldern PartnerID und UserID) das Feld SystemID mit der Teilnehmerkennung des technischen Teilnehmers versorgt. Am Vorhandensein des Feldes SystemID erkennt das Banksystem, dass der Request von einem technischen Teilnehmer versendet wurde.

- der technische Teilnehmer leistet die Authentifikationssignatur für den EBICS-Request (Ausnahme sind nur die Auftragsarten, die keine Authentifikationssignatur benötigen)

- der technische Teilnehmer kann alle EBICS-Requests für den im Feld UserID genannten Teilnehmer durchführen.

- der technische Teilnehmer kann keine bankfachliche Unterschrift leisten.

- der technische Teilnehmer kann Dateien mit einer eigenen Transportunterschrift einreichen (D-Datei oder Einreichung in die VEU).

- der technische Teilnehmer kann Dateien mit den bankfachlichen Unterschriften menschlicher Teilnehmer einreichen. In diesem Fall muss der technische Teilnehmer keine Transportunterschrift leisten.

Für die Prüfungen auf dem Banksystem gilt folgendes:

- Anhand der Belegung des Feldes SystemID wird die Prüfung der Authentifikationssignatur des vom technischen Teilnehmers erstellten EBICS-Requests vorgenommen.

- Zur Prüfung der Auftragsberechtigung wird nur der Inhalt der Felder PartnerID und UserID herangezogen. Der Inhalt des Feldes SystemID wird hier nicht berücksichtigt.

  Nur wenn der technische Teilnehmer EBICS-Requests im eigenen Namen (im Feld UserID steht die Teilnehmerkennung des technischen Teilnehmers) durchführt, benötigt er auch die entsprechende Auftragsberechtigung im Banksystem.

- Eine Kontoprüfung erfolgt für technische Teilnehmer nicht.

- Die Prüfung der elektronischen Unterschrift erfolgt, wie sonst auch, unabhängig von der Belegung der Felder SystemID und UserID.

3.8 Authentifikationssignatur

Die Authentifikation des Teilnehmers bzw. des Kundensystems und des Kreditinstituts in jedem Transaktionsschritt sind notwendig, um den bankseitigen Verbrauch von Ressourcen durch Unberechtigte und die unautorisierte Zustandsänderung von Aufträgen oder Daten zu unterbinden.

Die Authentifikationssignatur stellt als Haupt-XML-Zweig zwischen den EBICS-Kopf- und -Rumpfdaten einen integralen Bestandteil des EBICS-Protokolls dar. Sie wird gemäß dem XML-Signature-Standard erzeugt und hat mehrere Aufgaben zu erfüllen:

1. Identifikation und Authentisierung des (technischen) Teilnehmers: Das Banksystem MUSS sich mit Hilfe der Authentifikationsignatur von der Korrektheit der

(technischen) Teilnehmerkennung ihm bekannter Teilnehmer bzw. Kundensysteme überzeugen.

2. Integrität der Steuerdaten/EU(s): Änderungen – auch jenseits der TLS-Übertragungsstrecke – an der/den EU(s) sowie den technischen und auftragsbezogenen Daten (mit Ausnahme der Auftragsdaten, welche nicht von der Authentifikationssignatur, sondern ausschließlich von der bankfachlichen Unterschrift erfasst werden) werden mit Hilfe der Authentifikationssignatur aufgedeckt, solange die XML-Struktur der signierten Daten unverändert bleibt.

Die Authentifikationssignatur wird (im Gegensatz zur EU, die die Auftragsdaten signiert) über die Steuerdaten und über die EU(s) gebildet und MUSS bei jedem Transaktionsschritt jeder Auftragsart (mit Ausnahme der systembedingten Auftragsarten INI, HIA, H3K, HSA und HPB, siehe hierzu Kapitel 4.4) sowohl vom Kunden- als auch vom Banksystem geleistet werden. Die Authentikation der bankfachlichen EU(s) koppelt die mit dieser/diesen EU(s) signierten Auftragsdaten an die restlichen Protokollinformationen und verhindert so den unautorisierten Austausch von Aufträgen mitsamt ihren EU(s) innerhalb einer EBICS-Transaktion.

Details zu den verwendeten Algorithmen für die Authentifikationssignatur sind in Kapitel 11.1 zu finden. Dort ist auch festgelegt, dass ein Kanonisierungsverfahren (C14N) die Daten vor der Signaturerzeugung und -verifikation in eine standardisierte Form überführt.
Zur Signaturbildung MÜSSEN – zusätzlich zu den XML-Signature-eigenen Strukturen – genau diejenigen Elemente (und deren Substrukturen) in die Authentifikationssignatur eingehen, die den Attributmarker `@authenticate="true"` besitzen. Die Vorkommen dieser Attributmarker sind in den XML-Schemata festgelegt.

Die Authentifikationssignatur jeder EBICS-Nachricht MUSS vom jeweiligen Empfänger der Nachricht überprüft werden.

Kann die Authentifikationsignatur eines EBICS-Requests nicht erfolgreich verifiziert werden, so kann das Banksystem nicht davon ausgehen, dass der EBICS-Request auch tatsächlich vom entsprechenden (technischen) Teilnehmer stammt.

Der Sender des EBICS-Request erhält in diesem Fall einen entsprechenden Fehlercode (EBICS_AUTHENTICATION_FAILED). Weitere Einzelheiten sind jeweils in den Kapiteln 5.5.1.2.1 und 5.5.1.2.2 enthalten, jeweils zu dem Unterpunkt „Prüfung der Authentizität der EBICS-Requests“.

Kann umgekehrt die Authentifikationssignatur des EBICS-Response nicht erfolgreich überprüft werden, so kann das Kundensystem nicht davon ausgehen, dass der EBICS-Response vom erwarteten Banksystem stammt. In diesem Fall MUSS die betroffene EBICS-Transaktion durch das Kundensystem abgebrochen werden.

Auf Kundenseite MÜSSEN dann die Einstellungen der Kundensoftware überprüft werden, die beim Verbindungsaufbau zum Banksystem unter Einhaltung der Anforderungen aus Kapitel

3.1.2.2 verwendet werden. Des Weiteren MUSS die Aktualität der gespeicherten öffentlichen Schlüssel des Kreditinstituts überprüft werden (siehe dazu auch Kapitel 5.5.1.2.1, Unterpunkt „Überprüfung der Hashwerte der Bankschlüssel“).

3.9 X.509-Daten

Das EBICS-Protokoll greift in der Version H004 für die kryptographischen Algorithmen (d.h. für Authentifikation, Verschlüsselung, Signatur) standardmäßig auf öffentliche Schlüssel zurück, die im Rahmen der Teilnehmerinitialisierung zwischen Teilnehmer und Kreditinstitut ausgetauscht wurden (zum Schlüsselmanagement siehe Kapitel 4).
Für Nachfolger-Versionen der EBICS-Version „H004“ sind aber bereits Platzhalter für X.509-Daten vorgesehen (z.B. Zertifikate, Seriennummern, Distinguished Names, etc.).
Es wird direkt auf die Strukturen des W3C-Standards referenziert.
Das optionale Feld dazu befindet sich in den EBICS-XML-Schemata „ebics_request_H004.xsd“ und „ebics_keymgmt_request_H004.xsd“ z.B. im Pfad `ebicsRequest/body/X509Data`. Die Typdefinition benutzt die Vorgabe aus XML-Signature (siehe Abbildung 5).

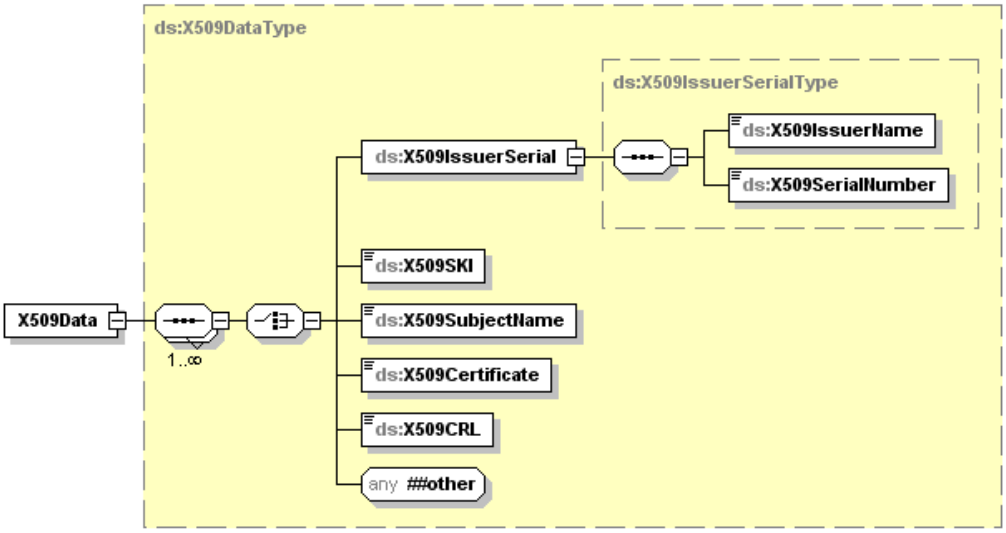


Abbildung 5: X509DataType

In Standard-Requests kann dies noch nicht verwendet werden. Zur Befüllung oder Auswertung dieses Felds sind noch keine fachlichen Vorgaben festgelegt.
In der EBICS-Schema-Version H004 können in den Auftragsarten des Schlüsselmanagements zusammen mit öffentlichen Schlüsseln optional zusätzliche X.509-Daten wie Zertifikate übertragen werden.

Die Information darüber, ob X.509-Daten vom Kreditinstitut bereits unterstützt werden, liefert die Auftragsart HPD (Bankparameter abholen) mit dem Feld `ProtocolParams/X509Data`. Siehe hierzu Kapitel 9.2.

3.10 Unterstützte Auftragsarten

Es werden alle standardisierten, systembedingten und reservierten Auftragsarten gemäß vollständiger Liste (siehe Anhang Kapitel 13) durch transparente Einbettung der Auftragsdaten in die XML-Struktur unterstützt.
Folgende bisher bekannte Auftragsarten werden mit folgender Begründung nicht mehr unterstützt (**Ausnahmen**):

- BPD (Bankparameterdatei abholen) wurde durch die neue verpflichtende Auftragsart HPD (Bankparameter abholen) ersetzt. Die ursprüngliche BPD-Struktur findet sich aber noch in den XML-Rückgabedaten (XML-Element `BankParameters`) zu den optionalen Auftragsarten HKD (Kunden- und Teilnehmerdaten des Kunden abholen) und HTD (Kunden- und Teilnehmerdaten des Teilnehmers abholen). Siehe hierzu Kapitel 9.3 und 9.4
- PWA (Passwortänderung senden): Passwörter werden bei EBICS nicht verwendet. PWA entfällt daher ersatzlos
- VPB (öffentlichen Verschlüsselungsschlüssel des Kreditinstitut abholen) wurde in die neue Auftragsart HPB (öffentliche Bankschlüssel abholen) integriert
- VPK (öffentlichen Verschlüsselungsschlüssel des Kunden senden) wurde in die neue Auftragsart HIA (öffentliche Verschlüsselungs- und Authentifikationsschlüssel des Teilnehmers senden) integriert. Außerdem werden die Schlüssel nun nicht mehr pro Kunde, sondern pro Teilnehmer eines Kunden erzeugt.

Die **neu hinzugekommenen Auftragsarten**

- FUL (Datei mit beliebigem Format senden)
- FDL (Datei mit beliebigem Format abholen)
- HAA (abrufbare Auftragsarten abholen)
- HCA (Änderung der Teilnehmerschlüssel für Authentifikation und Verschlüsselung)
- HCS (Änderung der Teilnehmerschlüssel für EU, Authentifikation und Verschlüsselung)
- HEV (Unterstützte EBICS-Versionen abrufen)
- HIA (Übermittlung der Teilnehmerschlüssel für Authentifikation und Verschlüsselung im Rahmen der Teilnehmerinitialisierung)
- HSA (Übermittlung der Teilnehmerschlüssel für Authentifikation und Verschlüsselung im Rahmen der Teilnehmerinitialisierung für Teilnehmer, die über einen DFÜ-Zugang über FTAM verfügen)

- HKD (Kunden- und Teilnehmerdaten des Kunden abholen)
- HPB (Transfer der öffentlichen Bankschlüssel)
- HPD (Bankparameter abholen)
- HTD (Kunden- und Teilnehmerdaten des Teilnehmers abholen)
- HVD (VEU-Status abrufen)
- HVE (VEU-Unterschrift hinzufügen)
- HVS (VEU-Storno)
- HVT (VEU-Transaktionsdetails abrufen)
- HVU (VEU-Übersicht abholen)
- H3K (Übermittlung aller drei Teilnehmerschlüssel (für EU, Authentifikation und Verschlüsselung im Rahmen der Teilnehmerinitialisierung im Falle der Verwendung von Zertifikaten)

werden in den Kapiteln 8 und 9 (HIA/H3K: Kapitel 4.4.1, HCA/HCS: Kapitel 4.6.1, HSA: Kapitel 4.8.2) ausführlich erläutert.
Informationen zur bankseitigen Unterstützung (verpflichtend, optional oder konditional) siehe Kapitel 13.

3.11 Auftragsparameter

Für die Übertragung von Auftragsparametern, die nicht Bestandteil der Auftragsdaten sind, wurde in die festen Steuerdaten (unter `ebicsRequest/header/static/OrderDetails`) das Element `OrderParams` integriert. Je nach Auftragsart hat dieses abstrakte Element eine spezifische, konkrete Ausprägung (siehe auch Abbildung 6):

- `HVDOrderParams` bei Auftragsparametern für die Auftragsart HVD
- `HVEOrderParams` bei Auftragsparametern für die Auftragsart HVE
- `HVSOrderParams` bei Auftragsparametern für die Auftragsart HVS
- `HVTOrderParams` bei Auftragsparametern für die Auftragsart HVT
- `HVUOrderParams` bei Auftragsparametern für die Auftragsart HVU
- `FULOrderParams` bei Auftragsparametern für die Auftragsart FUL
- `FDLOrderParams` bei Auftragsparametern für die Auftragsart FDL
- `StandardOrderParams` bei Auftragsparametern für Auftragsarten aus dem Anhang Kapitel 13 oder aus dem Document “EBICS Anhang 2 Auftragsartenkennungen”), die einen Datumsbereich übergeben (z.B. STA, Ausnahme: FDL)
- `GenericOrderParams` bei Auftragsparametern für alle weiteren Auftragsarten mit dem Bedarf zur zusätzlichen Informationsübertragung vom Kunden- zum Banksystem.

Die Strukturen für die Auftragsparameter zu den Auftragsarten HVD, HVE, HVS, HVT und HVU werden in Kapitel 8.3 näher erläutert. Die Strukturen zu den Auftragsarten FUL und FDL sind in Kapitel 9.6 näher erläutert.

Die `StandardOrderParams` definieren über das optionale Element `DateRange` und die Subelemente `Start` und `End` (beide vom Typ `date`) einen Datumsbereich.

Bei den `GenericOrderParams` kann über das optionale Element `Parameter` eine beliebige Anzahl von `Name-Value`-Paaren angegeben werden, wobei der Typ des Werts mit dem Attribut `Type` festzulegen ist (Die Empfehlung für einen Default ist `string`).

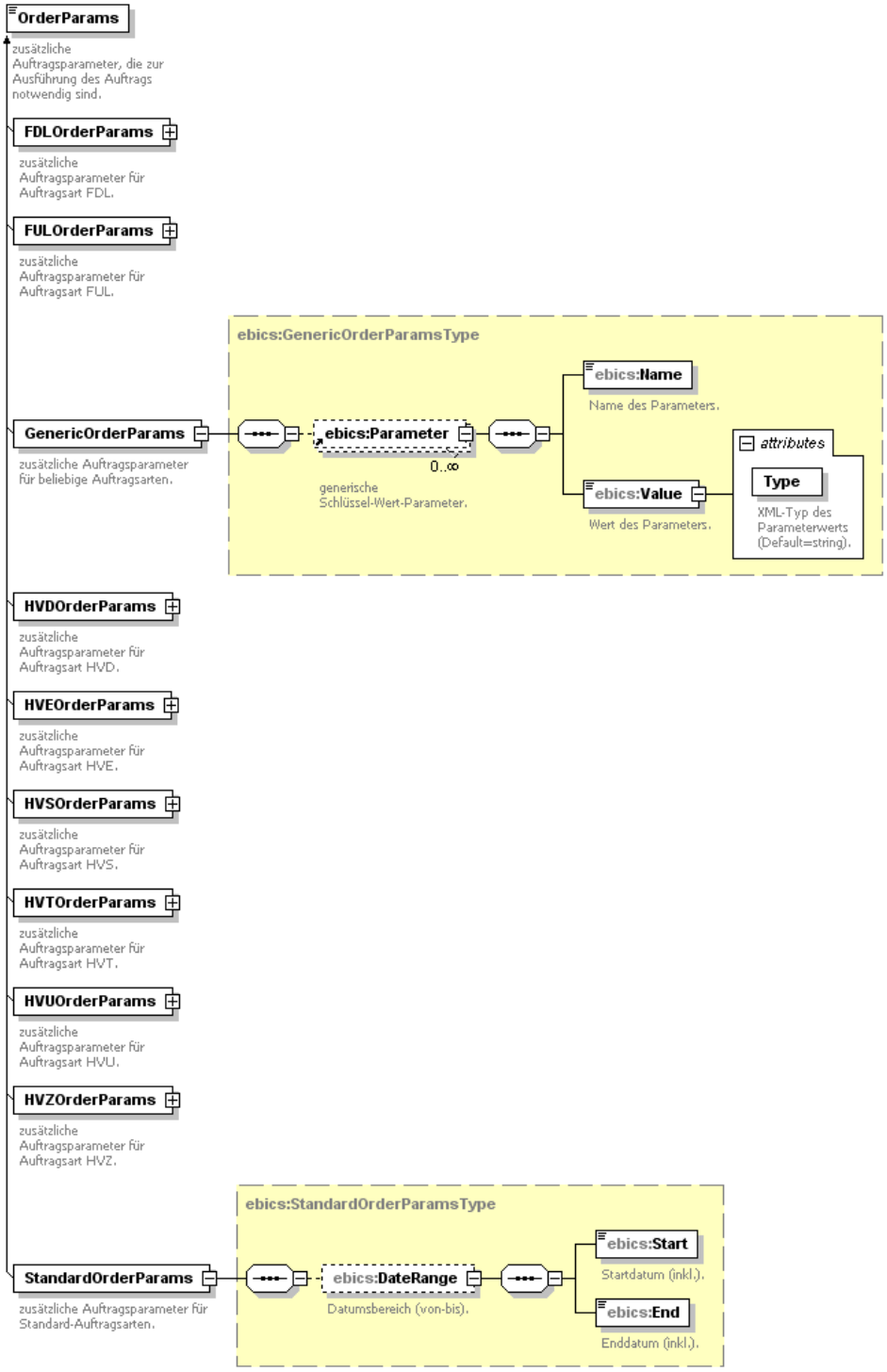


Abbildung 6: Mögliche Ausprägungen für die Auftragsparameter (OrderParams)

3.12 Ablauf der EBICS-Transaktionen

Dieses Kapitel enthält eine vereinfachte Beschreibung des Protokollablaufs für die Übermittlung eines DFÜ-Auftrags über EBICS, die den Festlegungen der vorhergehenden Kapitel Rechnung trägt.

Die Übermittlung erfolgt in einer EBICS-Transaktion, die aus mehreren Transaktionsschritten bestehen kann. Ein Transaktionsschritt ist ein Paar bestehend aus einem EBICS-Request und dem dazu gehörigen EBICS-Response.

Der erste Transaktionsschritt ist der Transaktionsinitialisierungsschritt. In diesem Schritt werden teilnehmerbezogenene Berechtigungsprüfungen durchgeführt, wie etwa die Prüfung der Auftragsartberechtigung. Die erfolgreiche Berechtigungsprüfung ist Voraussetzung für die Fortsetzung der Transaktion. Des Weiteren werden in diesem Transaktionsschritt die EUs des Auftrags übermittelt: bei Sendeaufträgen werden im EBICS-Request die EUs der Unterzeichner übertragen, bei Abholaufträgen eventuell die bankfachliche EU des Kreditinstituts im EBICS-Response.

Auf die Transaktionsinitialisierung folgen in der Regel mehrere Transaktionsschritte, in denen die Segmente der Auftragsdaten sequentiell und in fortlaufender Reihenfolge übertragen werden.

Sendeaufträge, die über EBICS an das Banksystem übermittelt werden, können auf zwei unterschiedliche Arten autorisiert werden:
Variante 1: Autorisierung mit Hilfe einer oder mehrerer bankfachlicher EUs

Die bankfachlichern EUs zu einer Auftragsdatei müssen von unterschiedlichen Teilnehmern geleistet werden, wobei im Falle der VEU diese Teilnehmer im Sonderfall auch zu unterschiedlichen Kunden gehören können (kundenübergreifende Unterschrift). Die EUs können mit drei unterschiedliche Auftragsarten zum Kreditinstitut übertragen werden, dabei stammen alle mit einer einzelnen EBICS-Transaktion eingereichten EUs jeweils vom selben Kunden.

1. Einreichung der Nutzdaten des Auftrags zusammen einer oder mehreren EUs durch einen Sendeauftrag mit den Auftragsattributen“**OZHNN**”.
   Alle auf diese Weise eingereichten EUs stammen vom Kunden des Einreichers des Auftrags.
   Sollten die übermittelten EUs für die bankfachliche Freigabe nicht ausreichen, wird der Auftrag an die VEU übergeben. Für die nachträgliche Autorisierung gibt es dann die folgenden Möglichkeiten:
2. Einreichung weiterer bankfachlicher EUs in einer eigenständigen EBICS-Transaktion mit den Auftragsattributen “**UZHNN**” zu derselben Auftragsart und -nummer
   Für diese EBICS-Transaktion entfallen die Transaktionsschritte zur Übermittlung der Nutzdatensegmente.

Die auf diese Weise nachgereichten EUs stammen vom Kunden des Einreichers des Auftrags.

3. Einreichung noch ausstehender bankfachlicher EUs mit Hilfe der Auftragsart **HVE**
   Werden EUs per HVE-Transaktion eingereicht, so müssen diese EUs jeweils vom Kunden des Einreichers dieser HVE-Transaktion stammen:
   HVE ermöglicht den Sonderfall der kundenübergreifenden Unterschrift, weil die über HVE eingereichten EUs nicht notwendigerweise vom Kunden des Einreichers des Auftrags stammen müssen.

Variante 2: Autorisierung mit Hilfe eines handschriftlich unterschriebenen Begleitzettels
Zum Übertragen der Auftragsdatei werden die Auftragsattribute des Sendeauftrags gleich "**DZHNN**" gesetzt.
Im Rahmen der EBICS-Transaktion wird eine EU der Unterschriftsklasse „T" übertragen zusammen mit den Nutzdaten des Auftrags. Der Auftrag wird nicht an die VEU weitergereicht, sondern direkt an die bankspezifische Weiterverarbeitung.
Verfügt der einreichende Teilnehmer im Banksystem über die Berechtigung zur Durchführung einer bankfachlichen EU, so werden diese Unterschriften bei der Einreichung mit den Auftragsattributen „DZHNN" grundsätzlich nur als Transportunterschrift gewertet. Es darf auch in diesem Fall keine Einleitung in die VEU erfolgen.

Die Bedeutung und die zulässigen Belegungen der Autragsattribute sind im Anhang (Kapitel 12.3) beschrieben.

Die Übermittlung eines Sendeauftrags mit einem (komprimierten, verschlüsselten und base64-kodierten) Nutzdatenvolumen von 3MB wird beispielhaft mit Hilfe des Sequenzdiagramms aus Abbildung 7 dargestellt. Die EBICS-Transaktion zu einem Sendeauftrag wird beendet, sobald das letzte Nutzdatensegment erfolgreich an das Kreditinstiut übertragen wurde.

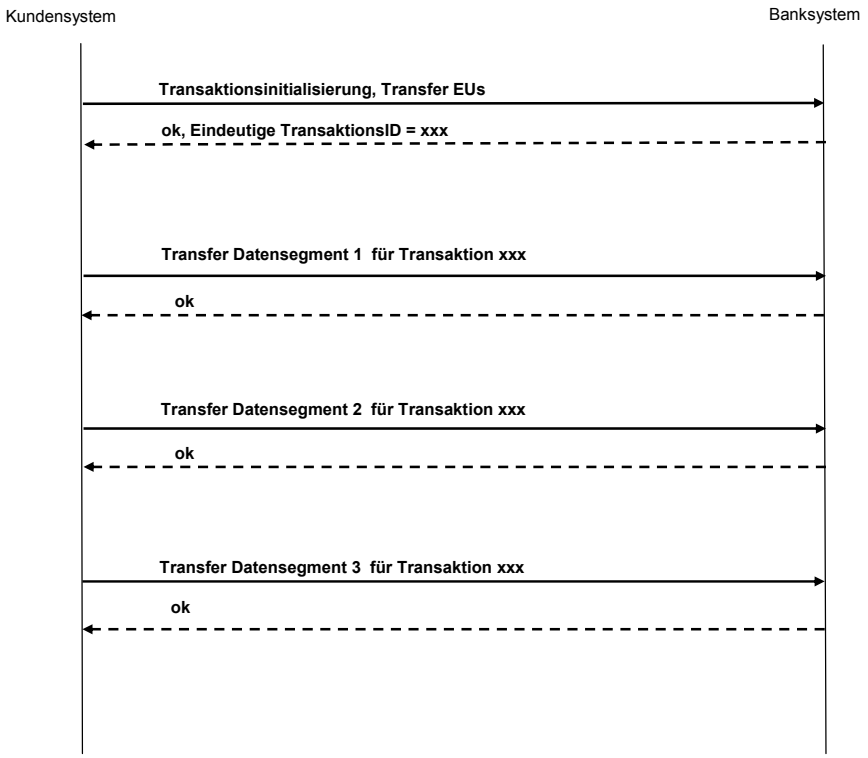


Abbildung 7: Beispielhafter Ablauf einer EBICS-Transaktion für einen Sendeauftrag

Die Übermittlung eines Abholauftrags mit einem (komprimierten, verschlüsselten und base64-kodierten) Nutzdatenvolumen von 3MB wird beispielhaft anhand des Sequenzdiagramms in Abbildung 8 verdeutlicht. Der Teilnehmer setzt die Auftragsattribute gleich

- „OZHNN“,
  wenn er die Abholdaten mit der bankfachlichen EU des Kreditinstituts anfordern möchte. In der Version „H004“ von EBICS ist die EU der Kreditinstitute nur vorgesehen (siehe Kapitel 3.5.2)
- „DZHNN“
  bei Anforderung der Abholdaten ohne die bankfachliche EU des Kreditinstituts.

Bei Abholaufträgen wird der Empfang der Abholdaten mit einem Quittierungsschritt bestätigt. Danach wird die EBICS-Transaktion zum Abholauftrag beendet.

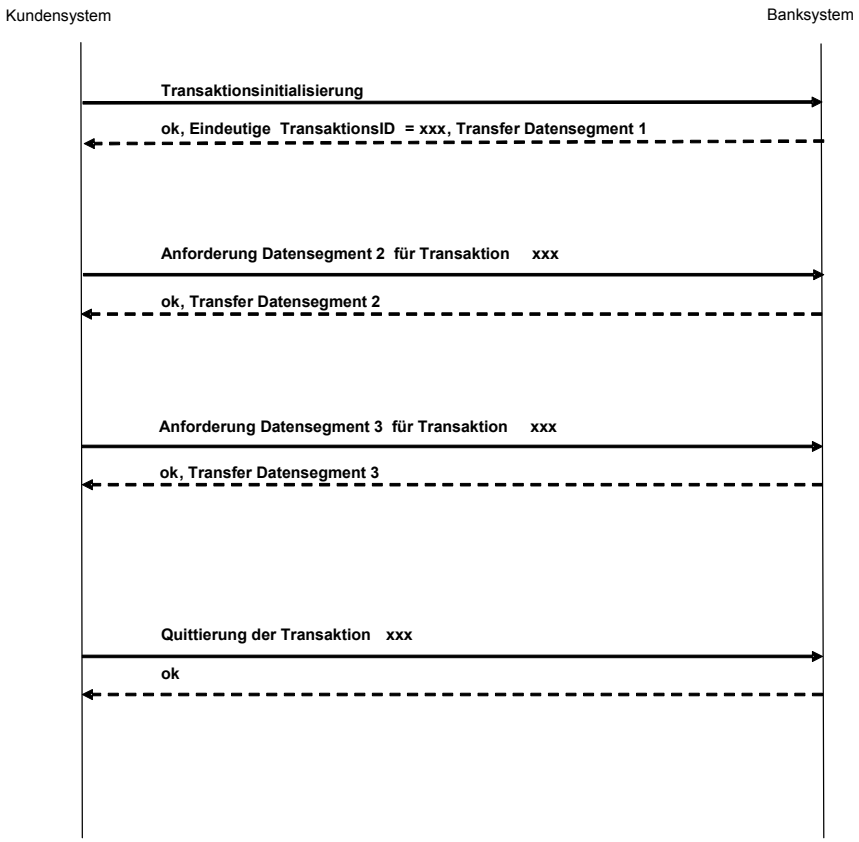


Abbildung 8: Beispielhafter Ablauf einer EBICS-Transaktion für einen Abholauftrag

Details zum Ablauf der EBICS-Transaktionen sind in Kapitel 5 enthalten.

4 Schlüsselmanagement

4.1 Übersicht der verwendeten Schlüssel

Das EBICS-Protokoll sieht für jeden Teilnehmer drei RSA-Schlüsselpaare vor. Diese werden für die folgenden Verwendungszwecke eingesetzt:

- Bankfachliche/technische EU der Auftragsdaten, die der Teilnehmer/Clientsystem an das Banksystem sendet
- Authentifikation des Teilnehmers gegenüber dem Banksystem mittels Authentifikationssignatur
- Entschlüsselung des (symmetrischen) Transaktionsschlüssels, der zur Verschlüsselung der Auftragsdaten eingesetzt wird, die der Teilnehmer vom Banksystem abruft.

Ihrem Verwendungszweck folgend spricht man dann auch von dem

- öffentlichen / privaten bankfachlichen Schlüssel
- öffentlichen / privaten Authentifikationsschlüssel
- öffentlichen / privaten Verschlüsselungsschlüssel.

EBICS ermöglicht die Verwendung von drei unterschiedlichen Schlüsselpaaren je Teilnehmer. EBICS fordert dabei die Verwendung von mindestens zwei unterschiedlichen Schlüsselpaaren je Teilnehmer:

- Ein Schlüsselpaar wird exklusiv für die bankfachliche elektronische Signatur verwendet
- Für die Authentifikation des Teilnehmers gegenüber dem Banksystem UND die Entschlüsselung von Transaktionsschlüsseln ist die Verwendung eines einzigen Schlüsselpaares erlaubt.

Analog zu den Teilnehmerschlüsseln sieht EBICS drei unterschiedliche RSA-Schlüsselpaare des Banksystems vor. Diese werden für die folgenden Verwendungszwecke eingesetzt:

- bankfachliche EU der Auftragsdaten, die ein Teilnehmer vom Banksystem abruft
  In der EBICS-Version „H004“ ist die bankfachliche EU der Kreditinstitute nur vorgesehen (siehe Kapitel 3.5.2)
- Authentifikation des Kreditinstituts gegenüber dem Teilnehmer mittels Authentifikationssignatur
- Entschlüsselung des (symmetrischen) Transaktionsschlüssels zur Verschlüsselung bankfachlicher Nutzdaten, die ein Teilnehmer an das Kreditinstitut sendet.

Bezüglich des Einsatzes eines RSA-Schlüsselpaares für unterschiedliche Zwecke gelten dieselben Einschränkungen wie für Teilnehmerschlüssel.

Die Schlüssel eines Teilnehmers sind an Verfahren gekoppelt, die der Teilnehmer für die Erzeugung/Verifikation der EU, für die Erzeugung/Verifikation der Authentifikationssignatur sowie für die Verschlüsselung von Auftragsdaten einsetzen möchte. Diese Verfahren werden durch eindeutige Versionen identifiziert, so dass unterschiedliche Teilnehmer beispielsweise unterschiedliche Verfahren für die EU verwenden können. EBICS setzt voraus, dass die jeweiligen Verfahren je Teilnehmer im Banksystem verwaltet werden.

Die Version „H004“ des EBICS-Protokolls sieht die Verwendung der folgenden Verfahren vor:

- „X002“ für die Authentifikationssignatur
- „A004“, „A005“ oder „A006“ für die EU
- „E002“ für die Verschlüsselung.

Details zu diesen Verfahren sind im Anhang (Kapitel 11.1, Kapitel 11.2, Kapitel 11.3) enthalten.

Teilnehmer desselben Kunden verwenden in der Regel dieselben Verfahren der Authentifikationssignatur, der Verschlüsselung sowie der EU.

Die Aufträge eines Teilnehmers können durch einen technischen Teilnehmer übermittelt werden, wenn beide Teilnehmer dieselben Verfahren der Authentifikationssignatur, der Verschlüsselung und der bankfachlichen Signatur einsetzen. Kundenseitig kann in diesem Fall die Verwaltung der öffentlichen Bankschlüssel für all die Teilnehmer zentralisiert werden, die mit denselben Verfahren arbeiten möchten.

4.2 Darstellung der öffentlichen Schlüssel

EBICS definiert die neuen Auftragsarten HIA, HCA, HSA, HCS und HPB, deren bankfachliche Nutzdaten öffentliche Schlüssel des Teilnehmers bzw. des Kreditinstituts sind. Für diese Auftragsarten erfolgt die Einbettung der öffentlichen Schlüssel in EBICS-Nachrichten unter Verwendung neu definierter Typen auf Basis von XML-Schema (siehe Schema-Definitionsdatei ebics_types_H004.xsd bzw. ebics_signature.xsd). Diese Typen sind in der nachfolgenden Tabelle enthalten:

Schlüsseltyp	**XML-Typ**
Authentfikitionsschlüssel	`AuthenticationPubKeyInfoType`
Bankfachlicher Schlüssel	`SignaturePubKeyInfoType`
Verschlüsselungsschlüssel	`EncryptionPubKeyInfoType`

Die graphische Darstellung der XML-Typen `AuthenticationPubKeyInfoType`, `SignaturePubKeyInfoType`, `EncryptionPubKeyInfoType` ist in Abbildung 9, Abbildung 10 und schließlich Abbildung 11 enthalten.

Die XML-Strukturen sind ähnlich zueinander aufgebaut: Sie enthalten den Wert des öffentlichen Schlüssels als Kombination aus Exponent und Modulus. Es können zusätzlich Informationen zu einem X509-Zertifikat geliefert werden. Bestandteil des Authentifikationsschlüssels ist zudem die Version des Verfahrens zur Bildung/ Verifikation der Authentifikationssignatur (siehe Element `AuthenticationVersion`). Analog dazu ist die Version des Verschlüsselungsverfahrens Bestandteil des Verschlüsselungsschlüssels sowie die Version der bankfachlichen EU Bestandteil des bankfachlichen Schlüssels.

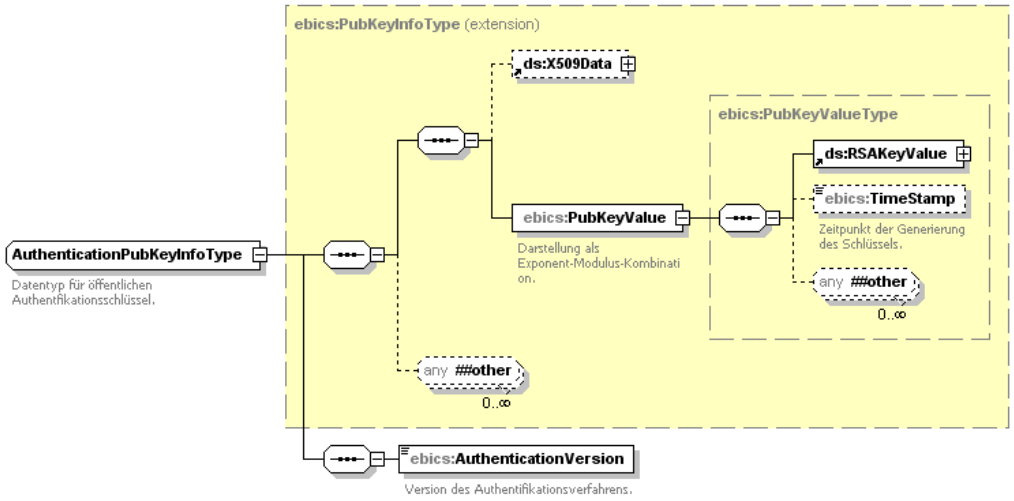


Abbildung 9: Definition des XML-Schema-Typs AuthenticationPubKeyInfoType

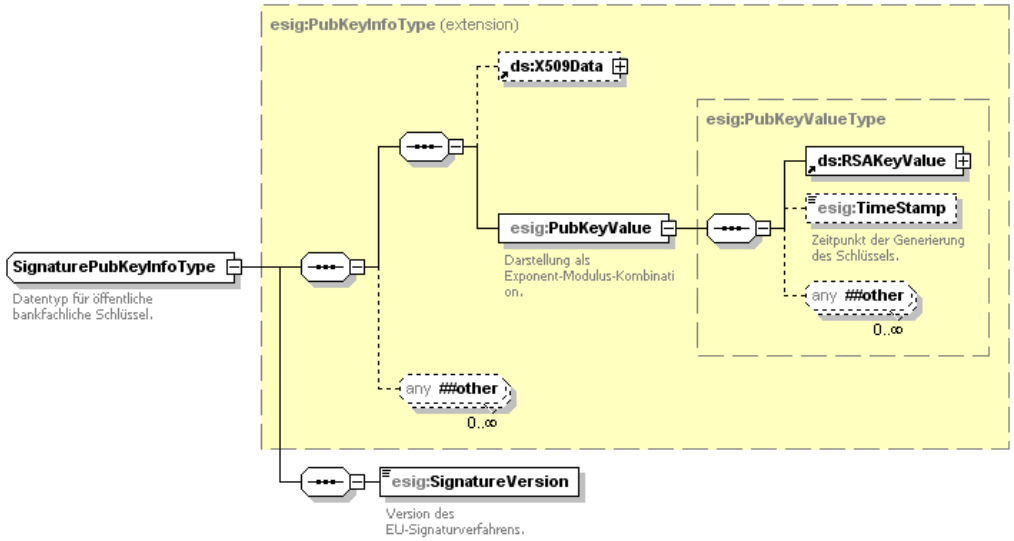


Abbildung 10: Definition des XML-Schema-Typs SignaturePubKeyInfoType

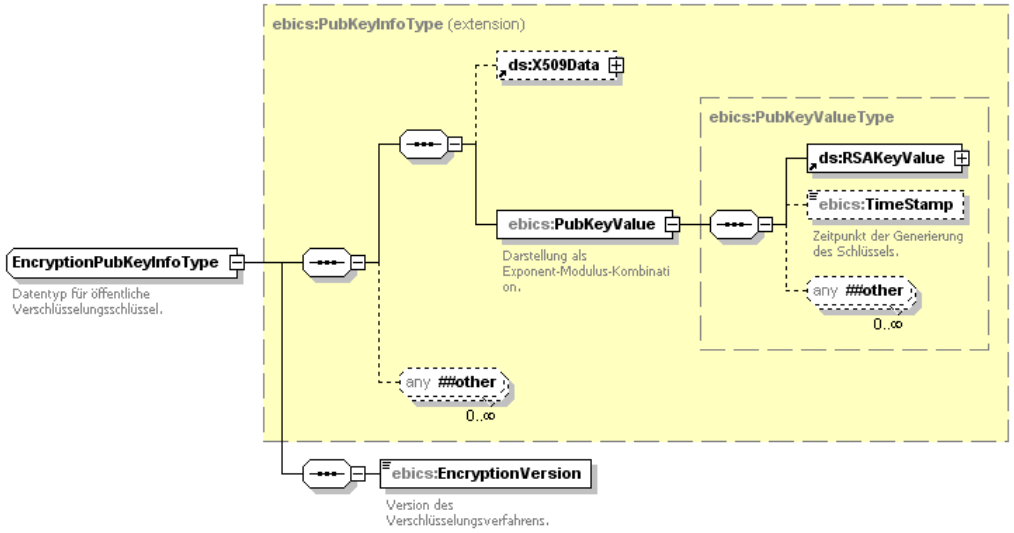


Abbildung 11: Definition des XML-Schema-Typs EncryptionPubKeyInfoType

4.3 Aktionen innerhalb des Schlüsselmanagements

Die eigentliche Durchführung der Sendeaufträge des Schlüsselmanagements muss synchron zu deren Übertragung über EBICS stattfinden. Die Durchführung muss somit abgeschlossen sein, bevor der letzte EBICS-Response dieser Übertragung an den Teilnehmer gesendet wird.

Diese Forderung gilt für INI, HIA und H3K (siehe Kapitel 4.4.1) sowie HSA (siehe Kapitel 4.8), um die Durchführung der Teilnehmerinitialisierung nicht unnötig zu verzögern. Sie gilt gleichermaßen für SPR (siehe Kapitel 4.5), um die Sperre des Teilnehmers sofort zu aktivieren. Nachträglich initiierte EBICS-Transaktionen für die Übermittlung eines bankfachlichen Auftrags werden auf Ebene des EBICS-Protokolls solange abgewiesen, bis der Teilnehmer wieder den Zustand „Bereit“ erreicht hat. (Vor dem Zeitpunkt der Sperrung erfolgreich verifizierte EUs des Teilnehmers bleiben auch nach der Sperrung gültig. Eine solche EU kann für die Autorisierung eines offenen Auftrags im Rahmen der VEU weiterhin verwendet werden).

Schließlich gilt diese Forderung auch für PUB, HCS und HCA (siehe Kapitel 4.6.1), um unmittelbar darauf folgende EBICS-Transaktionen des betroffenen Teilnehmers, die die aktualisierten Schlüssel bereits einsetzen, auch erfolgreich durchführen zu können. (Vor dem Zeitpunkt der Durchführung von PUB bzw. HCS erfolgreich verifizierte EUs des Teilnehmers bleiben auch nach der Durchführung von PUB bzw. HCS und der damit verbundenen Änderung des bankfachlichen Teilnehmerschlüssels gültig. Eine solche EU kann für die Autorisierung eines offenen Auftrags im Rahmen der VEU weiterhin verwendet werden).

4.4 Initialisierung

Um bankfachliche EBICS-Transaktionen mit einem bestimmten Kreditinstitut durchführen zu können, müssen für die Teilnehmer eines Kunden eine Reihe von Voraussetzungen erfüllt sein.

Grundvoraussetzung ist der Abschluss eines Vertrags zwischen Kunde und Kreditinstitut. In diesem Vertrag wird vereinbart, welche Geschäftsvorfälle (bankfachliche Auftragsarten) der Kunde mit dem Kreditinstitut abwickelt, welche Konten betroffen sind, welche Teilnehmer des Kunden mit dem System arbeiten und welche Berechtigungen diese Teilnehmer besitzen.

Sofern der Kunde noch nicht über ein entsprechendes Kundenprodukt verfügt, erhält er nach Abschluss des Vertrags die Client-Software sowie die Zugangsdaten des Kreditinstituts (Bankparameter). Das Kreditinstitut richtet die Kunden- und Teilnehmerstammdaten gemäß den vertraglichen Vereinbarungen im Banksystem ein. Der einzelne Teilnehmer erhält hierdurch den Status „**Neu**“.

Details der vertraglichen Vereinbarungen sind nicht Gegenstand dieses Standards, sie sind bankindividuell in der Beziehung Kunde-Kreditinstitut zu regeln.

Weitere Voraussetzungen sind die erfolgreiche Teilnehmerinitialisierung und Abholung der öffentlichen Schlüssel des Kreditinstituts durch den Teilnehmer. Die notwendigen Schritte auf Kreditinstitut-, Kunden- und Teilnehmerseite sowie die zeitlichen Abhängigkeiten dieser Schritte sind in Abbildung 12 enthalten. Abbildung 13 zeigt einen beipielhaften Ablauf anhand eines Sequenzdiagramms. Der Zustand der öffentlichen Bankschlüssel auf Teilnehmer-

seite ist auf der Lebenslinie des Teilnehmersystems dargestellt. Entsprechend sind der Zustand der öffentlichen Teilnehmerschlüssel auf Bankseite sowie der Zustand des Teilnehmers selbst auf der Lebenslinie des Banksystems dargestellt. Details dieser Abbildungen werden in den nachfolgenden Kapiteln erklärt.

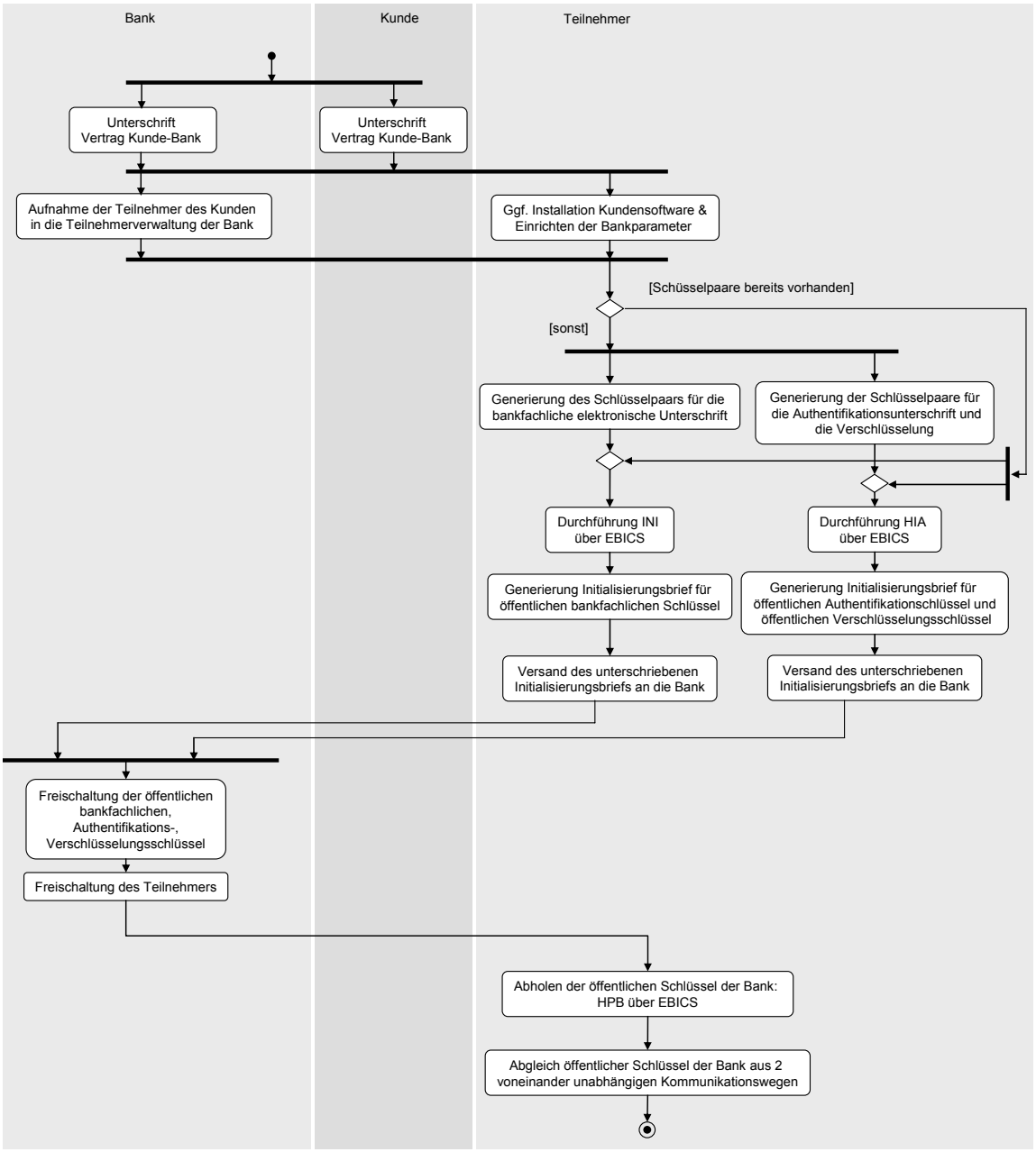


Abbildung 12: Notwendige Schritte im Vorfeld der eigentlichen Durchführung von Geschäftsvorfällen über EBICS (wenn mit INI/HIA initialisiert wird)

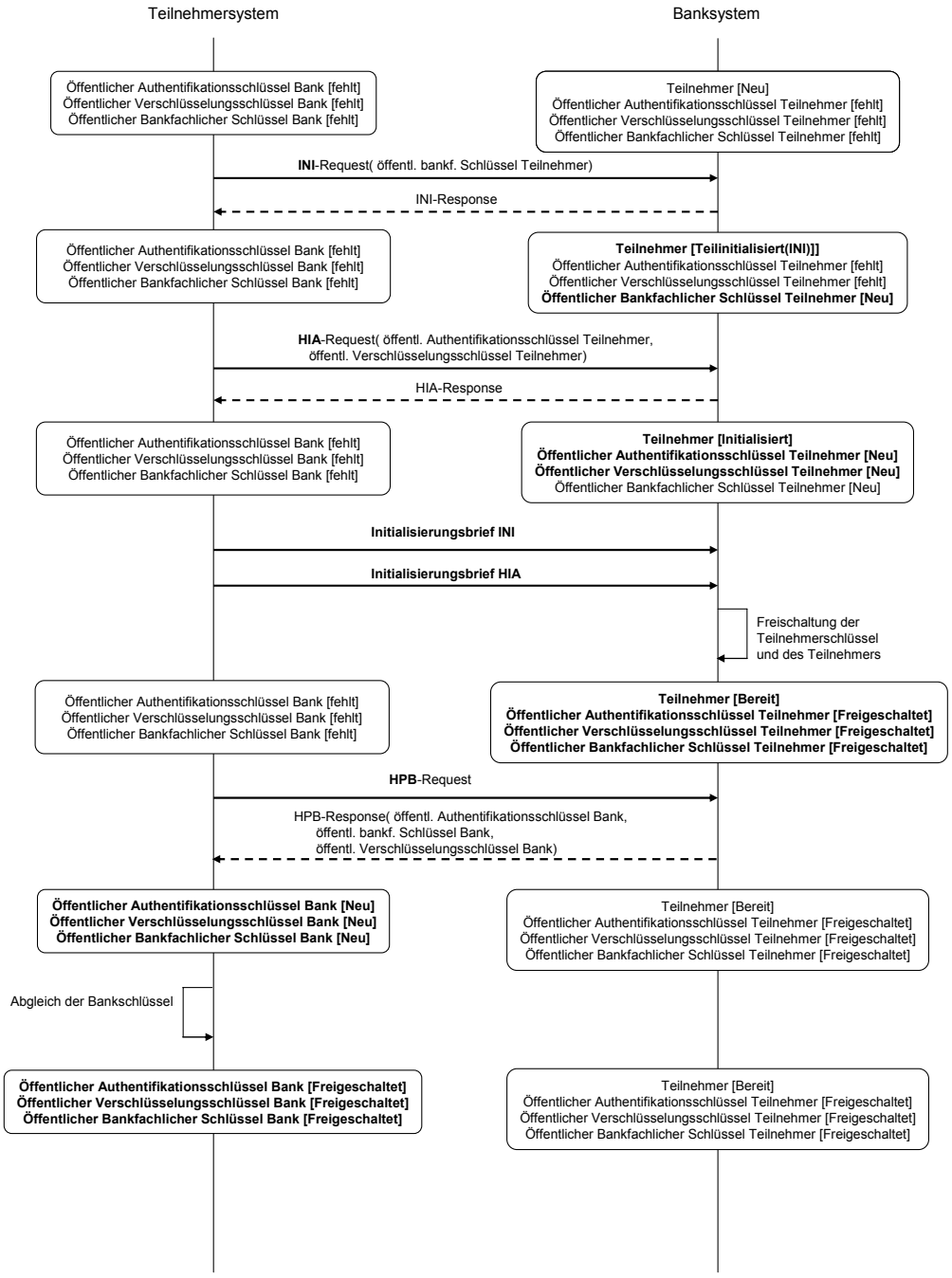


Abbildung 13: Beispielablauf: Teilnehmerinitialisierung mit anschließender Abholung und Überprüfung der Bankschlüssel (wenn mit INI/HIA initialisiert wird)

4.4.1 Teilnehmerinitialisierung

4.4.1.1 Allgemeine Beschreibung

Für die Initialisierung des Teilnehmers beim Kreditinstitut ist die Übermittlung der öffentlichen Schlüssel des Teilnehmers an das Banksystem erforderlich. Bestandteil der Bankparameter sind die unterstützten Versionen für die EU, die Verschlüsselung und die Authentifikationssignatur. Der bankfachliche Schlüssel des Teilnehmers muss neu generiert werden, wenn der Teilnehmer nicht über einen passenden bankfachlichen Schlüssel verfügt oder einen bereits vorhandenen bankfachlichen Schlüssel nicht für die neue Bankverbindung nutzen möchte. Das gleiche gilt für den Verschlüsselungsschlüssel und den Authentifikationsschlüssel.

Der Teilnehmer übermittelt dem Kreditinstitut seine öffentlichen Schlüssel wie folgt

Alternative 1: (auf zwei voneinander unabhängigen Kommunikationswegen):

- über EBICS mittels folgender systembedingter Auftragsarten:
- **INI**: Senden des öffentlichen bankfachlichen Schlüssels (Schlüssel für Autorisierung/EU)
- **HIA**: Senden des öffentlichen Authentifikationsschlüssels sowie des öffentlichen Verschlüsselungsschlüssels.

Die Übermittlung der öffentlichen Teilnehmerschlüssel an das Kreditinstitut über INI und HIA wird als **Teilnehmerinitialisierung** bezeichnet

- auf dem Postweg mit vom Teilnehmer unterschriebenen Initialisierungsbriefen.

Die Verwendung unterschriebener Initialisierungsbriefe ermöglicht dem Kreditinstitut:

- die Überprüfung der Authentizität der über EBICS übermittelten öffentlichen Teilnehmerschlüssel als Voraussetzung für die Freischaltung von Teilnehmern
- die Nachvollziehbarkeit der Schlüssel-Historie von Teilnehmern durch Aufbewahren der Initialisierungsbriefe.

Die Reihenfolge der Durchführung von INI und HIA ist nicht festgelegt, im Rahmen einer Teilnehmerinitialisierung werden jedoch genau ein INI-Auftrag und genau ein HIA-Auftrag durchgeführt. Die Übermittlung der öffentlichen Teilnehmerschlüssel über zwei separate Aufträge in beliebiger Reihenfolge erfordert die Definition der Teilnehmerzustände **„Teilweise initialisiert(INI)“** und **„Teilweise initialisiert(HIA)“**. In Abhängigkeit davon, ob im Rahmen der Teilnehmerinitialisierung der erste erfolgreich durchgeführte Auftrag INI oder HIA ist, nimmt der Teilnehmer den entsprechenden Zustand ein.

Eine Initialisierung mit Alternative 1 ist immer zulässig, egal, ob es für die RSA Schlüssel Zertifikate gibt oder nicht. Allerdings müssen bei Alternative 1 immer Initialisierungsbriefe verwendet warden. Die öffentlichen Teilnehmerschlüssel müssen dann mit den Auftragsarten INI (öffentlicher bankfachlicher Schlüssel, also Schlüssel für Autorisierung/EU) und HIA (öffentliche Schlüssel für Authentifikation und Verschlüsselung) zu Bank geschickt werden.

Um die Authentizität der Teilnehmerschlüssel zu garantieren, muss sichergestellt sein, dass die Bank diese Schlüssel über einen zweiten, davon unabhängigen, Kommunikationsweg erhält (Initialisierungsbriefe sowohl für INI als auch für HIA). Hat die Bank die Schlüssel über diese beiden unabhängigen Kommunikationswege erhalten, vergleicht sie diese zuerst, bevor sie sie anerkennt.
Alternative 1 kann insbesondere genutzt werden, wenn Alternative 2 fehlschlägt.
Details und Workflow zur Alternative 1 sind in Kapitel 4.4.1.2 beschrieben.

Alternative 2 (Initialisierung in einem Schritt):
Diese Alternative ist nur möglich, wenn Zertifikate verwendet werden und auch dann nur unter bestimmten Bedingungen: Nur wenn der bankfachliche (Autorisierungs-) Schlüssel auf einem Zertifikat basiert, das von einer Zertifizierungsinstanz (CA) ausgegeben wurde, die Authentizität dieses speziellen Schlüssels von der CA garantiert wird und Kunde und Bank vereinbart haben, dieses Zertifikat zu verwenden und so lange dieses auch gültig ist.
Dann ist der Initialisierungsbrief nicht notwendig.

- via EBICS mittels der Auftragsart:
  - **H3K**: Übermittlung der öffentlichen Teilnehmerschlüssel für die bankfachliche EU (Autorisierung), für Authentifikation und Verschlüsselung.
  -

Für die Initialisierung via Zertifikat vereeinfacht die Auftragsart H3K den Ablauf:

1. Alle öffentlichen Teilnehmerschlüssel können in einem Schritt übermittelt werden (H3K-Request) das von einer CA ausgegebene Zertifikat für den bankfachlichen EU (Autorisierungs-) Schlüssel bereits für eine Elektronische Unterschrift nutzend.
2. INI- und HIA-Brief sind daher nicht notwendig.

Details und Workflow siehe Kapitel 4.4.1.3.

4.4.1.2 Initialisierung via INI und HIA

4.4.1.2.1 INI

Die Durchführung von INI ist zulässig, wenn der Zustand des betroffenen Teilnehmers „Neu“, „Gesperrt“ oder „Teilweise Initialisiert(HIA) ist. INI besteht aus einem einzigen EBICS-Request/-Response-Paar. Für den EBICS-Request von INI gilt:

- er erfordert keine Authentifikationssignatur, da der öffentiche Authentifikationsschlüssel des Teilnehmers vom Kreditinstitut noch nicht freigeschaltet wurde und damit nicht zur Prüfung verwendet werden kann
- er enthält keine bankfachliche Unterschrift, da der öffentliche bankfachliche Schlüssel des Teilnehmers in diesem Request erst übertragen wird. Dieser öffentliche bankfachliche Schlüssel des Teilnehmers kann vom Kreditinstitut nicht zur Prüfung der bankfachlichen Unterschrift verwendet werden, da seine Authentizität zu diesem Zeitpunkt noch nicht gesichert ist

- er enthält die Nutzdaten, d.h. den öffentlichen bankfachlichen Schlüsel des Teilnehmers in unverschlüsselter Form, da der öffentliche Verschlüsselungsschlüssel des Kreditinstituts dem Teilnehmer (zumindest bei der ersten Initialisierung) noch nicht vorliegt.

Das Ablaufdiagramm in Abbildung 14 stellt die bankseitige Verarbeitung dar, die beim Empfang eines INI-Requests stattfindet. Fehlersituationen, die aus einer ungültigen Kombination aus Kunden-/TeilnehmerID oder einem unzulässigen Teilnehmerzustand resultieren, werden nicht direkt an den Sender des INI-Requests durchgereicht. Stattdessen erhält der Sender den technischen Fehlercode EBICS_INVALID_USER_OR_USER_STATE. INI liefert keine Fehler der Art „Unbekannter Teilnehmer“ oder „Unzulässiger Teilnehmerzustand“, um potenziellen Angreifern keine genaue Information über die Gültigkeit von TeilnehmerIDs oder den Zustand von Teilnehmern zu geben. Hingegen muss auf Seiten des Kreditinstituts eine interne Protokollierung erfolgen, die die genaue Fehlerursache dokumentiert.

Das Ablaufdiagramm sieht die Überprüfung des Teilnehmerzustands vor, um INI-Requests bereits auf EBICS-Ebene abzuweisen, wenn der Teilnehmerzustand für INI nicht zulässig ist. Zulässige Zustände für INI sind: „Neu“, „Gesperrt“ sowie „Teilweise initialisiert(HIA)“. Geprüft wird hier der Zustand des Teilnehmers aus den Headerdaten des Requests. Die Nutzdaten des INI-Requests (siehe Kapitel 4.4.1.2.5.1) enthalten letztlich den Teilnehmer, dessen bankfachlicher Schlüssel übermittelt werden soll. Der Teilnehmer aus den Headerdaten sollte deshalb mit dem Teilnehmer aus den Nutzdaten übereinstimmen. Das EBICS-Protokoll sieht keine Überprüfung dieser Übereinstimmung vor. Vor der eigentlichen Durchführung des Auftrags wird jedoch der Zustand des Teilnehmers (erneut) überprüft, welcher Teil der Nutzdaten von INI ist.

Die Durchführung eines INI-Auftrags kann die folgenden Fehlercodes liefern:

- EBICS_KEYMGMT_UNSUPPORTED_VERSION_SIGNATURE
Dieser fachliche Fehler tritt dann auf, wenn die Nutzdaten eine unzulässige Version des Verfahrens zur bankfachlichen Signatur enthalten
- EBICS_KEYMGMT_KEYLENGTH_ERROR_SIGNATURE
Dieser fachliche Fehler tritt dann auf, wenn die Nutzdaten einen bankfachlichen Schlüssel unzulässiger Länge enthalten
- EBICS_INVALID_ORDER_DATA_FORMAT
Dieser fachliche Fehler tritt dann auf, wenn die Nutzdaten nicht dem dafür vorgesehenen Format entsprechen (siehe Kapitel 4.4.1.2.5.1)
- EBICS_INVALID_USER_OR_USER_STATE
Dieser technische Fehler tritt dann auf, wenn die Nutzdaten einen Teilnehmer enthalten, der entweder unbekannt oder dessen Zustand für INI unzulässig ist. Zulässig sind die folgenden Teilnehmerzustände: Neu, Gesperrt, Teilweise Initialisiert(HIA).

- Für Return Codes, die Zertifikate betreffen, wird auf Anhang 1 verwiesen.

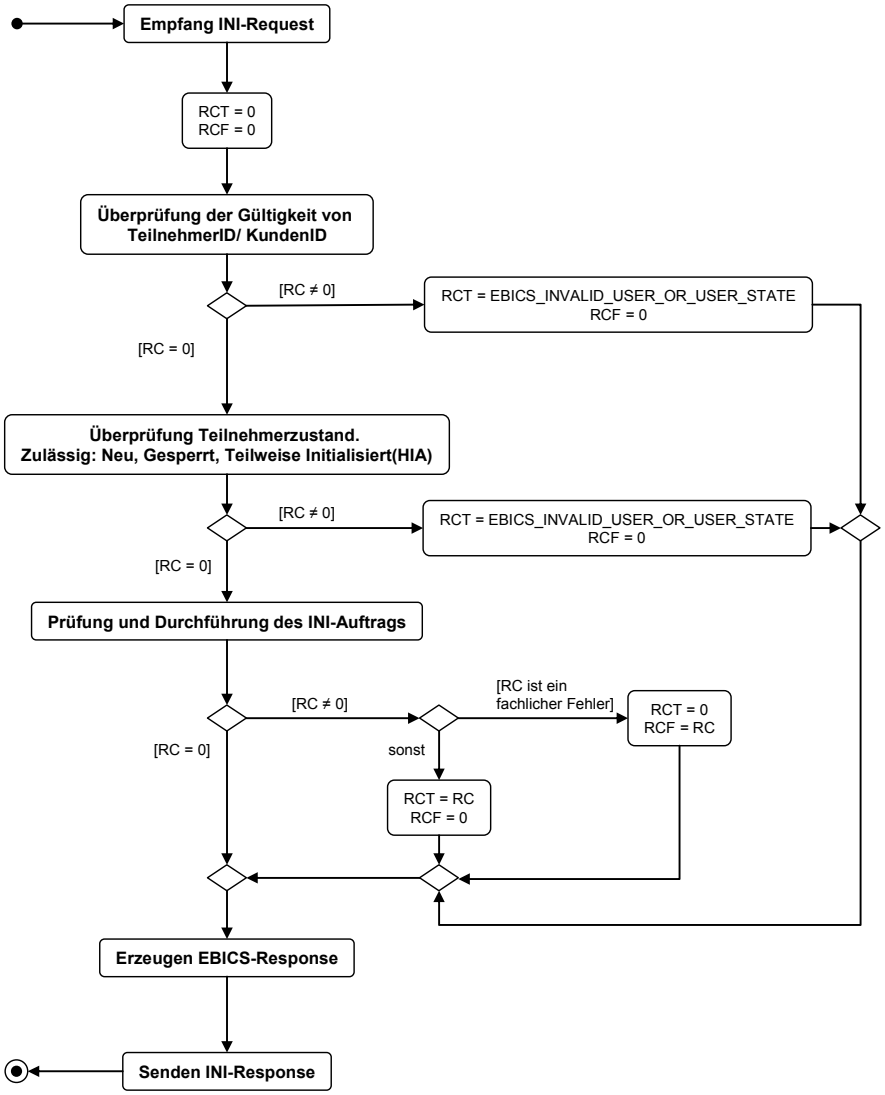


Abbildung 14: Bankseitige Verarbeitung eines INI-Requests

Die EBICS-Response für INI enthält keine Authentifikationssignatur des Kreditinstituts, da der öffentliche Authentifikationsschlüssel des Kreditinstituts dem Teilnehmer zur Prüfung noch nicht vorliegt.

In Abbildung 13 wird INI vor HIA durchgeführt, und dementsprechend gelangt der Teilnehmer vom Zustand „Neu“ in den Zustand „Teilweise initialisiert(INI)“. Der Zustand „Teilweise initialisiert(INI)“ bedeutet:

- das Banksystem verfügt über den öffentlichen bankfachlichen Schlüssel des Teilnehmers, den es allerdings noch nicht freigeschaltet hat
- das Banksystem verfügt (noch) nicht über den öffentlichen Authentifikationsschlüssel sowie den öffentlichen Verschlüsselungsschlüssel des Teilnehmers.

In diesem Zustand kann der Teilnehmer nur eine der beiden folgenden Aktionen durchführen:

- die Auftragsart HIA durchführen und danach in den Zustand „Initialisiert“ übergehen: Aufträge ungleich HIA, die der Teilnehmer in diesem Zustand einreicht, werden vom Banksystem zurückgewiesen. Bankfachliche Unterschriften des Teilnehmers zu bereits vorhandenen Aufträgen werden als ungültig gewertet, wenn sich der Teilnehmer zum Zeitpunkt der Überprüfung im Zustand „Teilweise initialisiert(INI)“ befindet.
- sich per Anruf beim Kreditinstitut sperren lassen:
Danach ist nur die erneute Initialisierung des Teilnehmers möglich.

Nach der erfolgreichen Durchführung von INI verschickt der Teilnehmer einen unterschriebenen Initialisierungsbrief für INI. Für Einzelheiten zum Inhalt des Initialisierungsbriefs siehe Kapitel 4.4.1.2.3..

4.4.1.2.2 HIA

Die Durchführung von HIA ist zulässig, wenn der Zustand des Teilnehmers „Neu“, „Gesperrt“ oder „Teilweise Initialisiert(INI)“ ist. HIA besteht aus einem einzigen EBICS-Request/ -Response-Paar. Für den EBICS-Request von HIA gilt:

- er enthält keine Authentifikationssignatur, da der öffentliche Authentifikationsschlüssel des Teilnehmers in diesem Request erst übertragen wird. Dieser öffentliche Authentifikationsschlüssel des Teilnehmers kann vom Kreditinstitut nicht zur Prüfung der Authentifikationsunterschrift verwendet werden, da seine Authentizität zu diesem Zeitpunkt noch nicht gesichert ist
- er enthält keine bankfachliche Unterschrift, da der öffentliche bankfachliche Schlüssel des Teilnehmers vom Kreditinstitut noch nicht freigeschaltet wurde und damit nicht zur Prüfung verwendet werden kann
- er enthält die Nutzdaten, d.h. den öffentlichen Verschlüsselungsschlüssel und den öffentlichen Authentifikationsschlüssel des Teilnehmers, in unverschlüsselter Form, da der öffentliche Verschlüsselungsschlüssel des Kreditinstituts dem Teilnehmer (zumindest bei der ersten Initialisierung) noch nicht vorliegt.

Das Ablaufdiagramm in Abbildung 15 stellt die bankseitige Verarbeitung dar, die beim Empfang eines HIA-Requests stattfindet. Analog zu INI werden auch hier Fehlersituationen, die aus einer ungültigen Kombination aus Kunden-/TeilnehmerID oder einem unzulässigen Teilnehmerzustand resultieren, nicht direkt an den Sender des HIA-Requests durchgereicht. Stattdessen erhält der Sender den technischen Fehlercode

EBICS_INVALID_USER_OR_USER_STATE. HIA liefert keine Fehler der Art „Unbekannter Teilnehmer“ oder „Unzulässiger Teilnehmerzustand“, um potenziellen Angreifern keine genaue Information über die Gültigkeit von TeilnehmerIDs oder den Zustand von Teilnehmern zu geben. Ebenfalls analog zu INI muss auf Seiten des Kreditinstituts eine interne Protokollierung erfolgen, die die genaue Fehlerursache dokumentiert.

Das Ablaufdiagramm sieht die Überprüfung des Teilnehmerzustands vor, um HIA-Requests bereits auf EBICS-Ebene abzuweisen, wenn der Teilnehmerzustand für HIA nicht zulässig ist. Zulässige Zustände für HIA sind: „Neu“, „Gesperrt“ sowie „Teilweise initialisiert(INI)“. Geprüft wird hier der Zustand des Teilnehmers aus den Headerdaten des Requests. Die Nutzdaten des HIA-Requests (siehe Kapitel 4.4.1.2.5.1) enthalten letztlich den Teilnehmer, dessen Authentifikations- und Verschlüsselungsschlüssel übermittelt werden soll. Der Teilnehmer aus den Headerdaten sollte deshalb mit dem Teilnehmer aus den Nutzdaten übereinstimmen. Das EBICS-Protokoll sieht keine Überprüfung dieser Übereinstimmung vor. Vor der eigentlichen Durchführung des Auftrags wird jedoch der Zustand des Teilnehmers (erneut) überprüft, welcher Teil der Nutzdaten von HIA ist.

Die Durchführung eines HIA-Auftrags kann die folgenden Fehlercodes liefern:

- EBICS_KEYMGMT_UNSUPPORTED_VERSION_ENCRYPTION
  Dieser fachliche Fehler tritt dann auf, wenn die Nutzdaten eine unzulässige Version des Verschlüsselungsverfahrens enthalten
- EBICS_KEYMGMT_UNSUPPORTED_VERSION_AUTHENTICATION
  Dieser fachliche Fehler tritt dann auf, wenn die Nutzdaten eine unzulässige Version des Verfahrens der Authentifikationssignatur enthalten
- EBICS_KEYMGMT_KEY_LENGTH_ERROR_ENCRYPTION
  Dieser fachliche Fehler tritt dann auf, wenn die Nutzdaten einen Verschlüsselungsschlüssel unzulässiger Länge enthalten
- EBICS_KEYMGMT_KEY_LENGTH_ERROR_AUTHENTICATION
  Dieser fachliche Fehler tritt dann auf, wenn die Nutzdaten einen Authentifikationsschlüssel unzulässiger Länge enthalten
- EBICS_INVALID_ORDER_DATA_FORMAT
  Dieser fachliche Fehler tritt dann auf, wenn die Nutzdaten nicht dem dafür vorgesehenen Format entsprechen (siehe Kapitel 4.4.1.2.5.1)
- EBICS_INVALID_USER_OR_USER_STATE
  Dieser technische Fehler tritt dann auf, wenn die Nutzdaten einen Teilnehmer enthalten, der entweder nicht gültig ist oder dessen Zustand für HIA unzulässig ist. Zulässig sind die folgenden Teilnehmerzustände: Neu, Gesperrt, Teilweise Initialisiert(INI)
- EBICS_KEYMGMT_NO_X509_SUPPORT
  Dieser fachliche Fehler tritt auf, wenn ein öffentlicher Schlüssel vom Typ `ds:X509Data` übertragen wurde, das Kreditinstitut jedoch nur den Typ `ebics:PubKeyValueType` unterstützt.

- Für Return Codes, die Zertifikate betreffen, wird auf Anhang 1 verwiesen.

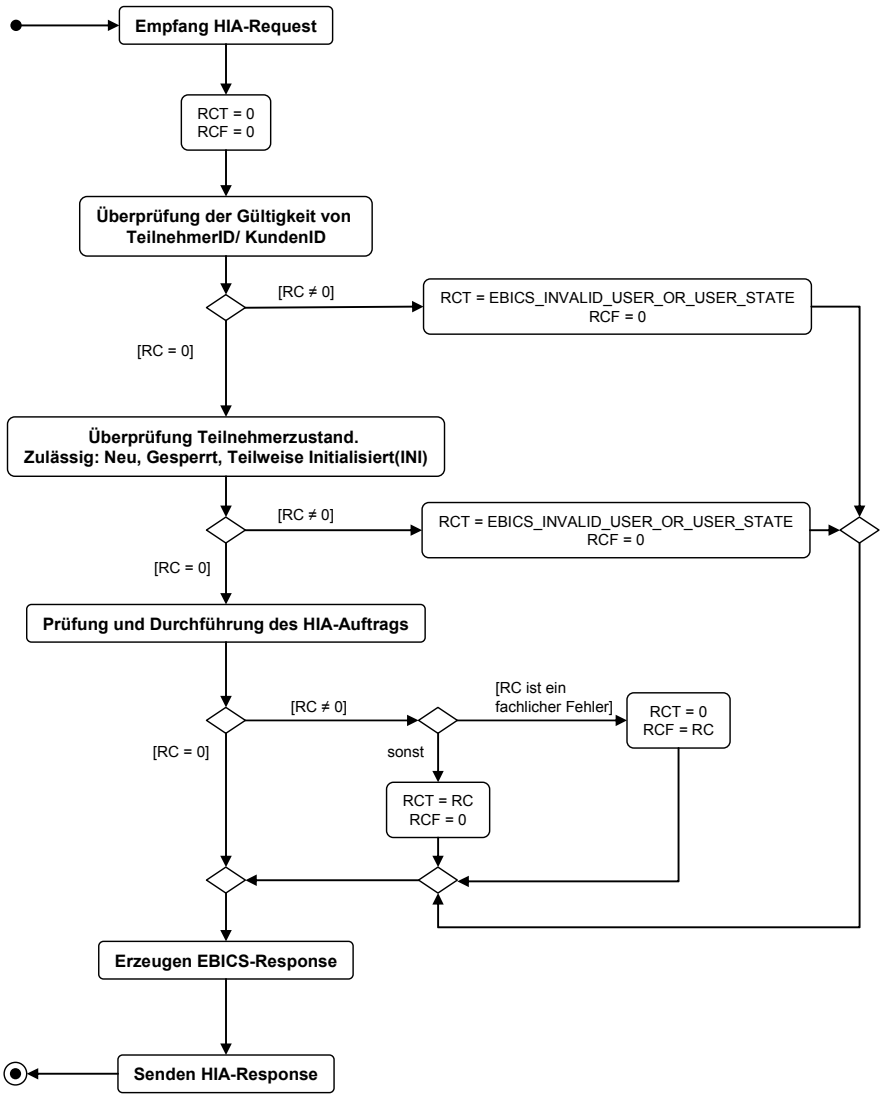


Abbildung 15: Bankseitige Verarbeitung eines HIA-Requests

Die EBICS-Response für HIA enthält keine Authentifikationssignatur des Kreditinstituts, da der öffentliche Authentifikationsschlüssel des Kreditinstituts dem Teilnehmer zur Prüfung noch nicht vorliegt.

Die Bedeutung des Zustands „Teilweise initialisiert(HIA)“, welcher in Abbildung 13 nicht berücksichtigt wurde, ist die folgende:

- Das Banksystem verfügt über den öffentlichen Authentifikationsschlüssel sowie den öffentlichen Verschlüsselungsschlüssel des Teilnehmers. Beide wurden noch nicht durch das Banksystem freigeschaltet
- Das Banksystem verfügt (noch) nicht über den öffentlichen bankfachlichen Schlüssel des Teilnehmers.

In diesem Zustand kann der Teilnehmer nur eine der beiden folgenden Aktionen durchführen:

- die Auftragsart INI durchführen:
Aufträge ungleich INI, die der Teilnehmer in diesem Zustand einreicht, werden vom Banksystem zurückgewiesen. Bankfachliche Unterschriften des Teilnehmers zu bereits vorhandenen Aufträgen werden als ungültig gewertet, wenn sich der Teilnehmer zum Zeitpunkt der Überprüfung im Zustand „Teilweise initialisiert(HIA)“ befindet
- sich per Anruf beim Kreditinstitut sperren lassen:
Danach ist lediglich eine erneute Inititialisierung des Teilnehmers möglich.

Nach der erfolgreichen Durchführung von HIA verschickt der Teilnehmer einen unterschriebenen Initialisierungsbrief für HIA an das Kreditinstitut. Für Einzelheiten zum Inhalt des Initialisierungsbriefes siehe Kapitel 4.4.1.2.3..

4.4.1.2.3 Initialisierungsbriefe

Initialisierungsbriefe für INI enthalten den öffentlichen bankfachlichen Teilnehmerschlüssel, Initialisierungsbriefe für HIA den öffentlichen Authentifikationsschlüssel des Teilnehmers sowie den öffentlichen Verschlüsselungsschlüssel des Teilnehmers. Neben den öffentlichen Teilnehmerschlüssel enthalten die Initialisierungsbriefe die folgenden Daten:

- Benutzername (optional): kundensoftware-interner Name des Teilnehmers
- Datum: Datum der Durchführung des entsprechenden EBICS-Auftrags
- Uhrzeit: Uhrzeit der Durchführung des entsprechenden EBICS-Auftrags
- Empfänger-Bank
- Teilnehmer-ID
- Kunden-ID.

Zu jedem öffentlichen Teilnehmerschlüssel enthält der Initialisierungsbrief die folgenden Daten:

- Verwendungszweck des öffentlichen Teilnehmerschlüssels:
  bankfachliche elektronische Unterschrift
  Authentifikationssignatur

Verschlüsselung.

- Verfahren:
  Verfahren der bankfachlichen elektronischen Unterschrift: A004, A005 oder A006
  Verfahren der Authentifikationssignatur: X002
  Verfahren der Verschlüsselung: E002.
- Längenangabe des Exponenten
- Exponent des öffentlichen Schlüssels in hexadezimaler Darstellung
- Längenangabe des Modulus
- Modulus des öffentlichen Schlüssels in hexadezimaler Darstellung
- Hashwert der öffentlichen Schlüssel in hexadezimaler Darstellung:
- Der Initialisierungsbrief für INI enthält den RIPEMD-160-Hashwert (im Falle von A004) bzw. SHA-256 (im Falle von A005 / A006) des öffentlichen bankfachlichen Schlüssels. Die Bildung des Hashwerts für die verschiedenen Unterschriftsversionen wird im Kapitel 14 beschrieben.

Der Initialisierungsbrief für HIA enthält den SHA-256-Hashwert des öffentlichen Authentifikationsschlüssels sowie den SHA-256-Hashwert des öffentlichen Verschlüsselungsschlüssels.
Die SHA-256-Hashwerte der öffentlichen X002 und E002 Schlüssel sowie für A005 und A006 Schlüssel werden über die Konkatenation des Exponenten mit einem Leerzeichen und dem Modulus in hexadezimaler Darstellung (mit Kleinbuchstaben) ohne führende Nullen (bezogen auf die hexadezimale Darstellung) gebildet. Dieser String muss basierend auf US-ASCII-Codierung in ein Byte-Array umgewandelt werden.

Initialisierungsbriefe für INI enthalten den öffentlichen Signaturschlüssel des Teilnehmers, Initialisierungsbriefe für HIA den öffentlichen Authentifikationsschlüssel des Teilnehmers sowie den öffentlichen Verschlüsselungsschlüssel des Teilnehmers. Beispiele für Initialisierungsbriefe sind im Anhang (Kapitel 11.5.1) enthalten.

4.4.1.2.4 Freischaltung des Teilnehmers durch das Kreditinstitut

Nach erfolgreicher Durchführung von INI und HIA befindet sich der Teilnehmer zunächst im Zustand **„Initialisiert“:** das Banksystem verfügt über alle notwendigen öffentlichen Schlüssel des Teilnehmers, die es allerdings noch nicht freigeschaltet hat. Teilnehmer im Zustand “Initialisiert” können keine Aufträge oder Unterschriften über EBICS einreichen: jeder Versuch wird vom Banksystem zurückgewiesen.

Nach erfolgreicher Prüfung der Initialisierungsbriefe auf Seiten des Kreditinstituts werden die öffentlichen Teilnehmerschlüssel freigegeben und der Zustand des Teilnehmers im Banksystem auf **„Bereit“** gesetzt. Der Zustand „Bereit“ bedeutet, dass die Bankseite über alle notwendigen Informationen verfügt, um Einreichungen von Aufträgen oder Unterschriften

durch den Teilnehmer erfolgreich durchführen zu können. Siehe dazu auch Abbildung 13. Insbesondere kann der Teilnehmer die sogenannten Bankparameter des Kreditinstituts über die Auftragsart HPD abholen (siehe Kapitel 9.2).

Das Diagramm in *Abbildung 16* verdeutlicht noch einmal die bisher beschriebenen Zustandsübergänge eines Teilnehmers. Das Löschen von Teilnehmern aus den Teilnehmerverwaltungen der Banksysteme ist nicht Gegenstand dieses Standards. Aus diesem Grund wird auf die Darstellung eines weiteren Zustands „Gelöscht“ sowie eines Endzustands verzichtet.

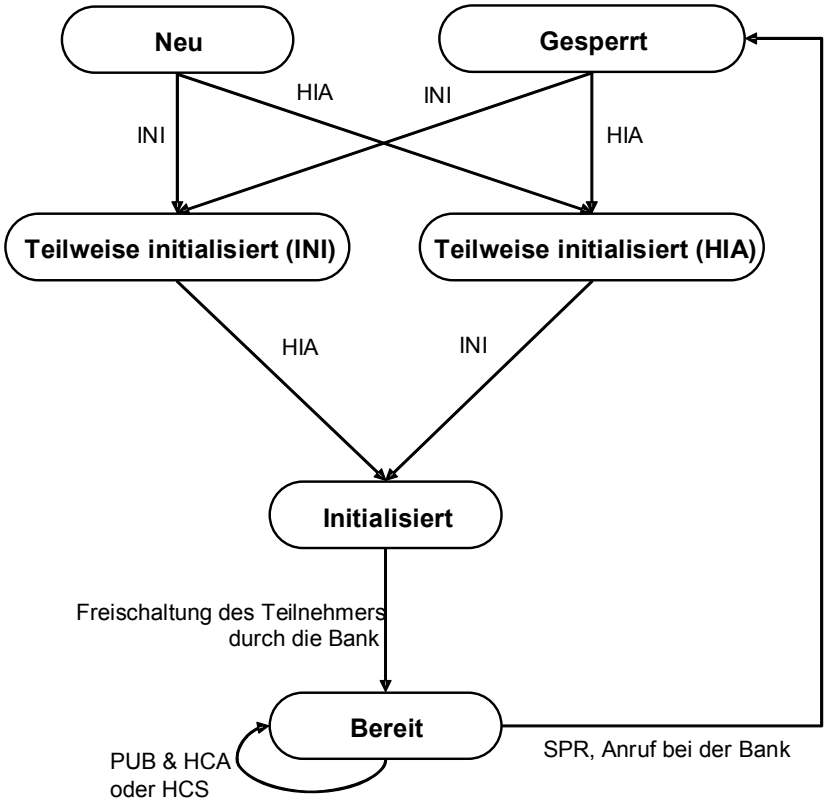


Abbildung 16: Zustandsübergangsdiagramm für Teilnehmer

Die (erneute) Durchführung von INI oder HIA ist im Teilnehmerzustand „Bereit“ nicht zulässig. Damit soll verhindert werden, dass Teilnehmer ungewollt vom Zustand „Bereit“ in den Zustand „Teilweise initialisiert(INI)“ oder „Teilweise initialisiert(HIA)“ versetzt werden. Das

hätte nämlich zur Folge, dass betroffene Teilnehmer zunächst keine weiteren bankfachlichen Aufträge ausführen könnten.

Teilnehmer im Zustand „Bereit“ müssen zuerst ihren DFÜ-Zugang zum Banksystem sperren, bevor sie eine erneute Teilnehmerinitialisierung vornehmen können. Details zum Sperren von Teilnehmern sind in Kapitel 4.5 beschrieben.

4.4.1.2.5 Beschreibung der EBICS-Nachrichten

4.4.1.2.5.1 Format der Auftragsdaten

INI unterstützt aus Gründen der Kompatibilität zum FTAM-Verfahren beim Signaturverfahren A004 weiterhin die Datenstruktur (Public-Key-Datei), die in der „Spezifikation für die FTAM-Anbindung“ (Anlage 2 des DFÜ-Abkommens) für INI-Dateien definiert wurde (siehe Kapitel 14.2.5.4).

Bei Verwendung der EU in strukturierter Form (ab dem Signaturverfahren A005/A006) sind die Auftragsdaten für INI ein zu ebics_signature.xsd konformes Instanzdokument, das aus dem Toplevel-Element `SignaturePubKeyOrderData` besteht.
`SignaturePubKeyOrderData` ist mittels XML-Schema wie folgt definiert:

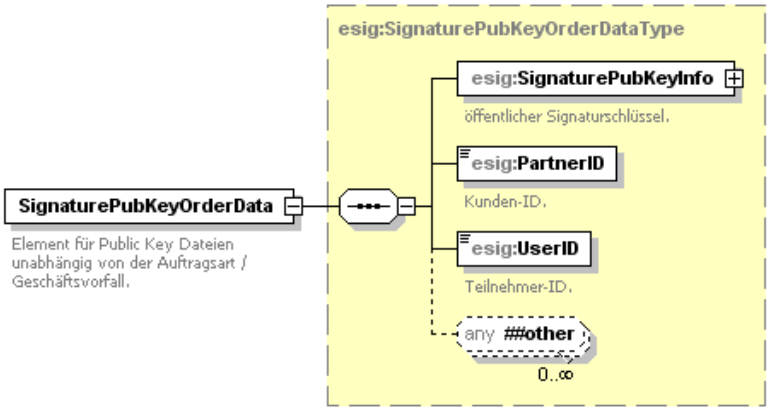


Abbildung 17: Definition des XML-Schema-Elements SignaturePubKeyOrderData der Auftragsdaten für INI (identisch mit PUB, siehe eigenes Kapitel)

Die Auftragsdaten für HIA sind ein zu ebics_orders_H004.xsd konformes Instanzdokument, das aus dem Toplevel-Element `HIARequestOrderData` besteht. `HIARequestOrderData` ist mittels XML-Schema wie folgt definiert:

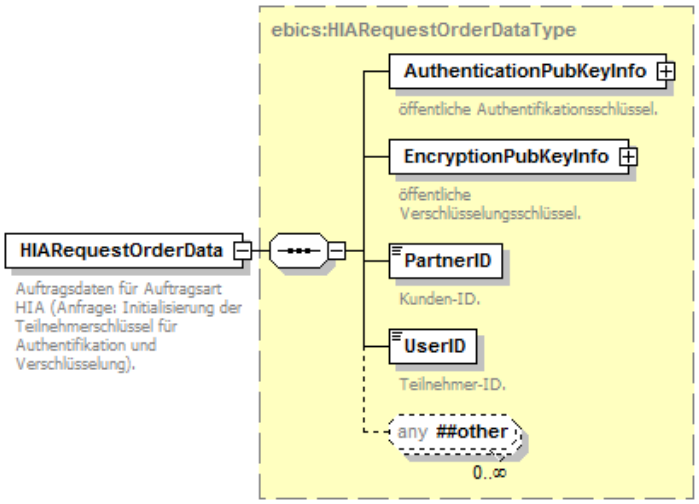


Abbildung 18: Definition des XML-Schema-Elements HIARequestOrderData der Auftragsdaten für HIA

Die Auftragsdaten von INI und HIA werden jeweils komprimiert und base64-kodiert in den entsprechenden EBICS-Request eingebettet.

4.4.1.2.5.2 Beschreibung und Beispielnachrichten

Dieses Kapitel beschreibt die EBICS-Nachrichten für die Auftragsarten INI und HIA. INI- und HIA-Requests sind zu ebics_keymgmt_request_H004.xsd konforme Instanzdokumente mit dem Toplevel-Element `ebicsUnsecuredRequest`. INI- und HIA-Responses sind zu ebics_keymgmt_response_H004.xsd konforme Instanzdokumente mit dem Toplevel-Element `ebicsKeyManagementResponse`.

Hier werden die Daten aufgezählt, die Bestandteil dieser Nachrichten sind. In Klammern stehen jeweils die entsprechenden XML-Elemente in XPath-Notation. Es gelten die folgenden Konventionen:

- Daten, die grundsätzlich optional sind, werden mit „(optional)“ gekennzeichnet
- Daten, die nur unter bestimmten Voraussetzungen fehlen dürfen, werden stattdessen mit „(bedingt)“ gekennzeichnet
- Optionale XML-Elemente der EBICS-Nachrichten, die in der Beschreibung fehlen, dürfen in der EBICS-Nachricht nicht vorkommen
- Optionale XML-Elemente der EBICS-Nachrichten, die in der Beschreibung ohne die Kennzeichnung „(optional)“ oder „(bedingt)“ vorkommen, müssen immer entsprechend der Beschreibung gesetzt sein.

Ergänzt wird diese Beschreibung durch Beispiele.

- **Übertragung der folgenden Daten im INI-Request** (siehe Beispiel in Abbildung 19)
  - Host-ID des EBCIS Bankrechners (ebicsUnsecuredRequest/header/static/HostId)
  - Teilnehmer (`ebicsUnsecuredRequest/header/static/PartnerID`, `ebicsUnsecuredRequest/header/static/UserID`), dessen öffentlicher bankfachlicher Schlüssel an das Kreditinstitut übermittelt werden soll
  - (optional) technischer Teilnehmer (`ebicsUnsecuredRequest/header/static/PartnerID`, `ebicsUnsecuredRequest/header/static/SystemID`) `SystemID` kann in der Nachricht enthalten sein, wenn es sich bei dem Kundensystem um ein Multi-Usersystem handelt. Da INI-Requests keine Authentifikationssignatur enthalten und die Auftragsdaten unverschlüsselt sind, ist die Angabe von SystemID optional.
  - (optional) Information zum Kundenprodukt (ebicsUnsecuredRequest/header/static/Product)
  - Auftragsart (ebics`UnsecuredRequest`/header/static/OrderDetails/OrderType) mit der Belegung “INI”
  - Auftragsattribute (ebicsUnsecuredRequest/header/static/OrderDetails/OrderAttribute) mit der Belegung “DZNNN”
  - Sicherheitsmedium für den bankfachlichen Schlüssel des Teilnehmers `(ebicsUnsecuredRequest/header/static/SecurityMedium)` Die zulässigen Belegungen sind im Anhang (Kapitel 12.4) enthalten
  - Auftragsdaten (ebicsUnsecuredRequest/body/DataTransfer/OrderData).

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsUnsecuredRequest
 xmlns=" urn:org:ebics:H004" xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_ keymgmt_request_H004.xsd"
 Version="H004" Revision="1">
 <header authenticate="true">
   <static>
     <HostID>EBIXHOST</HostID>
     <PartnerID>KUNDE001</PartnerID>
     <UserID>TLN100</UserID>
     <OrderDetails>
       <OrderType>INI</OrderType>
       <OrderAttribute>DZNNN</OrderAttribute>
     </OrderDetails>
     <SecurityMedium>0200</SecurityMedium>
   </static>
   <mutable/>
 </header>
```

```
 <body>
   <DataTransfer>
     <!--INI-Datei gemäß Kapitel 14.2.5.4, komprimiert und base64-kodiert-->
     <OrderData>
       …
     </OrderData>
   </DataTransfer>
 </body>
</ebicsUnsecuredRequest>
```

Abbildung 19: EBICS-Request für die Auftragsart INI

- **Übertragung der folgenden Daten im INI-Response** (siehe Beispiel in Abbildung 20):
  - bankfachlicher Returncode (ebicsKeyManagementResponse/body/ReturnCode)
  - Auftragsnummer (`ebicsKeyManagementResponse`/header/mutable/OrderID)
    Diese Nummer wird durch den Bankserver automatisch bestimmt.
  - technischer Returncode (ebicsKeyManagementResponse/header/mutable/ReturnCode)
  - technischer Reporttext (ebicsKeyManagementResponse/header/mutable/ReportText)
  - (optional) Zeitstempel der letzten Aktualisierung der Bankparameter (`ebicsKeyManagementResponse/body/TimestampBankParameter`).

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsKeyManagementResponse
 xmlns="urn:org:ebics:H004"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_keymgmt_response_H004"
 Version="H004" 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>
```

Abbildung 20: EBICS-Response für die Auftragsart INI

- **Übertragung der folgenden Daten im HIA-Request** (analog zu INI, siehe Beispiel in Abbildung 21):

- Host-ID des EBCIS Bankrechners (ebicsUnsecuredRequest/header/static/HostId)
- Teilnehmer (`ebicsUnsecuredRequest/header/static/PartnerID`, `ebicsUnsecuredRequest/header/static/UserID`), dessen öffentlicher Authenifikationsschlüssel und öffentlicher Verschlüsselungsschlüssel an das Kreditinstitut übermittelt werden soll
- (optional) technischer Teilnehmer (`ebicsUnsecuredRequest/header/static/PartnerID`, `ebicsUnsecuredRequest/header/static/SystemID`)
  `SystemID` kann in der Nachricht enthalten sein, wenn es sich bei dem Kundensystem um ein Multi-Usersystem handelt. Da HIA-Requests keine Authentifikationssignatur enthalten und die Auftragsdaten unverschlüsselt sind, ist die Angabe von `SystemID` optional
- (optional) Information zum Kundenprodukt (`ebicsUnsecuredRequest/header/static/Product`)
- Auftragsart (ebicsUnsecuredRequest/header/static/OrderDetails/OrderType) mit der Belegung “HIA”
- Auftragsattribute (ebicsUnsecuredRequest/header/static/OrderDetails/OrderAttribute) mit der Belegung “DZNNN”
- Sicherheitsmedium für den bankfachlichen Schlüssel des Teilnehmers (`ebicsUnsecuredRequest/header/static/SecurityMedium`) mit der Belegung „0000”.
  Das Sicherheitsmedium für den bankfachlichen Schlüssel des Teilnehmers ist mit „0000“ belegt, da HIA-Aufträge weder bankfachliche Schlüssel übermitteln noch EUs enthalten
- Auftragsdaten (`ebicsUnsecuredRequest/body/DataTransfer/OrderData`).

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsUnsecuredRequest
 xmlns=" urn:org:ebics:H004"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_keymgmt_request_H004.xsd"
 Version="H004" Revision="1">
 <header authenticate="true">
   <static>
     <HostID>EBIXHOST</HostID>
     <PartnerID>KUNDE001</PartnerID>
     <UserID>TLN100</UserID>
     <OrderDetails>
       <OrderType>HIA</OrderType>
       <OrderAttribute>DZNNN</OrderAttribute>
     </OrderDetails>
     <SecurityMedium>0000</SecurityMedium>
   </static>
```

```
    <mutable/>
  </header>
  <body>
    <DataTransfer>
      <!--Zu ebics_orders_H004.xsd konformes (XML)-Instanzdokument mit Wurzelelement
HIARequestOrderData, komprimiert und base64-kodiert-->
      <OrderData>
        …
      </OrderData>
    </DataTransfer>
  </body>
</ebicsUnsecuredRequest>
```

Abbildung 21: EBICS-Request für die Auftragsart HIA

- **Übertragung der folgenden Daten im HIA-Response** (analog zu INI, siehe Beispiel in Abbildung 22):
  - bankfachlicher Returncode (ebicsKeyManagementResponse/body/ReturnCode)
  - Auftragsnummer (ebicsKeyManagementResponse/header/mutable/OrderID). Diese Nummer wird durch den Bankserver automatisch bestimmt.
  - technischer Returncode (ebicsKeyManagementResponse/header/mutable/ReturnCode)
  - technischer Reporttext (ebicsKeyManagementResponse/header/mutable/ReportText)
  - (optional) Zeitstempel der letzten Aktualisierung der Bankparameter (`ebicsKeyManagementResponse/body/TimestampBankParameter`).

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsKeyManagementResponse
  xmlns=" urn:org:ebics:H004"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:org:ebics:H004 ebics_keymgmt_response_H004.xsd"
  Version="H004" 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>
```

Abbildung 22: EBICS-Response für die Auftragsart HIA

4.4.1.3 Initialisierung via H3K

Wenn der bankfachliche (EU-) Schlüssel auf einem selbstsignierten Zertifikat basiert oder ein RSA-Schlüssel ohne Zertifikat ist, dann sind die öffentlichen Teilnehmerschlüssel weiterhin via INI / HIA an die Bank zu übermitteln. Um die Authentizität der öffentliche Teilnehmerschlüssel zu garantieren, muss hier sicher gestellt sein, dass die Bank diese zusätzlich über einen zweiten unabhängigen Kommunikationsweg erhält (Initialisierungsbriefe sowohl für INI als auch für HIA). Hat die Bank die Schlüssel über diese beiden unabhängigen Kommunikationswege erhalten, vergleicht sie diese zuerst, bevor sie sie anerkennt.

Bei Nutzung von Zertifikaten ist der Prozess für H3K jetzt wie folgt:

1) Das Zertifikat für die bankfachliche Unterschrift (EU) muss von einer Zertifizierungsinstanz (CA) ausgegeben sein. In diesem Fall ist kein Initialisierungsbrief für den Schlüssel notwendig.
   Allerdings können die Zertifikate für Verschlüsselung und die Authentifikationssignatur sowohl selbst-signiert (self-signed) als auch von einer CA ausgegeben sein.
2) Für die Übermittlung der öffentlichen Teilnehmerschlüssel für Verschlüsselung und die Authentifikationssignatur gilt, dass für diese, wenn sie mit einer (gültigen) EU unterschrieben sind, ebenfalls kein Intialisierungsbrief notwendig ist.
3) Die notwendigen bankseitigen Prüfungen (vor der erstmaligen Verwendung der Schlüssel) sind:
   a. Gibt es eine Vereinbarung zur Nutzung des Zertifikats?
   b. Sind alle administrativen Schritte für den Kunden/Teilnehmer auf dem EBICS-Bankrechner erledigt und ist der Teilnehmer dem EBICS-Bankrechner bekannt?
   c. Ist das Zertifikat gültig?
4) Unter Berücksichtigung von 1) bis 3) kann die neue Auftragsart H3K definiert werden:
   a. Diese kombiniert INI und HIA (Übermittlung von drei öffentlichen Teilnehmerschlüsseln, alle Schlüssel basieren auf Zertifikaten); für den EU-Schlüssel muss das Zertifikat von einer CA ausgegeben sein.
   b. H3K enthält bereits eine Elektronische Unterschrift (durch die von einer CA zertifizierten EU-Schlüssel)

Dieses Konzept setzt bei den Zertifikaten, die im H3K-Prozess verwendet werden (und dort als EU-Schlüssel verwendet werden) eine hohe Authentizität voraus, gleiches gilt für die Zertifizierungsinstanz (CA), die diese Zertifikate ausgibt:

- Ausgabe des Zertifikats:
  - Strenge Vorgaben hinsichtlich Identifikation (die Person und die Organisation betreffend, die das Zertifikat anfordert)
  - Alle Daten des Zertifikats wurden sorgfältig durch die Registrierungsinstanz validiert.
- Namemsregeln:
  - Für den Namen in Zertifikat (SubjectName) muss ein festes Schema vorhanden sein, das eine eindeutige (automatische) Zuordnung einer natürlichen Person zu einem Unternehmen zulässt.
- Kryptografische Anforderungen:
  - Z.B. Schlüssellängen
- Validierungsanforderungen:

- Aktualität und Verfügbarkeit von Listen mit ungültig erklärten Zertifikaten (Zertifikatssperrlisten; revocation lists)

Für die elektronische Initialisierung (ohne INI-Brief) neuer EBICS-Teilnehmer mit der Auftragsart H3K ist die Verwendung von Zertifikaten verpflichtend.
Wenn ein Zertifikat nicht von der Bank selbst ausgegeben wurde, ist eine notwendige Voraussetzung für die elektronische Initialisierung, dass der neue EBICS-Teilnehmer seiner Bank sein Zertifikat für die (bankfachliche) EU, das seinen eindeutigen Eigentümernamen (subject name) und den Namen der CA enthält, und das relevante CA Rootzertifikat (Stammzertifikat) bekannt gegeben hat. Der neue EBICS-Teilnehmer unterschreibt mit diesem Zertifikat die Zertifikate für Authentifikation und Verschlüsselung.
Die Anforderungen, die in der Zertifikatspolicy erfüllt sein müssen, werden bilateral zwischen Kunde und Bank abgestimmt. Die Interoperabilität von verschiedenen Trust Domains (Vertrauensdomänen) kann nur erreicht werden, wenn entsprechende technische, organisatorische und gesetzliche Anforderungen definiert sind. Diese Anforderungen sind nicht Gegenstand der EBICS-Spezifikation, da sie für den Kommunikationsstandard selbst nicht relevant sind.

In der Schemadatei ebics_order_H004.xsd enthält die Elementgruppe `H3KRequestOrderData` drei Zertifikate. In der Schemadatei ebics_keymgmt_request_H004 enthält die Struktur ebics:UnsignedRequest `H3KRequestOrderData` (komprimiert und base-64 kodiert) und eine Signatur.
In der Schemadatei ebics_ types_H004.xsd wird der Datentyp `SignatureCertificateInfoType`, der für die Übermittlung eines X509-Zertifikats benötigt wird, definiert.
`AuthenticationCertificateInfoType` und `EncryptionInfoType` sind Erweiterungen dieses Typs und jeder dieser Typen wird in `H3KOrderData` verwendet.

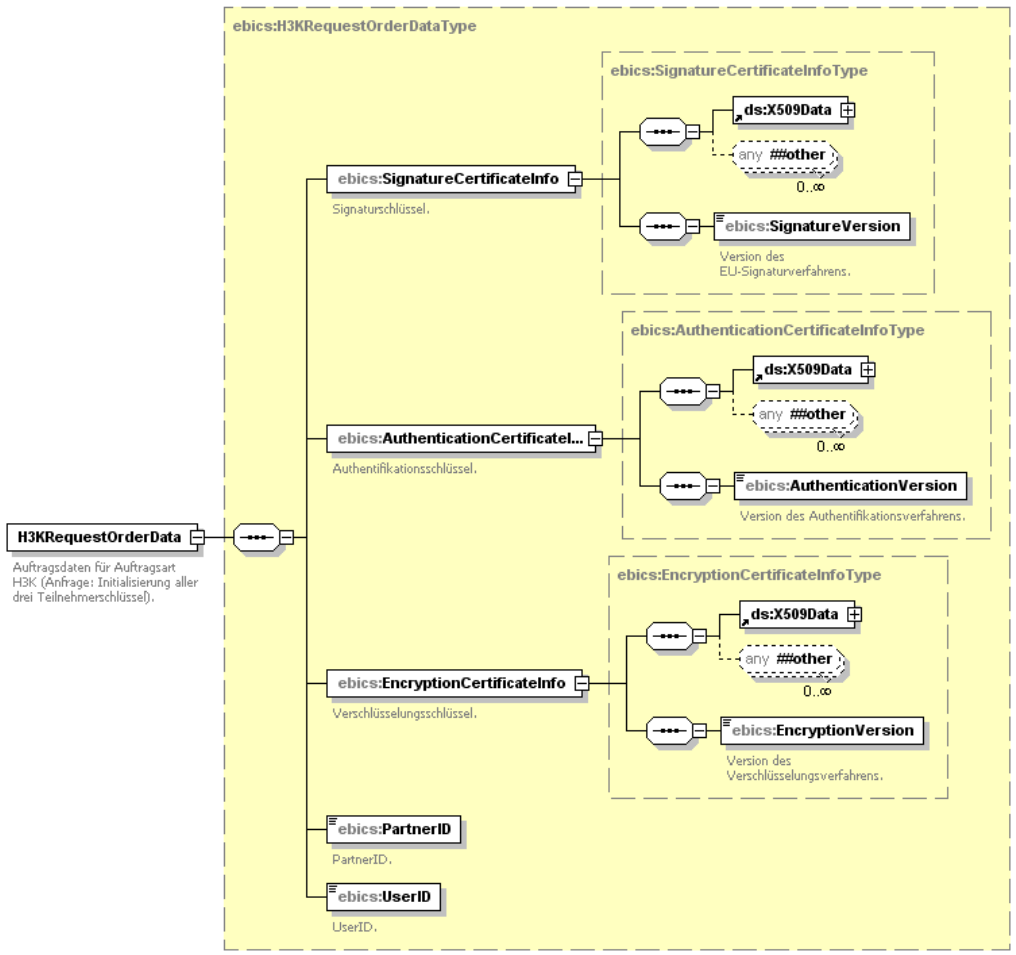


Diagram 23: Definition des XML-Schema-Elements H3KRequestOrderData für H3KOrder Data

Bei der Teilnehmerinitialisierung sieht der bankseitige Prüfprozess wie folgt aus:

1- Überprüfung der Struktue `H3KRequestOrderData` und Extrahieren des für die EU (Autorisierung) vorgesehenen Zertifikats.

2- Überprüfung, ob dieses Zertifikat von einer zulässigen Zertifizierungsinstanz (CA) stammt.

3- Überprüfung, ob die Unterschrift in SignatureData zum öffentlichen Teilnehmerschlüssel für die EU in `H3KRequestOrderData` passt.

4- Überprüfung, ob die Kundeninformationen im Zertifikat mit den früheren Angaben übereinstimmen (`ebicsUnsignedRequest` → `body` → `DataTransfer`)
Note: Die Bank wählt für die Prüfung der Übereinstimmung notwendigen Mittel und Informationen. Beispielsweise kann diese Information in einem Vertrag angegeben

oder durch das Hochladen von Zertifikaten auf die Website der Bank veröffentlicht worden sein.

a. Bei Übereinstimmung gibt der Bankrechner den Return Code EBICS_OK (Code-Nummer 000000) zurück. Der Status des Teilnehmers wird automatisch auf „**Bereit**“ gesetzt. Der Teilnehmer braucht keine Initialisierungsbriefe zu senden (auch keine anderer Validierungsprozess ist notwendig)

b. Wenn der Bankrechner das Zertifikat (zum öffentlichen Teilnehmerschlüssel für die EU) nicht erfolgreich mit den früheren Angaben abgleichen kann, sendet er eine Ablehnungs-Response und der H3KRequest wird abgebrochen. Der Bankrechner gibt den Return Code EBICS_CERTIFICATES_VALIDATION_ERROR (Code-Nummer 091219) zurück. Der Status des Teilnehmers bleibt gleich („**Neu**“). Der Teilnehmer hat zwei Möglichkeiten, weiter vorzugehen:

   i. Erneute Durchführung eines H3K-Requests mit einem korrekten Zertifikat (für den öffentlichen Teilnehmerschlüssel für die EU), das von einer CA ausgegeben wurde

   ii. Oder Durchführung der Initialisierung mit INI und HIA (INI/HIA ist also insbesondere als Ausweichprozess nutzbar)

Das zur EU zu verwendende Zertifikat muss gültig sein (vor allem darf das Gültigkeitsdatum nicht abgelaufen sein). Ansonsten muss der Kunde sein (von einer CA ausgegebenes) Zertifikat aktualisieren bzw. ein neues angeben.

Zudem können alle Fehlermeldungen , die sich auf die Prozesse des Schlüsselmanagements beziehen, auch im H3K Prozess vorkommen.

4.4.2 Abholung der öffentlichen Schlüssel des Kreditinstituts

4.4.2.1 Allgemeine Beschreibung

Der Teilnehmer holt sämtliche öffentlichen Schlüssel des Kreditinstituts mittels einer eigens dafür vorgesehenen systembedingten Auftragsart (HPB) ab. Die Abholung der öffentlichen Bankschlüssel erfordert den Teilnehmerzustand „Bereit“, da erst dann die Verfahren feststehen, die der Teilnehmer für die Authentifikationssignatur, bankfachliche Unterschrift und Verschlüsselung einsetzen möchte.

Die Durchführung von HPB erfordert lediglich ein einziges EBICS-Request-/ EBICS-Response-Paar. Der EBICS-Request von HPB enthält die Authentifikationsunterschrift des Teilnehmers selbst oder die eines technischen Teilnehmers desselben Kunden über die Steuerdaten.

Das Ablaufdiagramm in Abbildung 24 stellt die bankseitige Verarbeitung dar, die beim Empfang eines HPB-Requests stattfindet. Der Replay-Test erfolgt wie bei den bankfach-

lichen Auftragsarten (siehe dazu Kapitel 11.4). Demnach liefert HPB den technischen Fehler EBICS_TX_MESSAGE_REPLAY, wenn der HPB-Request eine wiedereingespielte Nachricht ist. Ebenso erfolgt die Überprüfung der Kunden-/TeilnehmerID, des Teilnehmerzustands sowie der Authentifikationssignatur wie im Falle bankfachlicher Auftragsarten innerhalb des Ablaufschritts “Überprüfung der Authentizität des EBICS-Requests” (siehe Kapitel 5.5.1.2.1, Abbildung 47).

Diese kann die folgenden Fehlercodes liefern:

- EBICS_AUTHENTICATION_FAILED
  Dieser technische Fehler tritt dann auf, wenn die Authentifikationssignatur des Teilnehmers (oder des technischen Teilnehmers) nicht erfolgreich geprüft werden konnte
- EBICS_USER_UNKNOWN
  Dieser technische Fehler tritt dann auf, wenn die Authentifikationssignatur des technischen Users erfolgreich verifiziert werden konnte, der (nichttechnische) Teilnehmer jedoch dem Kreditinstitut unbekannt ist
- EBICS_INVALID_USER_STATE
  Dieser technische Fehler tritt dann auf, wenn die Authentifikationssignatur des technischen Users erfolgreich verifiziert werden konnte und der (nichttechnische) Teilnehmer dem Kreditinstitut bekannt ist, jedoch nicht im Zustand „Bereit“ ist.

Nach der erfolgreichen Durchführung der „Überprüfung der Authentizität der Nachricht“ liefert die eigentliche Durchführung des HPB-Auftrags keine weiteren spezifischen technischen oder bankfachlichen Fehler.

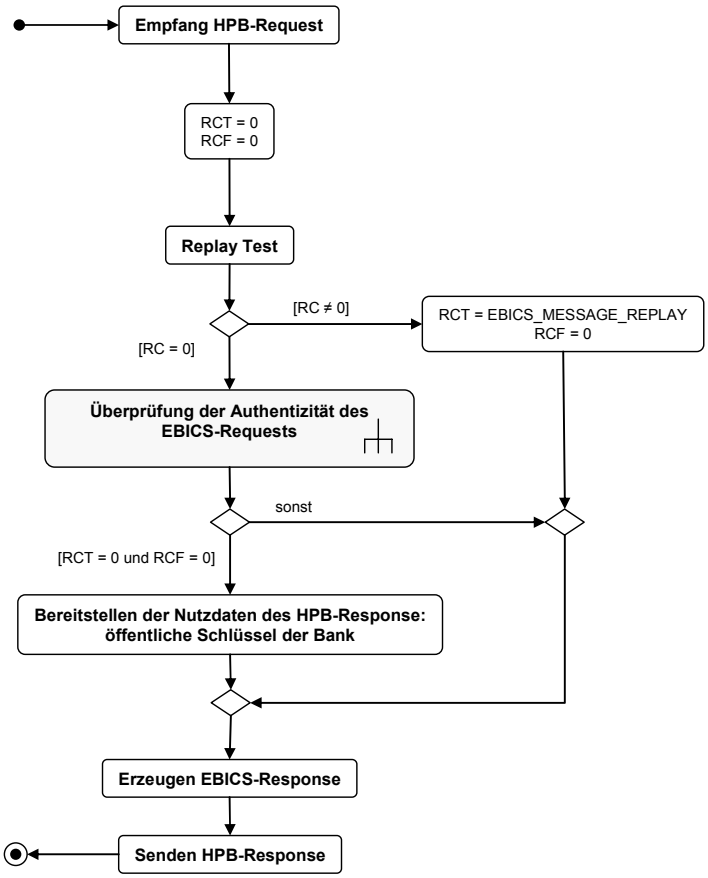


Abbildung 24: Bankseitige Verarbeitung eines HPB-Requests

Für die EBICS-Response gilt:

- sie enthält keine Authentifikationsunterschrift, da der öffentliche Authentifikationsschlüssel des Kreditinstituts in dieser Response erst übertragen wird. Der öffentliche Authentifikationsschlüssel des Kreditinstituts kann vom Teilnehmer noch nicht zur Prüfung der Authentifikationssignatur verwendet werden, da seine Authentizität zu diesem Zeitpunkt noch nicht gesichert ist
- sie enthält keine bankfachliche elektronische Unterschrift der Nutzdaten, d.h. der öffentlichen Bankschlüssel, da der bankfachliche Schlüssel des Kreditinstituts in dieser Response erst übertragen wird. Der öffentliche bankfachliche Schlüssel des Kreditinstituts kann vom Teilnehmer noch nicht zur Prüfung der bankfachlichen EU verwendet werden, da seine Authentizität zu diesem Zeitpunkt noch nicht gesichert ist

- sie enthält die Nutzdaten, d.h. die öffentlichen Bankschlüssel, in verschlüsselter Form, da der öffentliche Verschlüsselungsschlüssel des Teilnehmers durch das Kreditinstitut bereits freigeschaltet wurde.

Nach der erfolgreichen Durchführung von HPB verfügt der Teilnehmer über alle notwendigen öffentlichen Bankschlüssel, die er vor dem ersten Einsatz jedoch noch überprüfen muss: wie in Abbildung 13 dargestellt ist der Zustand dieser Schlüssel auf Teilnehmerseite „Neu“. Im Zustand „Neu“ dürfen Bankschlüssel nicht für die Kommunikation über EBICS eingesetzt werden, da ihre Authentizität nicht sichergestellt ist.

Um die Authentizität der Bankschlüssel zu gewährleisten, ist durch das Kreditinstitut sicherzustellen, dass der Teilnehmer die öffentlichen Bankschlüssel und/oder die Hashwerte über einen zweiten, unabhängigen Kommunikationsweg (z.B. über die Internetseite des Kreditinstituts) erhält. Es liegt in der Verantwortung des Teilnehmers, die Prüfung der Bankschlüssel vorzunehmen. Das Verfahren der Prüfung der Bankschlüssel ist nicht Bestandteil des vorliegenden Standards. Wenn die Bank von einer CA ausgegebene Zertifikate bereitstellt, ist der Kunde verpflichtet, deren Gültigkeit zu überprüfen (Anmerkungen siehe im Common Implemtation Guide). Es ist abhängig von der Implementierung der EBICS-Clientsysteme, welche sicherstellen, dass Teilnehmer die öffentlichen Schlüssel erst nach einer erfolgreichen Prüfung einsetzen.

Nach der erfolgreichen Prüfung verändert sich der Zustand der öffentlichen Bankschlüssel auf Teilnehmerseite von “Neu” auf “Freigeschaltet”. Diese Zustandsänderung ist ebenfalls in Abbildung 13 dargestellt. Nur Bankschlüssel im Zustand „Freigeschaltet“ dürfen für die Kommunikation über EBICS eingesetzt werden.

In der EBICS-Version „H004“ ist die EU der Kreditinstitute nur vorgesehen (siehe Kapitel 3.5.2). Abbildung 13 berücksichtigt den Zustand des öffentlichen bankfachlichen Schlüssels der Bank auf Teilnehmerseite in Vorbereitung zukünftiger EBICS-Versionen.

4.4.2.2 Beschreibung der EBICS-Nachrichten

4.4.2.2.1 Format der Auftragsdaten

Die Auftragsdaten für HPB sind ein zu ebics_orders_H004.xsd konformes Instanzdokument, das aus dem Toplevel-Element `HPBResponseOrderData` besteht.
`HPBResponseOrderData` ist mittels XML-Schema wie folgt definiert:

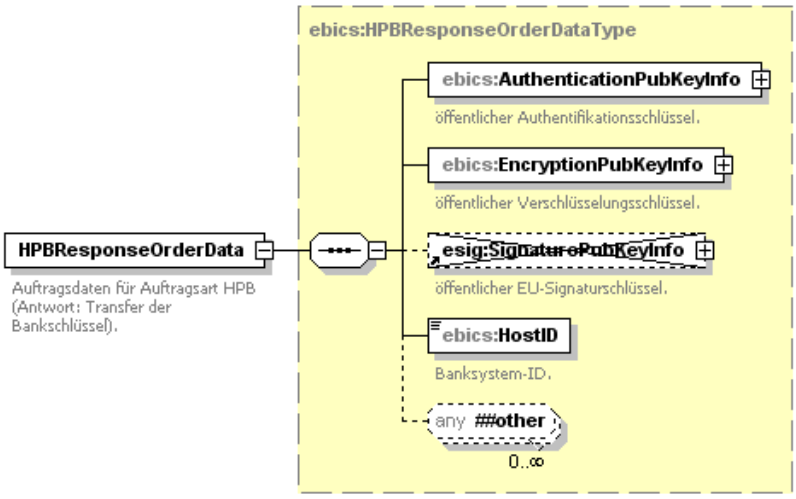


Abbildung 25: Definition des XML-Schema-Elements HPBResponseOrderData der Auftragsdaten für HPB

In der Version „H004“ des EBICS-Protokolls ist die EU der Kreditinstitute nur vorgesehen (siehe dazu Kapitel 3.5.2). Das Element `SignaturePubKeyInfo` ist in Vorbereitung zukünftiger EBICS-Versionen mit der maximalen Häufigkeit (maxOccurs) gleich 0 definiert.

Die Auftragsdaten werden komprimiert, verschlüsselt und base64-kodiert in den entsprechenden HPB-Response eingebettet.

4.4.2.2.2 Beschreibung und Beispielnachrichten

Dieses Kapitel beschreibt die EBICS-Nachrichten für die Auftragsart HPB. HPB-Requests sind zu ebics_keymgmt_request_H004.xsd konforme Instanzdokumente mit dem Toplevel-Element `ebicsNoPubKeyDigestsRequest`. Hingegen sind HPB-Responses zu ebics_keymgmt_response_H004.xsd konforme Instanzdokumente mit dem Toplevel-Element `ebicsKeyManagementResponse`.

Hier werden die Daten aufgezählt, die Bestandteil dieser Nachrichten sind. In Klammern stehen jeweils die entsprechenden XML-Elemente in XPath-Notation. Es gelten die folgenden Konventionen:

- Daten, die grundsätzlich optional sind, werden mit „(optional)“ gekennzeichnet.
- Daten, die nur unter bestimmten Voraussetzungen fehlen dürfen, werden stattdessen mit „(bedingt)“ gekennzeichnet
- Optionale XML-Elemente der EBICS-Nachrichten, die in der Beschreibung fehlen, dürfen in der EBICS-Nachricht nicht vorkommen

- Optionale XML-Elemente der EBICS-Nachrichten, die in der Beschreibung ohne die Kennzeichnung „(optional)“ oder „(bedingt)“ vorkommen, müssen immer entsprechend der Beschreibung gesetzt sein.

Ergänzt wird diese Beschreibung durch ein Beispiel für ein EBICS-Request-/-Response-Paar für die Auftragsart HPB.

- **Übertragung der folgenden Daten im HPB-Request** (siehe Beispiel in Abbildung 26):
  - Host-ID des EBCIS Bankrechners (ebicsNoPubKeyDigestsRequest/header/static/HostId)
  - Kombination aus Nonce und Timestamp, erforderlich für die Vermeidung von Replay alter EBICS-Nachrichten (`ebicsNoPubKeyDigestsRequest/header/static/Nonce`, `ebicsNoPubKeyDigestsRequest/header/static/Timestamp`)
  - Teilnehmer (`ebicsNoPubKeyDigestsRequest/header/static/PartnerID`, `ebicsNoPubKeyDigestsRequest/header/static/UserID`), der den HPB-Request initiiert
  - (bedingt) technischer Teilnehmer (`ebicsNoPubKeyDigestsRequest/header/static/PartnerID`, `ebicsNoPubKeyDigestsRequest/header/static/SystemID`) `SystemID` muss genau dann vorhanden sein, wenn es sich bei dem Kundensystem um ein Multi-Usersystem handelt. Der technische Teilnehmer ist für das Erstellen der EBICS-Requests (inklusive der Authentifkationssignaturen) zuständig, die zu Aufträgen gehören, die vom Teilnehmer eingereicht oder bankfachlich unterzeichnet werden.
  - (optional) Information zum Kundenprodukt (ebicsNoPubKeyDigestsRequest/header/static/Product)
  - Auftragsart (ebicsNoPubKeyDigestsRequest/header/static/OrderDetails/OrderType) mit der Belegung “HPB”
  - Auftragsattribute (ebicsNoPubKeyDigestsRequest/header/static/OrderDetails/OrderAttribute) mit der Belegung “DZHNN”
  - Sicherheitsmedium für den bankfachlichen Schlüssel des Teilnehmers (`ebicsNoPubKeyDigestsRequest/header/static/SecurityMedium`) mit der Belegung “0000”.
    Das Sicherheitsmedium für den bankfachlichen Schlüssel des Teilnehmers ist mit „0000“ belegt, da HPB-Aufträge keine EUs erfordern und auch keine bankfachlichen Teilnehmerschlüssel übermitteln

- Authentifikationssignatur des technischen Teilnehmers, falls ein solcher vorhanden ist, ansonsten die Authentifikationssignatur des Teilnehmers selbst (`ebicsNoPubKeyDigestsRequest/AuthSignature`)
  Die Authentifikationsignatur umfasst alle XML-Elemente des EBICS-Requests, deren Attributwert für `@authenticate` gleich „true“ ist. Die Definition des XML-Schemas „ebics_keymgmt_request_H004.xsd“ gewährleistet, dass der Wert des Attributs `@authenticate` für genau die Elemente gleich „true“ ist, die auch unterschrieben werden müssen

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsNoPubKeyDigestsRequest
 xmlns=" urn:org:ebics:H004"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_keymgmt_request_H004.xsd"
 Version="H004" Revision="1">
 <header authenticate="true">
   <static>
     <HostID>EBIXHOST</HostID>
     <Nonce>234AB2340FD2C23035764578FF3091FA</Nonce>
     <Timestamp>2005-01-30T15:40:45.123Z</Timestamp>
     <PartnerID>KUNDE001</PartnerID>
     <UserID>TLN100</UserID>
     <OrderDetails>
       <OrderType>HPB</OrderType>
       <OrderAttribute>DZHNN</OrderAttribute>
     </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/xmlenc#sha256"/>
       <ds:DigestValue> …hier Hashwert Authentifikation..</ds:DigestValue>
     </ds:Reference>
   </ds:SignedInfo>
   <ds:SignatureValue> …hier Signaturwert Authentifikation..</ds:SignatureValue>
 </AuthSignature>
 <body/>
</ebicsNoPubKeyDigestsRequest>
```

Abbildung 26: EBICS-Request für die Auftragsart HPB

- **Übertragung der folgenden Daten im EBICS-Response für HPB** (siehe Beispiel in Abbildung 27):
  - bankfachlicher Returncode (ebicsKeyManagementResponse/body/ReturnCode)

- technischer Returncode (ebicsKeyManagementResponse/header/mutable/ReturnCode)
- technischer Reporttext (ebicsKeyManagementResponse/header/mutable/ReportText)
- (bedingt) Information zur Verschlüsselung der Auftragsdaten und eventuell der EU der Auftragsdaten (`ebicsKeyManagementResponse/body/DataTransfer/DataEncryptionInfo`), falls keine Fehler technischer oder bankfachlicher Natur aufgetreten sind.
  `DataEncryptionInfo` enthält insbesondere auch den asymmetrisch verschlüsselten Transaktionsschlüssel (`ebicsKeyManagementResponse/body/DataTransfer/DataEncryptionInfo/TransactionKey`)
- (bedingt) die Nutzdaten (`ebicsKeyManagementResponse/body/DataTransfer/OrderData`), falls keine Fehler technischer oder bankfachlicher Natur aufgetreten sind
- (optional) Zeitstempel der letzten Aktualisierung der Bankparameter (`ebicsKeyManagementResponse/body/TimestampBankParameter`).

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsKeyManagementResponse
 xmlns="urn:org:ebics:H004"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_keymgmt_response_H004.xsd "
 Version="H004" 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">..hier Hashwert des öffentlichen
Schlüssels für die Verschlüsselung ..</EncryptionPubKeyDigest>
       <!--asymmetrisch verschlüsselter Transaktionsschlüssel-->
       <TransactionKey>…</TransactionKey>
     </DataEncryptionInfo>
     <!--Zu ebics_orders_H004.xsd konformes (XML-)Instanzdokument mit Wurzelelement
HPBResponseOrderData, komprimiert, verschlüsselt, base64-kodiert-->
     <OrderData>
       …
     </OrderData>
   </DataTransfer>
   <ReturnCode authenticate="true">000000</ReturnCode>
 </body>
</ebicsKeyManagementResponse>
```

Abbildung 27: EBICS-Response für die Auftragsart HPB

4.5 Sperren eines Teilnehmers

4.5.1 Alternativen

Besteht der Verdacht der Kompromittierung der Teilnehmerschlüssel, MUSS der Teilnehmer seine Zugangsberechtigung zu allen Banksystemen sperren, die den/die kompromittierten Schlüssel verwenden.

Teilnehmer, die ihren DFÜ-Zugang zu einem Banksystem sperren möchten, können dies auf zwei Arten tun:
Im Zustand „Bereit“ kann der Teilnehmer die Sperrung über EBICS mit Hilfe der systembedingten Auftragsart SPR vornehmen.

- Die SPR Transaktion ist eine Standard-Upload Transaktion, die ausschließlich eine EU-Datei mit der Unterschrift des zu sperrenden Teilnehmers über eine Dummy-Datei übermittelt. Die Dummy-Datei enthält genau ein Leerzeichen und wird nicht übertragen. Der entsprechende EBICS-Request muss neben dieser Unterschrift des zu sperrenden Teilnehmers auch eine Authentifikationssignatur enthalten. Die Authentifikationssignatur darf auch von einem technischen Teilnehmer geleistet werden.
  Der SPR-Auftrag umfasst keine weiteren Nutzdaten und somit auch keine Auftragsdatei.
- Der Teilnehmer hat zudem die Möglichkeit, die Sperre über einen zweiten Kommunikationsweg vorzunehmen, z.B. telefonisch über einen bestimmten Ansprechpartner des Kreditinstituts. Bei Verlust oder Beschädigung des Teilnehmerschlüssels ist nur diese Alternative möglich.

Nach der erfolgreichen Durchführung der Sperrung hat der Teilnehmer den Zustand „Gesperrt“, und eine erneute Initialisierung des Teilnehmers ist erforderlich.

4.5.2 Sperren eines Teilnehmers über SPR

SPR ist eine reguläre Upload-Transaktion, deren Ablauf (einschließlich des Verhaltens in Fehlersituationen) in Kapitel 5.5 detailliert beschrieben ist. Nachfolgend sind daher nur Abweichungen und Ergänzungen dargestellt.

Da für SPR nur eine EU-Datei übertragen wird, sendet das Kundensystem nur einen Request mit dem Auftragsattribut UZHNN. Die Verarbeitung findet bereits in der Initialisierungsphase statt, d.h. das Banksytem liefert in der Response keine Transaktions-ID.

Das Banksystem muss sicherstellen, dass der SPR-Request die Authentifikationssignatur des zu sperrenden Teilnehmers bzw. des technischen Teilnehmer enthält. Die Überprüfung der Kunden-/TeilnehmerID, des Teilnehmerzustands sowie der Authentifikationssignatur wird

innerhalb des Ablaufschritts „Überprüfung der Authentizität des EBICS-Requests“ durchgeführt (für Einzelheiten siehe Kapitel 5.5.1.2.1, Abbildung 47).

Die EU-Datei muss eine gültige Elektronische Unterschrift des zu sperrenden Teilnehmers über eine Datei mit einem Leerzeichen enthalten.

Die anschließende eigentliche synchrone Durchführung der Sperrung des Teilnehmers liefert keine weiteren spezifischen technischen oder bankfachlichen Fehler.

4.6 Schlüsseländerungen

4.6.1 Änderung der Teilnehmerschlüssel

Bis zur EBICS-Version 2.3 mussten die Schlüsseländerungen über die beiden Auftragsarten PUB (Änderung des bankfachliches Schlüssels) und HCA (Änderung des Authentifikationsschlüssels und des Verschlüsselungsschlüssels) durchgeführt werden, die unabhängig voneinader ausgeführt werden konnten. Um das key-Management sowohl für die Kunden- als auch für die Bankseite zu vereinfachen, wird ab EBICS 2.4 die Auftragsart HCS eingeführt, mit der alle 3 Schlüssel in einer Transaktion geändert werden können. HCS umfasst fachlich daher PUB und HCA.
HCS bzw. PUB und HCA erfordern jeweils die bankfachliche EU des betroffenen Teilnehmers in der jeweils unterstützen EU-Version (z.B. A004, A005, A006), nicht jedoch den zusätzlichen Versand von Initialisierungsbriefen. Aus Kompatibilitätsgründen können die Auftragsarten PUB und HCA auch weiterhin alternativ zu HCS genutzt werden.
Abhängig vom seinem Zustand hat ein Teilnehmer zwei Möglichkeiten, seine öffentlichen Teilnehmerschlüssel auf dem Banksystem zu aktualisieren:

- Im Zustand “Gesperrt” MUSS grundsätzlich eine Teilnehmerinitialisierung vorgenommen werden, um bankfachliche Aufträge über EBICS übermitteln zu können. Die Sperrung der Zugangsberechtigung mit anschließender Teilnehmerinitialisierung ist somit eine Alternative, Teilnehmerschlüssel zu aktualisieren. Teilnehmerinitialisierung erfolgt unter Verwendung der Auftragsarten INI und HIA und erfordert den zusätzlichen Versand von Initialisierungsbriefen. Der Ablauf der Teilnehmerinitialisierung ist in Kapitel 4.4.1 beschrieben. Informationen zum Thema Sperrung der Zugangsberechtigung eines Teilnehmers sind in Kapitel 4.5 enthalten
- Im Zustand “Bereit” ist es den Teilnehmern möglich, ihre öffentlichen Teilnehmerschlüssel unter Verwendung der systembedingten Auftragsarten PUB, HCS und HCA zu aktualisieren, ohne den Umweg über die Teilnehmerinitialisierung gehen zu müssen. PUB, HCS und HCA erfordern jeweils die EU des betroffenen Teilnehmers, nicht jedoch den zusätzlichen Versand von Initialisierungsbriefen. Das vereinfacht einerseits das Verfahren der Schlüsseländerung, andererseits entfällt

dadurch die Möglichkeit, Initialisierungsbriefe für die Dokumentation der Keyhistorie von Teilnehmern einzusetzen. Es liegt in der Verantwortung des Kreditinstituts, die Schlüsseländerungen mittels PUB, HCS und HCA so zu dokumentieren, dass sie nachvollziehbar bleiben.

Gegenstand dieses Kapitels ist die detaillierte Beschreibung der Schlüsseländerung mittels PUB, HCS und HCA.

4.6.1.1 Allgemeine Beschreibung

Teilnehmer im Zustand „Bereit“ können ihre öffentlichen Teilnehmerschlüssel unter Verwendung der folgenden systembedingten Auftragsarten aktualisieren:

- **PUB**: Aktualisierung des öffentlichen bankfachlichen Schlüssels
- **HCA:** Aktualisierung des öffentlichen Authentifikationsschlüssels und des öffentlichen Verschlüsselungsschlüssels.
- **HCS:** Aktualisierung des öffentlichen bankfachlichen Schlüssels, des öffentlichen Authentifikationsschlüssels und des öffentlichen Verschlüsselungsschlüssels.

PUB, HCS und HCA sind reguläre Upload-Transaktionen, deren Ablauf (einschließlich des Verhaltens in Fehlersituationen) in Kapitel 5.5 detailliert beschrieben ist. Darin enthalten ist die Abbildung 52, die den Ablauf der bankseitigen Verarbeitung der EBICS-Requests während der Datentransferphase einer Upload-Transaktion beschreibt. Teil dieses Ablaufs ist der Ablaufschritt „Überprüfung und Durchführung des Auftrags“. Für die Auftragsart PUB liefert dieser Schritt die folgenden Fehlercodes:

- EBICS_KEYMGMT_UNSUPPORTED_VERSION_SIGNATURE
  Dieser fachliche Fehler tritt dann auf, wenn die Nutzdaten eine unzulässige Version des Verfahrens zur bankfachlichen Signatur enthalten
- EBICS_KEYMGMT_KEYLENGTH_ERROR_SIGNATURE
  Dieser fachliche Fehler tritt dann auf, wenn die Nutzdaten einen bankfachlichen Schlüssel unzulässiger Länge enthalten
- EBICS_INVALID_ORDER_DATA_FORMAT
  Dieser fachliche Fehler tritt dann auf, wenn die Nutzdaten nicht dem dafür vorgesehenen Format entsprechen (siehe Kapitel 4.4.1.2.5.1)
- EBICS_USER_UNKNOWN
  Dieser technische Fehler tritt dann auf, wenn der Teilnehmer, der Bestandteil der Nutzdaten von PUB ist, kein registrierter Teilnehmer ist
- EBICS_INVALID_USER_STATE
  Dieser technische Fehler tritt dann auf, wenn der Teilnehmer, welcher Bestandteil der Nutzdaten von PUB ist, sich nicht im Zustand „Bereit“ befindet
- EBICS_SIGNATURE_VERIFICATION_FAILED
  Dieser fachliche Fehler tritt auf, wenn die EU des betroffenen Teilnehmers nicht erfolgreich verifiziert werden konnte.

- Für Return Codes, die Zertifikate betreffen, wird auf Anhang 1 verwiesen.

Für HCA liefert der Ablaufschritt „Prüfung und Durchführung des Auftrags“ die folgenden Fehlercodes:

- EBICS_KEYMGMT_UNSUPPORTED_VERSION_ENCRYPTION
  Dieser fachliche Fehler tritt dann auf, wenn die Nutzdaten eine unzulässige Version des Verschlüsselungsverfahrens enthalten
- EBICS_KEYMGMT_UNSUPPORTED_VERSION_AUTHENTICATION
  Dieser fachliche Fehler tritt dann auf, wenn die Nutzdaten eine unzulässige Version des Verfahrens der Authentifikationssignatur enthalten
- EBICS_KEYMGMT_KEY_LENGTH_ERROR_ENCRYPTION
  Dieser fachliche Fehler tritt dann auf, wenn die Nutzdaten einen Verschlüsselungs-schlüssel unzulässiger Länge enthalten
- EBICS_KEYMGMT_KEY_LENGTH_ERROR_AUTHENTICATION
  Dieser fachliche Fehler tritt dann auf, wenn die Nutzdaten einen Authentifikationsschlüssel unzulässiger Länge enthalten
- EBICS_INVALID_ORDER_DATA_FORMAT
  Dieser fachliche Fehler tritt dann auf, wenn die Nutzdaten nicht dem dafür vorgesehenen Format entsprechen (siehe Kapitel 4.4.1.2.5.1)
- EBICS_USER_UNKNOWN
  Dieser technische Fehler tritt dann auf, wenn der Teilnehmer, der Bestandteil der Nutzdaten von HCA ist, kein registrierter Teilnehmer ist
- EBICS_INVALID_USER_STATE
  Dieser technische Fehler tritt dann auf, wenn der Teilnehmer, welcher Bestandteil der Nutzdaten von HCA ist, sich nicht im Zustand „Bereit“ befindet
- EBICS_KEYMGMT_NO_X509_SUPPORT
  Dieser fachliche Fehler tritt auf, wenn ein öffentlicher Schlüssel vom Typ `ds:X509Data` übertragen wurde, das Kreditinstitut jedoch nur den Type `ebics:PubKeyValueType` unterstützt
- EBICS_SIGNATURE_VERIFICATION_FAILED
  Dieser fachliche Fehler tritt auf, wenn die EU des betroffenen Teilnehmers nicht erfolgreich verifiziert werden konnte.
- Für Return Codes, die Zertifikate betreffen, wird auf Anhang 1 verwiesen.

Da HCS eine Kombination aus PUB und HCA ist, sind im Ablaufschritt „Überprüfung und Durchführung des Auftrags“ alle unter PUB und HCA genannten Fehlercodes möglich.

Entweder PUB und HCA oder HCS müssen von dem Teilnehmer eingereicht werden, dessen Schlüssel aktualisiert werden sollen. PUB, HCS und HCA erfordern jeweils genau

eine EU, die von dem Teilnehmer geleistet werden muss, dessen Schlüssel aktualisiert werden sollen. Die Unterschriftsklasse dieser EU ist unerheblich.

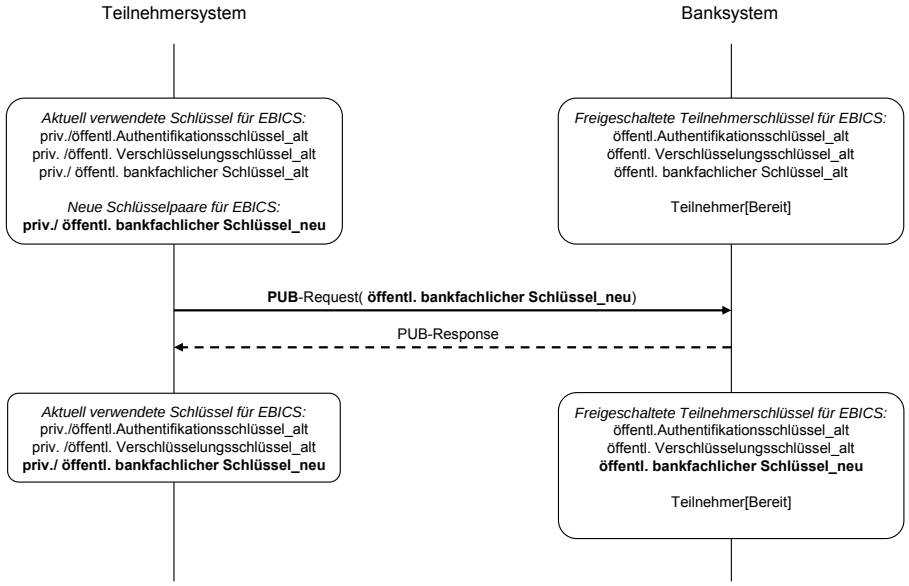


Abbildung 28: Änderung des bankfachlichen Teilnehmerschlüssels mittels PUB

Abbildung 28 stellt den Zustand der öffentlichen Teilnehmerschlüssel und des Teilnehmers vor und nach der Durchführung von PUB dar. Für die Durchführung von PUB gilt:

- Die Nutzdaten, d.h. der neue öffentliche bankfachliche Schlüssel des Teilnehmers, werden komprimiert, verschlüsselt und schließlich base64-kodiert in die EBICS-Nachrichten eingebettet
- Die Nutzdaten werden von dem Teilnehmer mittels EU signiert, dessen öffentlicher bankfachlicher Schlüssel aktualisiert werden soll. Für diese EU wird der alte, zu dem Zeitpunkt jedoch freigeschaltete bankfachliche Schlüssel dieses Teilnehmers verwendet.

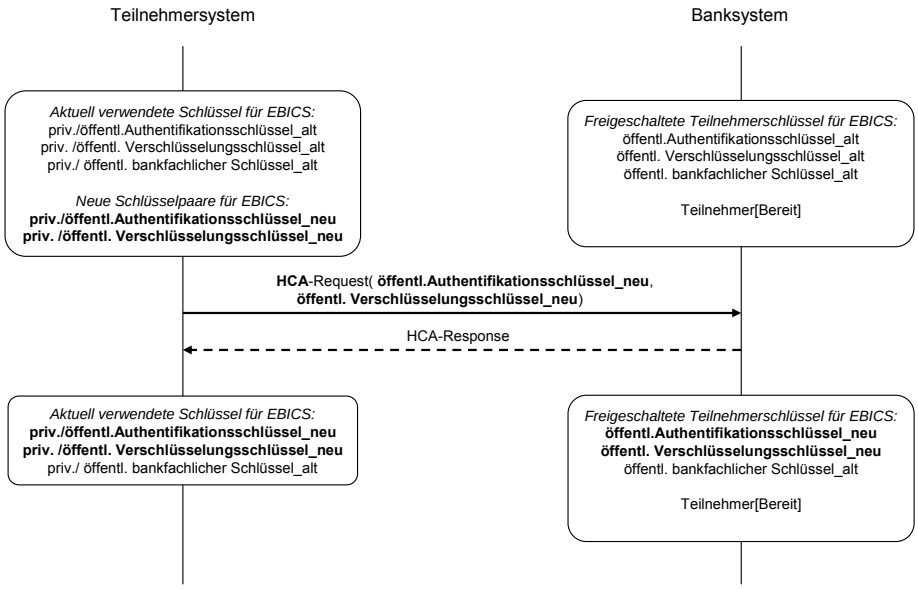


Abbildung 29: Änderung des Authentifikationsschlüssels sowie des Verschlüsselungsschlüssels mittels HCA

In Abbildung 29 ist der Zustand der Teilnehmerschlüssel und des Teilnehmers vor und nach der Durchführung von HCA dargestellt. Zudem gilt für die Durchführung von HCA:

- Die Nutzdaten, d.h. der neue öffentliche Authentifikationsschlüssel des Teilnehmers sowie dessen neuer öffentlicher Verschlüsselungsschlüssel, werden komprimiert, verschlüsselt und schließlich base64-kodiert in die EBICS-Nachrichten eingebettet
- HCA-Requests enthalten die Authentifikationssignatur des betroffenen Teilnehmers oder eines technischen Teilnehmers. Die Authentifikationsunterschrift des betroffenen Teilnehmers wird mit dem alten, zu dem Zeitpunkt jedoch freigeschalteten Authentifikationsschlüssel erzeugt. Die EBICS-Responses des Kreditinstituts enthalten die Authentifikationssignatur des Kreditinstituts.

Mittels HCS werden alle Schlüssel geändert.
HCS ist alternativ zu PUB und HCA zu verstehen, bei denen die Schlüssel für die bankfachliche Elektronische Unterschrift (PUB) und für die Authentifikationssignatur und Verschlüsselung (HCA) getrennt geändert werden. Demnach sieht der Ablauf wie folgt aus:

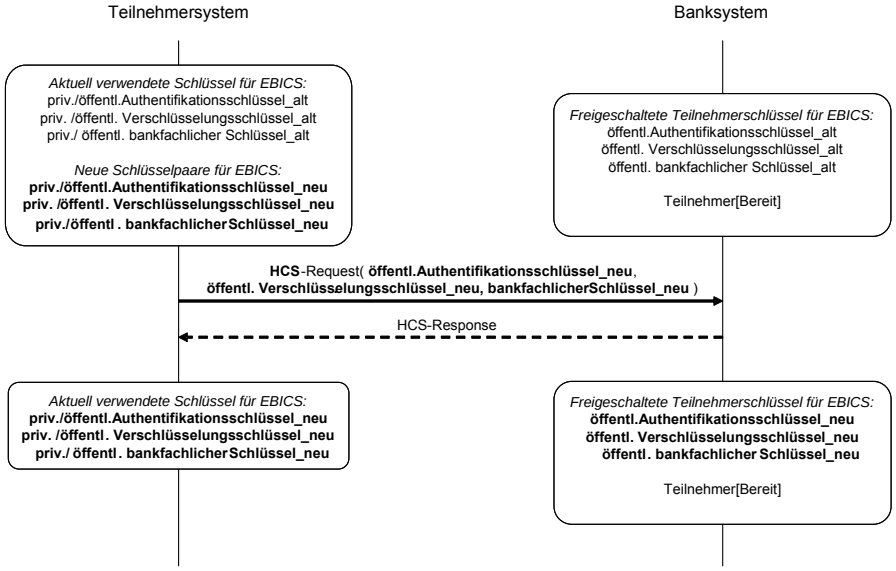


Abbildung 30: Änderung des bankfachlichen Teilnehmerschlüssels sowie Authentifikationsschlüssel und Verschlüsselungsschlüssel mittels HCS

4.6.1.2 Format der Auftragsdaten

PUB unterstützt im Signaturverfahren A004 die Datenstruktur für INI-Dateien (bzw. die Public-Key-Datei, siehe Kapitel 14.2.5.4)
Bei Verwendung der EU in strukturierter Form (ab dem Signaturverfahren A005/A006) sind die Auftragsdaten für PUB ein zu ebics_signature.xsd konformes Instanzdokument, das aus dem Toplevel-Element `SignaturePubKeyOrderData` besteht (analog INI, XML-Darstellung siehe daher auch Kapitel 4.4.1.2.5)

`SignaturePubKeyOrderData` ist mittels XML-Schema wie folgt definiert:

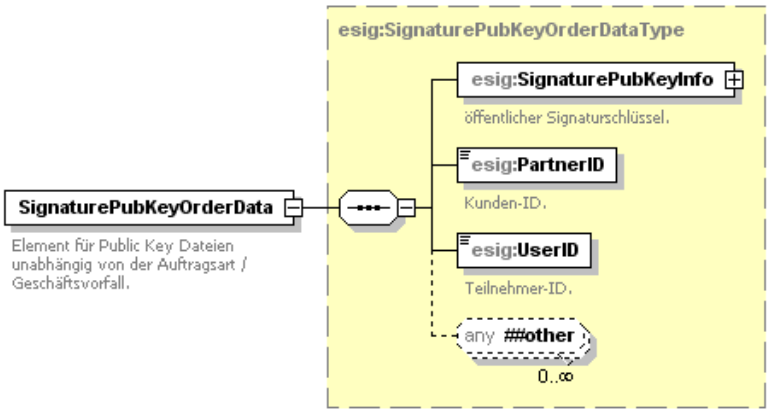


Abbildung 31: Definition des XML-Schema-Elements SignaturePubKeyOrderData der Auftragsdaten für PUB (identisch mit INI, siehe eigenes Kapitel)

Die Auftragsdaten für HCA sind ein zu ebics_orders.xsd konformes Instanzdokument, das aus dem Toplevel-Element `HCARequestOrderData` besteht. `HCARequestOrderData` ist mittels XML-Schema wie folgt definiert:

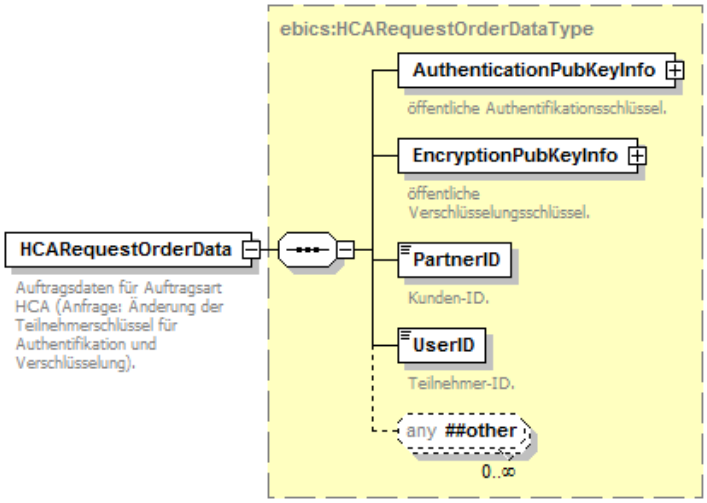


Abbildung 32: Definition des XML-Schema-Elements HCARequestOrderData der Auftragsdaten für HCA

Die Auftragsdaten für HCS sind ein zu ebics_orders.xsd konformes Instanzdokument, das aus dem Toplevel-Element `HCSRequestOrderData` besteht. `HCSRequestOrderData` ist mittels XML-Schema wie folgt definiert:

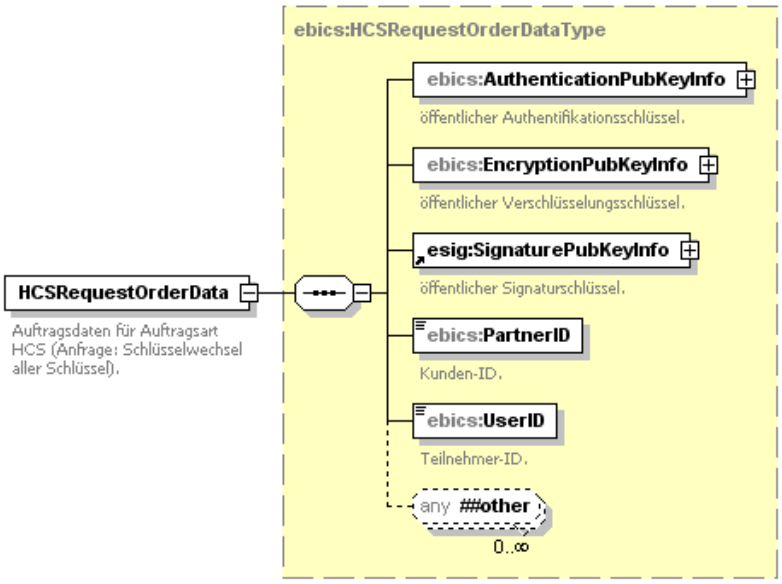


Abbildung 33: Definition des XML-Schema-Elements HCSRequestOrderData der Auftragsdaten für HCS

Die Auftragsdaten von PUB, HCS und HCA werden komprimiert, verschlüsselt und base64-kodiert in den entsprechenden EBICS-Request eingebettet.

4.6.2 Änderung der Bankschlüssel

Der Ablauf für die Aktualisierung der Bankschlüssel ist nicht Bestandteil des vorliegenden Standards. Die Gültigkeitsdauer der Bankschlüssel ist nicht Teil der EBICS-Schnittstelle. Aus Sicht des EBICS-Protokolls existiert zu jedem Zeitpunkt und zu jeder zulässigen Kombination von Verfahren der Authentifikationssignatur, der Verschlüsselung und der EU ein Satz aktuell gültiger Bankschlüssel. In der Version H004 besteht dieser aus genau den folgenden Schlüsseln besteht:

- privater/ öffentlicher Verschlüsselungsschlüssel für das Verfahren E002
- privater/ öffentlicher Authentifikationsschlüssel für das Verfahren X002
- privater/öffentlicher bankfachlicher Schüssel für das Verfahren A004, A005 oder A006.

In EBICS gibt es keine Übergangsfristen, in denen mehrere Schlüssel für dasselbe Verfahren gültig sind. Auf Bankseite geänderte Schlüssel werden in EBICS sofort gültig.

Der Teilnehmer ist für die Abholung der jeweils aktuellen Bankschlüssel über HPB verantwortlich. Nach der Durchführung von HPB ist der Zustand der Bankschlüssel auf Teilnehmerseite gleich „Neu“. In diesem Zustand dürfen die Bankschlüssel (noch) nicht in der Kommunikation über EBICS eingesetzt werden. Das Kreditinstitut MUSS die neuen Schlüssel und/oder deren Hashwerte auf einem zweiten, unabhängigen Kommunikationsweg zugängig machen. Der Teilnehmer MUSS wie bei der erstmaligen Abholung der Bankschlüssel unmittelbar nach der Initialisierung den Abgleich der über unterschiedliche Kommunikationswege erhaltenen Schlüssel vornehmen. Nach der erfolgreichen Überprüfung der Bankschlüssel ist deren Zustand auf Teilnehmerseite „Freigeschaltet“. Im Zustand „Freigeschaltet“ können die Bankschlüssel in der Kommunikation über EBICS verwendet werden.

Um sicherzustellen, dass der Teilnehmer über die aktuellen Bankschlüssel verfügt, sieht der Ablauf einer EBICS-Transaktion (mit Ausnahme von INI, HIA, HPB, HSA) im ersten EBICS-Request die Übermittlung der Hashwerte der öffentlichen Schlüssel des Kreditinstituts (XML-Struktur `ebicsRequest/header/static/BankPubKeyInfo`) vor, über die der Teilnehmer verfügt. Das Banksystem überprüft die Aktualität dieser Schlüssel und liefert das Ergebnis dieser Überprüfung an den Teilnehmer zurück. Sollte einer dieser Schlüssel nicht mehr aktuell sein, wird die Transaktion mit dem technischen Returncode EBICS_BANK_PUBKEY_UPDATE_REQUIRED abgebrochen. Der Teilnehmer muss dann die Bankschlüssel mit HPB abholen.

In der Version „H004“ des EBICS-Protokolls ist die EU der Kreditinstitute nur vorgesehen (siehe Kapitel 3.5.2). Die XML-Struktur `BankPubKeyInfo` enthält in Vorbereitung zukünftiger EBICS-Versionen den Hashwert des öffentlichen bankfachlichen Schlüssels mit der maximalen Häufigkeit gleich 0. Weitere Einzelheiten zur Überprüfung der Hashwerte sind in Kapitel 5.5.1.2 beschrieben.

4.7 Übergang zu größeren Schlüssellängen

Um die Sicherheit des RSA-Verfahrens zu gewährleisten, müssen die Schlüssellängen kontinuierlich erhöht werden. Siehe dazu etwa die regelmäßigen Veröffentlichungen der „Übersicht über geeignete Algorithmen“ der Regulierungsbehörde für Telekommunikation und Post.

Gegenstand dieses Kapitels ist der Übergang zu Schlüsseln größerer Länge in EBICS.

In der Version „H004“ ist für das Signaturverfahren A004 die Länge der bankfachlichen Schlüssel aufgrund der Unterstützung bereits vor der Spezifikation von EBICS bestehender Formate implizit auf 1024 festgelegt. Dabei handelt es sich um:

- **das Format der A004-Datei**. Siehe dazu Kapitel 14.2.5.3.
  Solange nur das Format der A004-Datei verwendet wird, können nur bankfachliche Schlüssel der Länge 1024 genutzt werden

- **das Format der INI-Datei**. Siehe dazu Kapitel 14.2.5.4.
  Dieses Format wird in EBICS „H004“ weiterhin für INI, PUB und HCS verwendet, soweit das Signaturverfahren A004 verwendet wird . Solange nur dieses Format verwendet wird, können nur bankfachliche Schlüssel der Länge 1024 genutzt werden.

Sollen für Teilnehmer bankfachliche Schlüssel der Länge > 1024 eingesetzt werden, so ist die strukturierte Form der EU zu verwenden (siehe auch Kapitel 14.1)

EBICS legt in der Version „H004“ für Authentifikationsschlüssel und Verschlüsselungsschlüssel eine Mindestlänge von 1024 Bit (=1KBit) und eine Maximallänge von 16 KBit fest. Die Mindestlänge muss angepasst werden, sobald Schlüssel dieser Länge aus Sicherheitsgründen nicht mehr verwendet werden sollen. Die Maximallänge muss angepasst werden, sobald Schlüssel der Länge größer als diese Maximallänge erlaubt unterstützt werden sollen.

Die Nutzdatenformate der neuen Auftragsarten des Schlüsselmanagements HIA, HPB, HCA und HCS ermöglichen beliebig große Schlüssellängen. Das bedeutet, dass diese Nutzdatenformate nach einer Erhöhung der Schlüssellängen keiner Anpassung bedürfen.

Neue öffentliche Authentifikationsschlüssel oder Verschlüsselungsschlüssel größerer Länge werden dem Banksystem über HIA, HCS oder HCA genauso übermittelt wie neue Authentifikationsschlüssel gleich bleibender Länge.

Ebenso werden die neuen öffentlichen Schlüssel des Kreditinstituts über HPB abgeholt unabhängig davon, ob die Länge des Authentifikationsschlüssels oder des Verschlüsselungsschlüssels des Kreditinstituts sich verändert hat.

4.8 Migration von DFÜ über FTAM zu EBICS

4.8.1 Allgemeine Beschreibung

Alternativ zum Anfangszustand „Neu“ können Teilnehmer der Teilnehmerverwaltung von EBICS im Zustand „Neu_FTAM“ hinzugefügt werden, wenn die folgenden Bedingungen zutreffen:

- der Teilnehmer verfügt aufgrund seines DFÜ-Zugangs für FTAM bereits über einen gültigen, vom Kreditinstitut freigeschalteten bankfachlichen Schlüssel
- im Zuge der Migration von FTAM nach EBICS soll der vorhandene bankfachliche Schlüssel beibehalten werden.

Zusätzlich zur Teilnehmerinitialisierung gemäß Kapitel 4.4.1 mittels INI und HIA kann die Teilnehmerinitialisierung im Zustand „Neu_FTAM“ mit Hilfe der Auftragsart HSA erfolgen. Analog zu HIA werden mit HSA der öffentliche Authentifikationsschlüssel und der öffentliche Verschlüsselungsschlüssel eines Teilnehmers an ein Kreditinstitut übermittelt. Im Unterschied zu HIA wird mit HSA auch die EU des Teilnehmers über die Auftragsdaten versendet.

Die Unterschriftsklasse dieser EU ist unerheblich, sie muss jedoch von dem Teilnehmer geleistet werden, dessen Schlüssel übermittelt werden. HSA erfordert nicht den zusätzlichen Versand eines Initialisierungsbriefes, da die Authentizität der übermittelten Schlüssel durch die EU des betroffenen Teilnehmers gesichert wird. Es liegt in der Verantwortung der Kreditinstitute, die Teilnehmerinitialisierung mittels HSA so zu dokumentieren, dass die Schlüsselhistorie der Teilnehmer nachvollziehbar ist.

Nach der erfolgreichen Durchführung von HSA befindet sich der betroffene Teilnehmer im Zustand „Bereit“. Die nachfolgenden notwendigen Schritte der Abholung und Überprüfung der Bankschlüssel werden gemäß Kapitel 4.4.2 durchgeführt.

Das Sequenzdiagramm in Abbildung 34 zeigt analog zu Abbildung 13 einen beispielhaften Ablauf der Teilnehmerinitialisierung mittels HSA und der anschließenden Abholung der Bankschlüssel. Die Abbildung stellt gleichzeitig den Zustand der öffentlichen Bankschlüssel auf Teilnehmerseite sowie den Zustand des Teilnehmers und den Zustand der öffentlichen Teilnehmerschlüssel auf Bankseite dar.

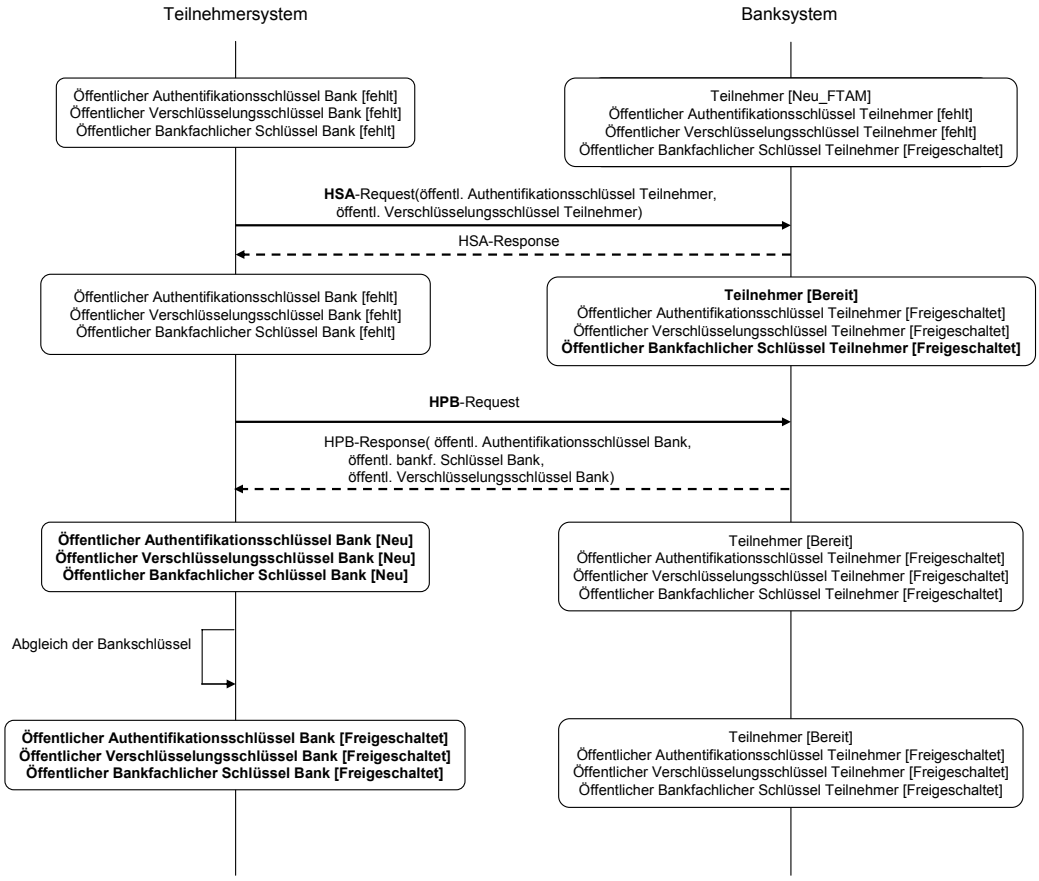


Abbildung 34: Beispielablauf: Teilnehmerinitialisierung mit HSA und anschließende Abholung und Prüfung der Bankschlüssel Abbildung 35 stellt die Erweiterung des Zustandsdiagramms für Teilnehmer dar, die den Anfangszustand „Neu_FTAM“ berücksichtigt. Daraus ist ersichtlich, dass die Teilnehmerinitialisierung von Teilnehmern im Zustand „Neu_FTAM“ sowohl über INI und HIA als auch über HSA durchgeführt werden kann.

Für die Kreditinstitute ist die Unterstützung von HSA optional.

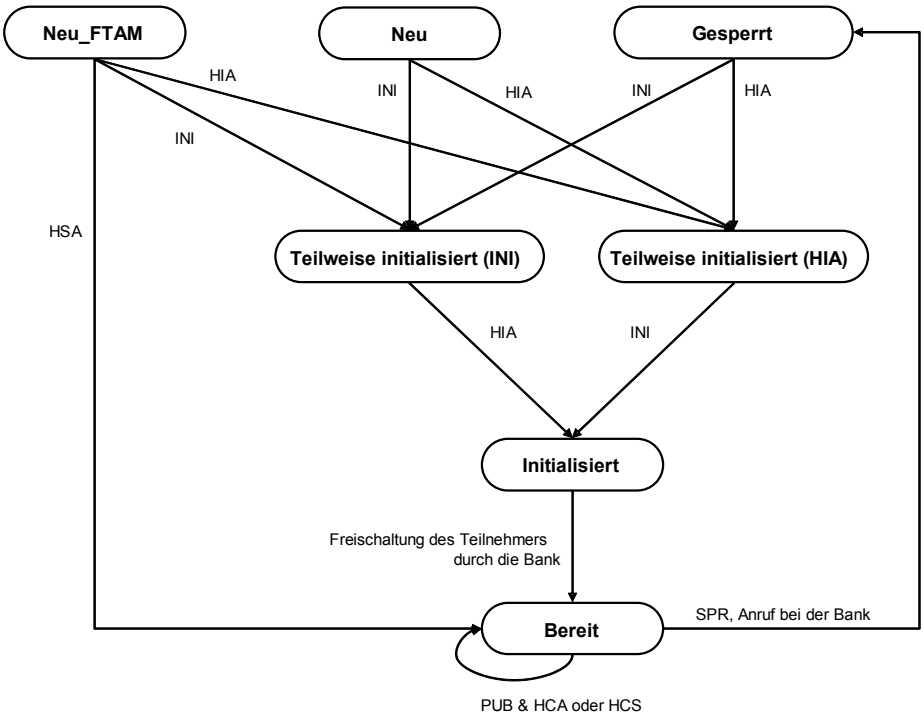


Abbildung 35: Erweitertes Zustandsübergangsdiagramm für Teilnehmer

4.8.2 HSA [optional]

Die Durchführung von HSA ist zulässig, wenn der Zustand des betroffenen Teilnehmers „Neu_FTAM“ ist. Die Übermittlung von HSA-Aufträgen geschieht mit einem einzigen EBICS-Request/ -Response-Paar. Für den EBICS-Request von HSA gilt:

- Der HSA-Request enthält keine Authentifikationssignatur, da der öffentliche Authentifikationsschlüssel des Teilnehmers in diesem Request erst übertragen wird
- Der HSA-Request enthält die Nutzdaten, d.h. den öffentlichen Verschlüsselungsschlüssel und den öffentlichen Authentifikationsschlüssel des Teilnehmers, in unverschlüsselter Form, da dem Teilnehmer der öffentliche Verschlüsselungsschlüssel des Kreditinstituts noch nicht vorliegt.

Das Ablaufdiagramm in Abbildung 36 stellt die bankseitige Verarbeitung dar, die beim Empfang eines HSA-Requests stattfindet. Analog zu HIA werden auch hier Fehlersituationen, die aus einer ungültigen Kombination aus Kunden-/TeilnehmerID oder einem un-

zulässigen Teilnehmerzustand resultieren, nicht direkt an den Sender des HSA-Requests durchgereicht. Stattdessen erhält der Sender den technischen Fehlercode EBICS_INVALID_USER_OR_USER_STATE. HSA liefert keine Fehler der Art „Unbekannter Teilnehmer“ oder „Unzulässiger Teilnehmerzustand“, um potenziellen Angreifern keine genaue Information über die Gültigkeit von TeilnehmerIDs oder den Zustand von Teilnehmern zu geben. Ebenfalls analog zu HIA muss auf Seiten des Kreditinstituts eine interne Protokollierung erfolgen, die die genaue Fehlerursache dokumentiert.

Das Ablaufdiagramm sieht die Überprüfung des Teilnehmerzustands vor, um HSA-Requests bereits auf EBICS-Ebene abzuweisen, wenn der Teilnehmerzustand für HSA nicht zulässig ist. Der einzig zulässige Zustand für HSA ist „Neu_FTAM“. Geprüft wird hier der Zustand des Teilnehmers aus den Headerdaten des Requests. Die Nutzdaten des HSA-Requests (siehe Kapitel 4.8.3.1) enthalten letztlich den Teilnehmer, dessen Authentifikations- und Verschlüsselungsschlüssel übermittelt werden soll. Der Teilnehmer aus den Headerdaten sollte deshalb mit dem Teilnehmer aus den Nutzdaten übereinstimmen. Das EBICS-Protokoll sieht keine Überprüfung dieser Übereinstimmung vor. Vor der eigentlichen Durchführung des Auftrags wird jedoch der Zustand des Teilnehmers (erneut) überprüft, welcher Teil der Nutzdaten von HSA ist.

Die Prüfung und Durchführung eines HSA-Auftrags kann die folgenden Fehlercodes liefern:

- EBICS_KEYMGMT_UNSUPPORTED_VERSION_ENCRYPTION
  Dieser fachliche Fehler tritt dann auf, wenn die Nutzdaten eine unzulässige Version des Verschlüsselungsverfahrens enthalten
- EBICS_KEYMGMT_UNSUPPORTED_VERSION_AUTHENTICATION
  Dieser fachliche Fehler tritt dann auf, wenn die Nutzdaten eine unzulässige Version des Verfahrens der Authentifikationssignatur enthalten
- EBICS_KEYMGMT_KEY_LENGTH_ERROR_ENCRYPTION
  Dieser fachliche Fehler tritt dann auf, wenn die Nutzdaten einen Verschlüsselungsschlüssel unzulässiger Länge enthalten
- EBICS_KEYMGMT_KEY_LENGTH_ERROR_AUTHENTICATION
  Dieser fachliche Fehler tritt dann auf, wenn die Nutzdaten einen Authentifikationsschlüssel unzulässiger Länge enthalten
- EBICS_INVALID_ORDER_DATA_FORMAT
  Dieser fachliche Fehler tritt dann auf, wenn die Nutzdaten nicht dem dafür vorgesehenen Format entsprechen (siehe Kapitel 4.8.3.1)
- EBICS_INVALID_USER_OR_USER_STATE
  Dieser technische Fehler tritt dann auf, wenn die Nutzdaten einen Teilnehmer enthalten, der entweder nicht gültig ist oder dessen Zustand für HSA unzulässig ist. Zulässig ist nur der Teilnehmerzustand Neu_FTAM
- EBICS_KEYMGMT_NO_X509_SUPPORT
  Dieser fachliche Fehler tritt auf, wenn ein öffentlicher Schlüssel vom Typ

`ds:X509Data` übertragen wurde, das Kreditinstitut jedoch nur den Typ `ebics:PubKeyValueType` unterstützt.

- EBICS_INVALID_SIGNATURE_FILE_FORMAT
  Dieser fachliche Fehler tritt auf, wenn die übergebene EU-Datei nicht dem festgelegten Format entspricht.
- EBICS_SIGNATURE_VERIFICATION_FAILED
  Dieser fachliche Fehler tritt auf, wenn die EU des betroffenen Teilnehmers nicht erfolgreich verifiziert werden konnte.

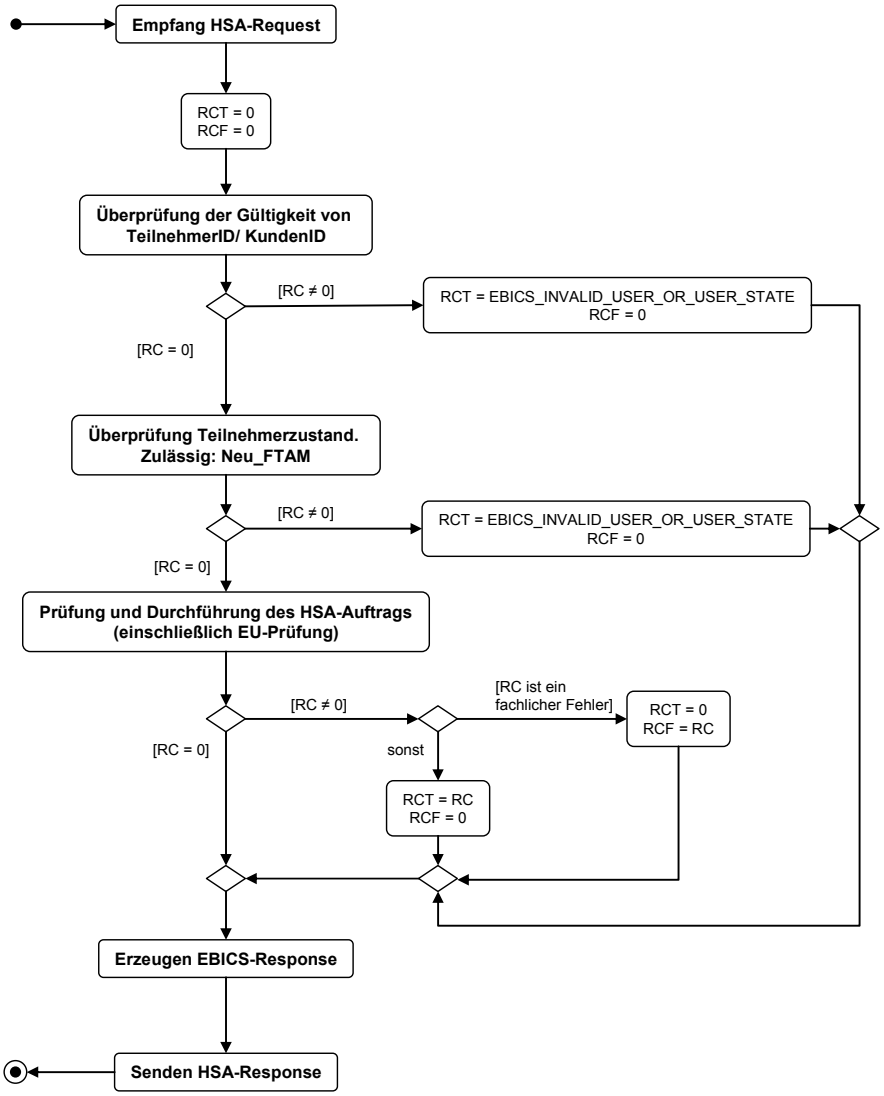


Abbildung 36: Bankseitige Verarbeitung eines HSA-Requests

Die EBICS-Response von HSA enthält keine Authentifikationssignatur des Kreditinstituts, da der öffentliche Authentifikationsschlüssel des Kreditinstituts dem Teilnehmer zur Prüfung noch nicht vorliegt.

HSA-Aufträge müssen von dem Teilnehmer eingereicht und mittels EU signiert werden, dessen Schlüssel übermittelt werden sollen. Dieser ist Bestandteil der Auftragsdaten von HSA. HSA erfordert jeweils genau eine EU, die Unterschriftsklasse dieser EU ist unerheblich.

In Abbildung 34 ist der Zustand der Teilnehmerschlüssel und des Teilnehmers vor und nach der Durchführung von HSA dargestellt.

4.8.3 Beschreibung der EBICS-Nachrichten für HSA

4.8.3.1 Format der Auftragsdaten

Die Auftragsdaten von HSA haben dieselbe Struktur wie die Auftragsdaten von HIA und sind mittels XML-Schema wie folgt definiert:

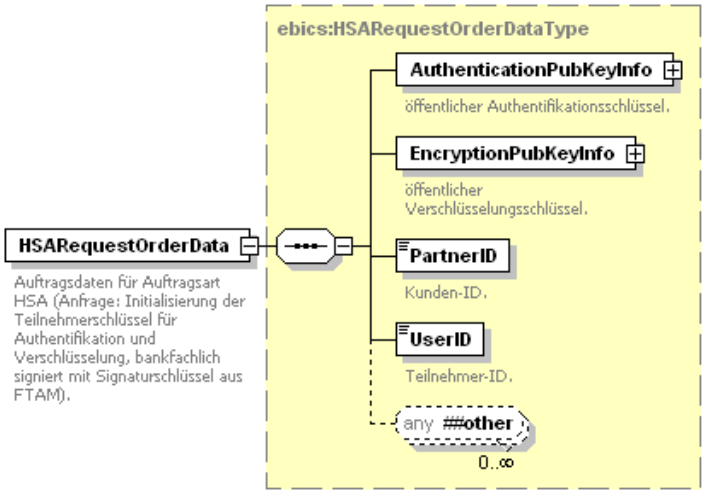


Abbildung 37: Definition des XML-Schema-Elements HSARequestOrderData der Auftragsdaten für HSA

Die Auftragsdaten für HSA sind ein zu ebics_orders_H004.xsd konformes Instanzdokument, das aus dem Toplevel-Element `HSARequestOrderData` besteht. Sie werden komprimiert und base64-kodiert in den entsprechenden EBICS-Request eingebettet.

4.8.3.2 Beschreibung und Beispielnachrichten

Dieses Kapitel beschreibt die EBICS-Nachrichten für die Auftragsart HSA. HSA-Requests sind zu ebics_keymgmt_request_H004.xsd konforme Instanzdokumente bestehend aus dem Toplevel-Element `ebicsUnsignedRequest`. HSA-Responses sind zu ebics_keymgmt_response_H004.xsd konforme Instanzdokumente bestehend aus dem Toplevel-Element `ebicsKeyManagementResponse`.

- Übertragung der folgenden Daten im HSA-Request (analog zu INI, siehe Beispiel in Abbildung 21):
  - Host-ID des EBCIS Bankrechners (ebicsUnsignedRequest/header/static/HostId)
  - Teilnehmer (`ebicsUnsignedRequest/header/static/PartnerID`, `ebicsUnsignedRequest/header/static/UserID`), dessen öffentlicher Authenifikationsschlüssel und öffentlicher Verschlüsselungsschlüssel an das Kreditinstitut übermittelt werden soll
  - (optional) technischer Teilnehmer (`ebicsUnsignedRequest/header/static/PartnerID`, `ebicsUnsignedRequest/header/static/SystemID`) `SystemID` kann in der Nachricht enthalten sein, wenn es sich bei dem Kundensystem um ein Multi-Usersystem handelt. Da HSA-Requests keine Authentifikationssignatur enthalten, ist die Angabe von `SystemID` optional
  - (optional) Information zum Kundenprodukt (`ebicsUnsignedRequest/header/static/Product`)
  - Auftragsart (ebicsUnsignedRequest/header/static/OrderDetails/OrderType) mit der Belegung “HSA”
  - Auftragsattribute (ebicsUnsignedRequest/header/static/OrderDetails/OrderAttribute) mit der Belegung „OZNNN”
  - Sicherheitsmedium für den bankfachlichen Schlüssel des Teilnehmers `(ebicsUnsignedRequest/header/static/SecurityMedium)`
  - EU der Auftragsdaten (ebicsUnsignedRequest/body/DataTransfer/SignatureData)
  - Auftragsdaten (ebicsUnsignedRequest/body/DataTransfer/OrderData).

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsUnsignedRequest
 xmlns="urn:org:ebics:H004"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_keymgmt_request_H004.xsd"
 Version="H004" Revision="1">
 <header authenticate="true">
   <static>
     <HostID>EBIXHOST</HostID>
```

```
      <PartnerID>KUNDE001</PartnerID>
      <UserID>TLN100</UserID>
      <OrderDetails>
        <OrderType>HSA</OrderType>
        <OrderID>A201</OrderID>
        <OrderAttribute>OZNNN</OrderAttribute>
      </OrderDetails>
      <SecurityMedium>0000</SecurityMedium>
    </static>
    <mutable/>
  </header>
  <body>
    <DataTransfer>
      <!--Zu ebics_orders_H004.xsd konformaes (XML-)Instanzdokument mit Wurzelelement
UserSignatureData, kompimiert und base64-kodiert-->
      <SignatureData>
        …
      </SignatureData>
      <!--Zu ebics_orders.xsd konformes (XML-)Instanzdokument mit Wurzelelement
HSARequestOrderData, komprimiert und base64-kodiert-->
      <OrderData>
        …
      </OrderData>
    </DataTransfer>
  </body>
</ebicsUnsignedRequest>
```

Abbildung 38: EBICS-Request für die Auftragsart HSA

- Übertragung der folgenden Daten in der HSA-Response (analog zu HIA, siehe Beispiel in Abbildung 22):
  - bankfachlicher Returncode (ebicsKeyManagementResponse/body/ReturnCode)
  - Auftragsnummer (ebicsKeyManagementResponse/header/mutable/OrderID). Diese Nummer wird vom Banrechner automatisch vergeben.
  - technischer Returncode (ebicsKeyManagementResponse/header/mutable/ReturnCode)
  - technischer Reporttext (ebicsKeyManagementResponse/header/mutable/ReportText)
  - (optional) Zeitstempel der letzten Aktualisierung der Bankparameter (`ebicsKeyManagementResponse/body/TimestampBankParameter`).

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsKeyManagementResponse
 xmlns="urn:org:ebics:H004"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_keymgmt_response_H004.xsd"
 Version="H004" Revision="1">
 <header authenticate="true">
   <static/>
   <mutable>
     <OrderID>A201</OrderID>
```

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

Abbildung 39: EBICS-Response für die Auftragsart HSA

4.9 Zusammenfassung

In der nachfolgenden Tabelle sind die wichtigsten Merkmale der systembedingten Auftragsarten des Schlüsselmanagements zusammengefasst:

Auftragsart	Auftragsdatenformat	Auftrags Attribut	Authentifikationssignatur Teilnehmer / Kreditinstitut	EU der Auftragsdaten
INI	INI-Datei (gemäß Kapitel 14)	DZNNN	nein/nein	nein
HIA	ebics: HIARequestOrder» Data	DZNNN	nein/nein	nein
HSA	ebics: HSARequestOrder» Data	OZNNN	nein/nein	ja
HPB	ebics: HPBRequestOrder» Data	DZHNN	ja/nein	nein
PUB	siehe INI	OZHNN	ja/ja	ja
HCA	ebics: HCARequestOrder» Data	OZHNN	ja/ja	ja
HCS	ebics: HCSRequestOrder» Data	OZHNN	ja/ja	ja
H3K	ebics: H3KRequestOrder» Data	OZNNN	nein/nein	ja (Zertifikat)
SPR	--	UZHNN	ja/ja	ja

5 EBICS-Transaktionen

Die Beschreibungen und Festlegungen dieses Kapitels gelten für alle Auftragsarten mit Ausnahme der nachfolgenden systembedingten Auftragsarten des Schlüsselmanagements: INI, HIA, HSA, HPB, PUB, SPR, HCA, H3K und HCS.

5.1 Allgemeine Festlegungen

5.1.1 EBICS-Transaktionen

EBICS-Transaktionen dienen der Übermittlung von Aufträgen an das bankfachliche Zielsystem. Entsprechend der Einteilung von Aufträgen in Sende- und Abholaufträge unterscheidet EBICS zwischen Upload- und Download-Transaktionen: Upload-Transaktionen übertragen bankfachliche Nutzdaten und/oder bankfachliche Unterschriften an das bankfachliche Zielsystem, umgekehrt werden im Rahmen einer Download-Transaktion bankfachliche Nutzdaten und/oder bankfachliche Unterschriften vom bankfachlichen Zielsystem abgeholt.

5.1.2 Transaktionsphasen und Transaktionsschritte

Jede EBICS-Transaktion durchläuft unterschiedliche Transaktionsphasen. Die Phasen einer Upload-Transaktion sind Initialisierung und Datentransfer, die Phasen einer Download-Transaktion Initialisierung, Datentransfer und schließlich Quittierung. Eine Transaktionsphase kann aus einem oder mehreren zusammenhängenden Transaktionsschritten bestehen, wobei unter einem Transaktionsschritt ein Paar aus EBICS-Request und zugehöriger EBICS-Response zu verstehen ist. So umfasst die Initialisierung den ersten Transaktionsschritt, der Datentransfer kann sich hingegen über mehrere weitere Transaktionsschritte erstrecken, in denen jeweils ein Nutzdatensegment übertragen wird.

EBICS-Transaktionen können aus einem einzigen Transaktionsschritt bestehen, beispielsweise wenn sie zu einem Auftrag nur die bankfachlichen elektronischen Unterschriften übermitteln.

5.1.3 Durchführung der Aufträge

5.1.3.1 Zeitliche Abhängigkeiten zwischen Übermittlung und Durchführung von Sendeaufträgen

EBICS unterstützt die zeitliche Entkopplung der Einreichung von bankfachlichen Sendeaufträgen über EBICS von deren eigentlichen Ausführung auf den Backendsystemen des Kreditinstituts. Die innerhalb einer EBICS-Transaktion eingereichten EUs und Nutzdatensegmente werden zunächst vorverarbeitet. Diese **Vorverarbeitung** ist nicht Bestandteil von EBICS, sie ist abhängig von der Implementierung des Banksystems, die Zwischenspeicherung der Nutzdatensegmente ist beispielsweise ein Teil davon. Nach der Übertragung des

letzten Nutzdatensegments werden die gesamten Nutzdaten, Auftragsparameter sowie EUs zunächst an eine Komponente des Banksystems weitergereicht, deren Verantwortlichkeit in der **Verwaltung offener Aufträge** liegt. Die Realisierung ist abhängig von der Implementierung des Banksystems, sie ist nicht Bestandteil von EBICS.

Im Unterschied zu den bankfachlichen Sendeaufträgen wird gefordert, dass die **Durchführung** der Sendeaufträge systembedingter Auftragsarten vor dem Senden der letzten EBICS-Response der Upload-Transaktion abgeschlossen sein MUSS. Diese Forderung gilt neben den Auftragsarten des Schlüsselmanagements auch für die Abholaufträge der Auftragsarten der VEU, um das Verfahren der Verteilten EU möglichst effizient abwickeln zu können und beteiligten Teilnehmern einen möglichst aktuellen Status der verteilten EUs zu einem Auftrag liefern zu können.

5.1.3.2 Zeitliche Abhängigkeiten zwischen Übermittlung und Durchführung von Abholaufträgen

Die Abholdaten sind Bestandteil der EBICS-Responses des Kreditinstituts. Mit jedem EBICS-Transaktionsschritt stellt das Banksystem ein weiteres Nutzdatensegment bereit. Um den Abholvorgang zu beschleunigen, können die Abholdaten vom Banksystem im Voraus erzeugt (wie beispielsweise im Falle von Kontoauszügen) oder erst auf Anfrage hin generiert werden.

5.1.4 Transaktionsverwaltung

Die Steuerung des Ablaufs einer EBICS-Transaktion obliegt im Normalfall dem Kundensystem, die einzelnen Transaktionsschritte einer EBICS-Transaktion werden jeweils vom Kundensystem initiiert. Das Banksystem kann in Sonderfällen den Ablauf einer Transaktion ebenfalls steuern, z.B. indem es dem Kundensystem im Recovery-Fall einen möglichen Wiederaufsetzpunkt mitteilt.

Im Banksystem müssen die EBICS-Transaktionen verwaltet werden, um Folgendes zu ermöglichen:

- Zuordnung der einzelnen Transaktionsschritte zu einer bestimmten EBICS-Transaktion
- Aufzeichnung des Ablaufs der EBICS-Transaktion zur Verwaltung der Transaktionszustände mit dem Ziel, das Fortschreiten der EBICS-Transaktion sicherzustellen.
- Wiederaufsetzen einer EBICS-Transaktion.

Daraus ergeben sich die folgenden Verantwortlichkeiten für die EBICS-Transaktionsverwaltung des Banksystems:

- Erzeugung von EBICS-Transaktionen während der Transaktionsinitialisierung. Siehe dazu Kapitel 5.2.

- Abbruch von EBICS-Transaktionen, wenn die Weiterführung aufgrund von Fehlersituationen nicht sinnvoll oder nicht möglich ist
- Beenden von EBICS-Transaktionen, wenn alle Transaktionsschritte erfolgreich durchgeführt werden konnten
- Kontrolle des Ablaufs von EBICS-Transaktionen, um deren Ablauf gemäß Kapitel 5.5.1 sicherzustellen
- Unterstützung des Verfahrens zum Wiederaufsetzen von EBICS-Transaktionen gemäß Kapitel 5.5.2 und 5.6.2, falls das Banksystem Recovery unterstützt.

5.2 Zuordnung von EBICS-Request zu EBICS-Transaktion

Die erste Phase jeder EBICS-Transaktion ist die Initialisierung. Sie wird durch den ersten EBICS-Request der Transaktion ausgelöst und umfasst:

- Prüfungen, deren erfolgreiche Durchführung die notwendige Voraussetzung für die Annahme des Auftrags durch das Kreditinstitut ist
- weitere Verarbeitungsschritte, die notwendig sind, um EBICS-Transaktionen, die aus mehr als einem Transaktionsschritt bestehen, in die Transaktionsverwaltung aufzunehmen.

Beispiele solcher Prüfungen sind die Prüfung der Zustands und der Auftragsartberechtigung des Teilnehmers, der den Auftrag einreicht. Der genaue Umfang dieser Prüfungen/ Verarbeitungsschritte ist für Upload-Transaktionen in Kapitel 5.5.1.2.1 und für Download-Transaktionen in Kapitel 5.6.1.2.1 beschrieben.

Wurden alle notwendigen Prüfungen erfolgreich durchgeführt und besteht die Transaktion aus mehreren Transaktionsschritten, so erzeugt die Transaktionsverwaltung des Banksystems eine EBICS-Transaktion mit einer banksystemweit eindeutigen TransaktionsID (Details zur Generierung der TransaktionsID sind im Anhang (Kapitel 11.6) enthalten). Diese wird dem Teilnehmer über die Antwortnachricht des Kreditinstituts mitgeteilt. Die Transaktionsverwaltung des Banksystems ordnet dieser Transaktion die folgenden Daten fest zu, die Bestandteil der Headerdaten des EBICS-Requests sind:

- KundenID/ TeilnehmerID/ technische TeilnehmerID
- Auftragsart
- Auftragsnummer
- Auftragsattribute
- Auftragsparameter
- Auftragsnummer
  Die Auftragsnummer ist nur vorhanden wenn sich die Datei, die zur Bank übertragen wird, auf einen Auftrag bezieht, zu dem es schon eine Auftragsnummer gibt

(Übertragung einer EU-Datei mit Auftragsattributen UZHNN), um diese Dateien abzugleichen. Grundsätzlich ist die Auftragsnummer ein Teil der Header-Daten der EBICS-Response (von Sendevorgängen). Sie wird vom Bankrechner automatisch generiert.

Diese Daten sind der Transaktion fest zugeordnet und können im Verlauf der Transaktion nicht verändert werden.

Außerhalb der Initialisierung enthalten EBICS-Requests diese TransaktionsID für die Zuordnung zur passenden EBICS-Transaktion. Insgesamt enthalten sie die folgenden den Transaktionsschritt identifizierenden Elemente:

- die banksystemweit eindeutige TransaktionsID
- die Transaktionsphase (Initialisierung, Datentransfer, Quittierung) innerhalb der Transaktion
- die laufende Nummer des Datensegments bankfachlicher Daten, falls es sich um die Transaktionsphase Datentransfer handelt.

Die detaillierte Beschreibung des Aufbaus der EBICS-Requests von Upload-Transaktionen und Download-Transaktionen ist in den Kapiteln 5.5.1.1 bzw. 5.6.1.1 enthalten.

5.3 Vorabprüfung von Aufträgen [optional]

Um zu vermeiden, dass ein Teilnehmer große Datenbestände an das Banksystem überträgt und auf Bankseite erst nach der Übertragung festgestellt wird, dass die Unterzeichner des Sendeauftrags nicht über die erforderlichen Berechtigungen verfügen, KANN das Banksystem optional die Funktionalität der Vorabprüfung unterstützen. Die Angabe, ob ein Banksystem Vorabprüfungen durchführt, ist Bestandteil seiner abrufbaren Bankparameter (siehe dazu Kapitel 12.2). Wird die Vorabprüfung von Sendeaufträgen unterstützt, so ist der Umfang der Vorabprüfung die Entscheidung der einzelnen Kreditinstitute. Möglich ist die Unterstützung einer oder mehrerer der folgenden Prüfungen:

- **Kontoberechtigungsprüfung**
  Mit Hilfe der Kontoberechtigungsprüfung wird überprüft, ob für jeden Unterzeichner die folgende Bedingung erfüllt ist:
  - der Unterzeichner hat das Recht, eine EU der Mindestklasse „B“ für Aufträge der gegebenen Auftragsart für jedes der Auftraggeberkonten des gegebenen Auftrags zu leisten.
- **Limitprüfung**
  Durch die Limitprüfung wird sichergestellt, dass für jeden Unterzeichner die folgende Bedingung erfüllt ist:

- der Unterzeichner hat das Recht, eine EU der Mindestklasse „B“ für Aufträge der gegebenen Auftragsart und in der jeweils vorliegenden Höhe für jedes der Auftraggeberkonten des gegebenen Auftrags zu leisten.

- **Prüfung der EUs**
  Die EU-Prüfung verifiziert die EUs der Unterzeichner des Auftrags und überprüft, ob die EUs jeweils von unterschiedlichen Teilnehmern stammen.

Für eine erfolgreiche Vorabprüfung benötigt der Teilnehmer eine der Unterschriftklassen ‚E', ‚A' oder ‚B'. Wenn Aufträge nur eingereicht werden (also Unterschriftsklasse ‚T'), wird die Vorabprüfung nicht durchlaufen. Der Returncode EBICS_SIGNATURE_VERIFICATION_FAILED wird zurückgeliefert, wenn die Signatur nicht gültig ist.

Wenn der Unterzeichner den vorliegenden Auftrag bereits unterschrieben hat, wird der Returncode EBICS_DUPLICATE SIGNATURE zurückgeliefert.

Die Vorabprüfung eines Sendeauftrags ist Teil des ersten Transaktionsschritts im Rahmen der entsprechenden Upload-Transaktion. Das Ergebnis der Vorabprüfung steht im bankfachlichen Returncode der entsprechenden EBICS-Response des ersten Transaktionsschritts. Die Vorabprüfung erfolgt vor der Übertragung der Nutzdaten des Auftrags auf Basis der Angaben des Kundensystems über die noch ausstehenden Auftragsdaten. Sie ersetzt nicht die entsprechenden Prüfungen auf Grundlage der eigentlichen Nutzdaten nach deren Übertragung an das Banksystem.

Das Kundensystem KANN den Umfang der Vorabprüfungen weiter einschränken. Die Vorabprüfungen Kontoberechtigungsprüfung, Limitprüfung oder bankfachliche EU-Prüfung werden bankseitig nur dann ausgeführt, wenn die dazu notwendigen Daten vom Kundensystem zur Verfügung gestellt werden. Die Daten der Vorabprüfung werden im ersten EBICS-Request einer Upload-Transaktion über das (optionale) Element `ebicsRequest/body/PreValidation` (siehe ebics_request_H004.xsd) übermittelt, dessen Typdefinition in Abbildung 40 dargestellt ist. Dieser Typ heißt `PreValidationRequestType` (siehe ebics_types_H004.xsd) und ist eine Liste aus den folgenden optionalen Elementen:

- `DataDigest`
  Dieses Element enthält den Hashwert der Auftragsdaten, die von den Unterzeichnern des Auftrags mittels Transportunterschrift oder bankfachlicher EU signiert wurden. Bei der Vorabprüfung eines Auftrags erfolgt die Prüfung der EUs allein auf Basis dieses Hashwerts, dessen Korrektheit zum Zeitpunkt der Vorabprüfung nicht überprüft werden kann. Für das von den Unterzeichnern des Auftrags verwendete (und bzgl. EBICS von der Bank unterstützte) Signaturverfahren kann ein Hashwert eingestellt werden, welcher im Hashverfahren des jeweiligen Signaturverfahrens zu bilden ist. Das zugehörige Signaturverfahren wird anhand des Attributs `SignatureVersion` identifiziert. `DataDigest` kann mehrfach vorkommen, wenn die Unterzeichner verschiedene Unterschriftsversionen verwenden (das ist der Fall, wenn nicht alle Teilnehmer eines Kunden mit dem gleichen Signaturverfahren signieren. Daher ist die korrekte Befüllung des Attributes `SignatureVersion`

Pro `DataDigest` wichtig (default A004).

- `AccountAuthorisation`
  Dieses Element enthält ein Auftraggeberkonto des gegebenen Auftrags.
  Es müssen ein oder zwei Kontonummern angegeben werden unter:
  - AccountAuthorisation/AccountNumber:
  Hier ist die Angabe des deutschen und/oder internationalen Formats (IBAN) möglich
  - AccountAuthorisation/NationalAccountNumber:
  Hier ist die Angabe in freiem Format für Kontonummern möglich, die weder den deutschen noch den nationalen Vorgaben entsprechen.

  Es müssen eine oder zwei Bankleitzahlen angegeben werden unter:
  - AccountAuthorisation/BankCode
  Hier ist die Angabe des deutschen und/oder internationalen Formats (BIC) möglich
  - AccountAuthorisation/NationalBankCode
  Hier ist die Angabe in freiem Format für Bankleitzahlen möglich, die weder den deutschen noch den nationalen Vorgaben entsprechen.

  Optional kann auch der Kontoinhaber (`AccountAuthorisation/AccountHolder`) mitgeliefert werden. Diese Kontoinformationen werden von der Kontoberechtigungsprüfung und der Limitprüfung benötigt. Für die Limitprüfung ist zudem die Angabe der Summe der Einzelaufträge zu dem gegebenen Auftraggeberkonto erforderlich.. Dieser Betrag ist in dem (optionalen) Element `AccountAuthorisation/Amount` enthalten. Die Währung des Betrags ist der Wert des optionalen Attributs `AccountAuthorisation/Amount@Currency`, falls dieses vorhanden ist. Sonst ist die Währung der Wert des Attributs `AccountAuthorisation@Currency`, welches die Währung des Kontos beinhaltet.

Die einzelnen Vorabprüfungen benötigen die TeilnehmerID/ KundenID jedes einzelnen Unterzeichners. Diese sind Bestandteil des XML-Typs OrderSignature, welcher zur Darstellung einzelner EUs verwendet wird. Für weitere Einzelheiten zur Einbettung von EUs in EBICS-Nachrichten siehe Kapitel 3.5.

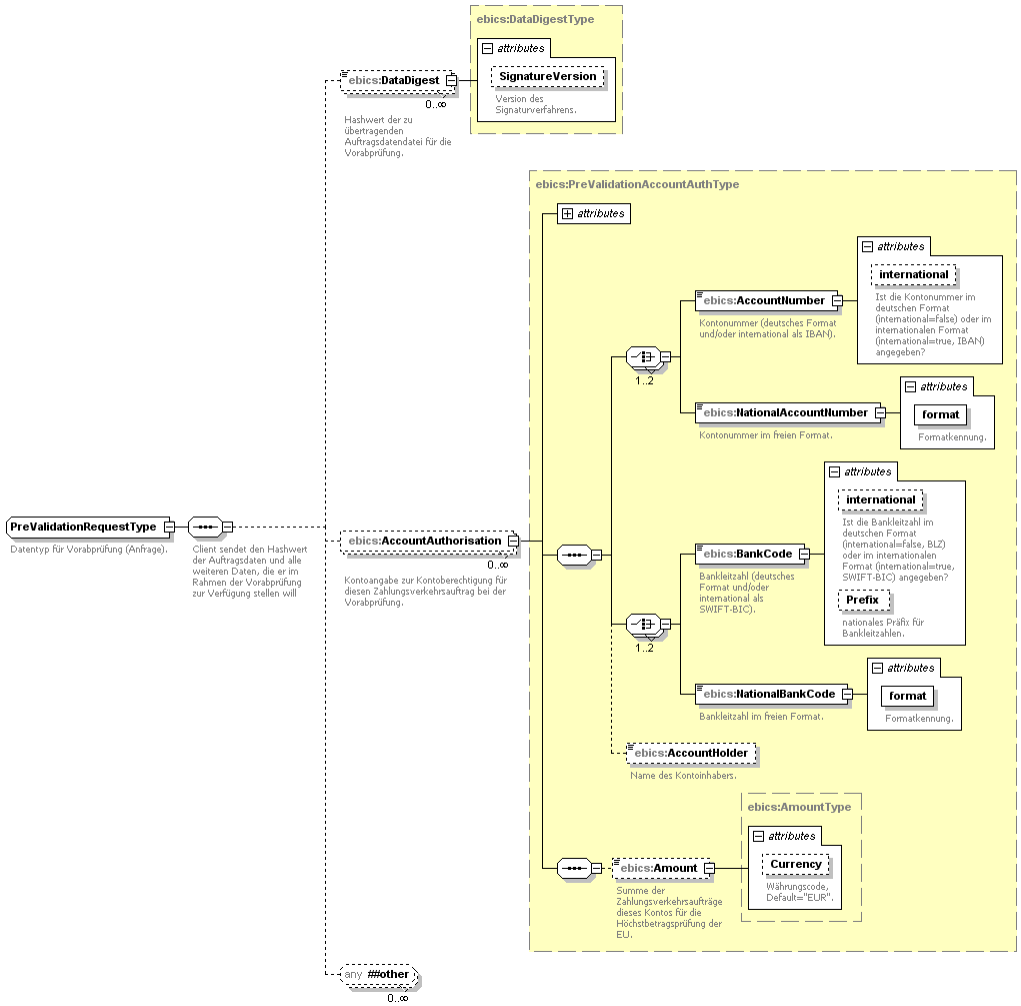


Abbildung 40: XML-Schema-Typdefinition für die Übermittlung der Daten zur Vorabprüfung eines Auftrags

5.4 Recovery – Wiederaufsetzen von Transaktionen [optional]

In diesem Kapitel werden die Grundzüge der Recovery-Vorgehensweise beschrieben, die sowohl für Upload- als auch für Download-Transaktionen gelten.

Mit dem Recovery-Mechanismus von EBICS wird erreicht, dass bereits vom Kunden- oder Banksystem empfangene Auftragsdaten einer Transaktion nicht erneut übertragen werden müssen, wenn eine der folgenden Fehlersituationen eintritt:

- Transportfehler

- Verarbeitungsfehler der EBICS-Nachricht, die die Auftragsdaten enthält:
  Bei Upload-Transaktionen handelt es sich um Verarbeitungsfehler der EBICS-Requests, die auf Bankseite auftreten können, bei Download-Transaktionen um Verarbeitungsfehler der EBICS-Responses, die auf Seite des Kunden/ Teilnehmers auftreten. Beispielsweise können Fehler bei der (Zwischen-) Speicherung der Nutzdaten auftreten.

Recovery ist ein wichtiger Aspekt des Protokolls, da die Größenordnung der Auftragsdaten durchaus im Bereich von mehreren hundert MegaByte liegen kann.

Der Mechanismus setzt die Kenntnis der TransaktionsID der betroffenen EBICS-Transaktion voraus und basiert auf der Definition von **Wiederaufsetzpunkten** von Transaktionen:

- Bei Upload-Transaktionen ist der Wiederaufsetzpunkt der letzte Transaktionsschritt im Rahmen der Transaktion, dessen EBICS-Request aus Sicht des Banksystems erfolgreich empfangen und dessen zugehörige EBICS-Response erfolgreich versendet wurde. Der Wiederaufsetzpunkt wird bestimmt durch den Zustand der Transaktion auf dem Banksystem.
- Bei Download-Transaktionen kann es mehrere Wiederaufsetzpunkte geben, dabei handelt es sich um all die bisherigen Transaktionsschritte der betroffenen Transaktion, deren EBICS-Request aus Sicht des Banksystems bereits erfolgreich empfangen und deren zugehörige EBICS-Response erfolgreich versendet wurde.

Nach Transport- oder Verarbeitungsfehlern kann ein Wiederaufsetzpunkt verwendet werden, um Transaktionen mit dem Transaktionsschritt fortzusetzen, der in der sequenziellen Abfolge der Transaktionsschritte auf diesen Wiederaufsetzpunkt folgt.

Jeder EBICS-Request zu einer offenen Transaktion, der nicht zum Zustand dieser Transaktion passt, wird von der EBICS-Transaktionsverwaltung auf Bankseite als Recovery-Versuch gewertet.

Um ein Fortschreiten der EBICS-Transaktionen zu gewährleisten, MUSS die Anzahl der möglichen Recovery-Versuche je Transaktion durch einen Maximalwert begrenzt werden. Die Transaktionsverwaltung des Banksystems ist verantwortlich für die Verwaltung des entsprechenden Zählers je Transaktion. Transaktionen, deren Zähler die erlaubte Grenze überschreitet, werden durch die Transaktionsverwaltung des Banksystems abgebrochen. Zusätzlich KANN das Banksystem je Teilnehmer die Anzahl der offenen Transaktionen mit einem positiven Recovery-Zähler durch einen Maximalwert begrenzen. Der Zähler für die bereits erfolgten Recovery-Versuche je Transaktion und/oder der Zähler der offenen Transaktionen im Recovery-Modus je Teilnehmer sowie die erlaubten Maximalzahlen sind nicht Bestandteil der EBICS-Nachrichten. Sie sind vielmehr Teil der Implementierung der EBICS-Transaktionsverwaltung auf der Bankseite.

Analog dazu begrenzt die Transaktionssteuerung des Kundensystems die Anzahl der Versuche, einen bestimmten Transaktionsschritt einer EBICS-Transaktion erfolgreich

durchzuführen. Zähler und erlaubte Maximalzahl sind auch in diesem Fall nicht Bestandteil des EBICS-Nachrichten, sondern lediglich der Implementierung der Transaktionssteuerung auf Kundenseite.

Details zum Wiederaufsetzen von Upload- sowie Download-Transaktionen sind in den Kapiteln 5.5.2 bzw. 5.6.2 enthalten.

5.5 Upload-Transaktionen

5.5.1 Ablauf von Upload-Transaktionen

Der Ablauf einer Upload-Transaktion ist in Abbildung 41 anhand eines Sequenzdiagramms dargestellt. Die Übertragung der Nutzdatensegmente erfolgt innerhalb einer Schleife, die abgebrochen wird, sobald das letzte Nutzdatensegment übertragen wurde (beachte Teilausdruck „[letztes Datensegment wurde übertragen]“ der Abbruchbedingung). Der Ablauf verdeutlicht, dass innerhalb einer Upload-Transaktion nicht notwendigerweise auch Nutzdaten übertragen werden müssen (beachte Teilausdruck „[OrderAttribute == „UZHNN“]“ der Abbruchbedingung). Das ist der Fall, wenn im Rahmen der Upload-Transaktion nur bankfachliche EUs zu einem bereits bestehenden Auftrag übertragen werden.

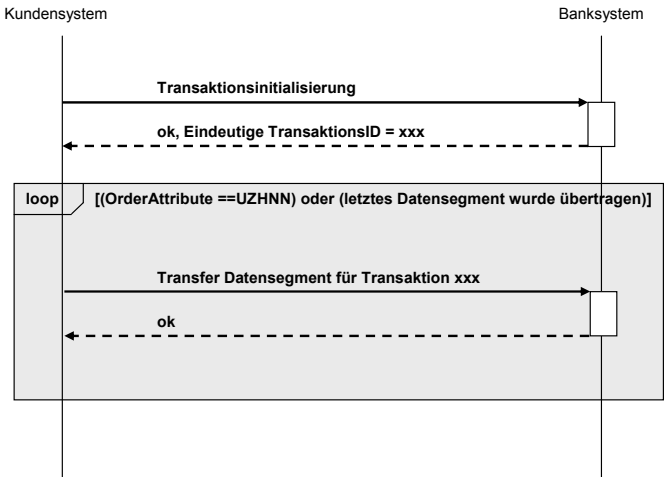


Abbildung 41: Fehlerfreier Ablauf einer Upload-Transaktion

5.5.1.1 Beschreibung der EBICS-Nachrichten

Die nachfolgende Beschreibung der Transaktionsschritte einer Upload-Transaktion verwendet zur Verdeutlichung Beispielnachrichten für die Durchführung eines IZV-Auftrags. Sie nimmt Bezug auf Elemente dieser Beispielnachrichten unter Verwendung der XPath-Notation.

Die folgenden Kapitel beschreiben die Nachrichten der einzelnen Transaktionsphasen. Es werden die Daten aufgezählt, die Bestandteil dieser Nachrichten sind. Daten, die grundsätzlich optional sind, werden mit „(optional)“ gekennzeichnet. Daten, die nur unter bestimmten Voraussetzungen fehlen dürfen, werden stattdessen mit „(bedingt)“ gekennzeichnet. Optionale XML-Elemente, die in der Beschreibung einer EBICS-Nachricht zu einer bestimmten Transaktionsphase fehlen, dürfen in dieser EBICS-Nachricht nicht vorkommen. Optionale XML-Elemente, die in der Beschreibung einer EBICS-Nachricht zu einer bestimmten Transaktionsphase vorkommen, MÜSSEN in dieser EBICS-Nachricht immer entsprechend gesetzt sein.

EBICS-Requests von Upload-Transaktionen sind zu ebics_request_H004.xsd konforme (XML-) Instanzdokumente bestehend aus dem Toplevel-Element `ebicsRequest`, welches in ebics_request_H004.xsd deklariert ist. EBICS-Responses von Upload-Transaktionen sind zu ebics_response_H004.xsd konforme Instanzdokumente, die aus dem Toplevel-Element `ebicsResponse` bestehen, das wiederum in ebics_response_H004.xsd deklariert ist.

5.5.1.1.1 EBICS-Nachrichten der Transaktionsinitialisierung

- **Übertragung der folgenden Daten im EBICS-Request (siehe** Abbildung 42**):**
  - Host-ID des EBICS Bankrechners (`ebicsRequest/header/static/HostID`)
  - Transaktionsphase (`ebicsRequest/header/mutable/TransactionPhase`) mit der Belegung „Initialisation“
  - Kombination aus Nonce und Timestamp, erforderlich für die Vermeidung von Replay alter EBICS-Nachrichten (`ebicsRequest/header/static/Nonce`, `ebicsRequest/header/static/Timestamp`)
  - Anzahl zu übertragender Datensegmente (`ebicsRequest/header/static/NumSegments`)
  - Teilnehmer (`ebicsRequest/header/static/PartnerID`, `ebicsRequest/header/static/UserID`), der einen neuen Auftrag einreicht bzw. zu einem bereits vorhandenen Auftrag bankfachliche EUs liefert.
  - (bedingt) technischer Teilnehmer (`ebicsRequest/header/static/PartnerID`, `ebicsRequest/header/static/SystemID`)
    `SystemID` muss genau dann vorhanden sein, wenn es sich bei dem Kundensystem um ein Multi-Usersystem handelt. Der technische Teilnehmer ist für das Erstellen der EBICS-Requests (inklusive der Authentifkationssignaturen) zuständig, die zu Aufträgen gehören, die vom Teilnehmer eingereicht oder bankfachlich unterzeichnet werden.

- (optional) Information zum Kundenprodukt (`ebicsRequest/header/static/Product`)
- Auftragsart (ebicsRequest/header/static/OrderDetails/OrderType)
- (bedingt) Auftragsnummer (ebicsRequest/header/static/OrderDetails/OrderID) OrderID (Auftragsnummer) ist nur vorhanden, wenn eine Datei zur Bank übertragen wird, die sich auf einen Auftrag mit bereits existierender Auftragsnummer bezieht.
- Auftragsattribute (`ebicsRequest/header/static/OrderDetails/OrderAttribute`). Ist der Wert für `OrderAttribute` gleich „UZHNN“, dann werden innerhalb der vorliegenden Transaktion nur die bankfachlichen EUs zu einem Auftrag übermittelt und `ebicsRequest/header/static/NumSegments` muss den Wert „0“ haben. Der Wert „OZHNN“ bedeutet, dass zusätzlich zu den EUs auch Nutzdaten(segmente) übertragen werden: sofern diese EUs nicht ausreichend sind, wird die Datei an die VEU übergeben. Ist der Wert schließlich „DZHNN“, so wird der Auftrag über handschriftlich unterzeichnete Begleitzettel bankfachlich freigegeben: die Auftragsdaten werden mit einer Transportunterschrift signiert, nach deren erfolgreicher Prüfung der Auftrag direkt an die bankspezifische Folgeverarbeitung (und nicht an die VEU) weitergereicht wird
- Auftragsparameter (`ebicsRequest/header/static/OrderDetails/OrderParams`); die Ausprägung der Auftragsparameter ist abhängig von der Auftragsart (siehe auch Kapitel 3.11)
- Hashwerte der öffentlichen Schlüssel des Kreditinstituts, über die der Teilnehmer verfügt (ebicsRequest/header/static/BankPubKeyDigests/Authentication, ebicsRequest/header/static/BankPubKeyDigests/Encryption, ebicsRequest/header/static/BankPubKeyDigests/Signature).
  Zu jedem dieser Hashwerte wird sowohl der verwendete Hashalgorithmus als auch die Version des entsprechenden Authentifikations- bzw. Verschlüsselungs- bzw. Signaturverfahrens angegeben.
  Die SHA-256-Hashwerte der öffentlichen Schlüssel des Kreditinstitus für X002 und E002 werden über die Konkatenation des Exponenten mit einem Leerzeichen und dem Modulus in hexadezimaler Darstellung (mit Kleinbuchstaben) ohne führende Nullen (bezogen auf die hexadezimale Darstellung) gebildet. Dieser String muss basierend auf US-ASCII-Codierung in ein Byte-Array umgewandelt werden. In der Version „H004“ des EBICS-Protokolls ist die EU der Kreditinstitute nur vorgesehen (siehe dazu Kapitel 3.5.2). Das Element `BankPubKeyDigests/Signature` ist in Vorbereitung zukünftiger EBICS-Versionen bereits in dieser Beschreibung enthalten,

obwohl seine maximale Häufigkeit (maxOccurs) in der Version „H004“ gleich 0 ist

- Sicherheitsmedium für den bankfachlichen Schlüssel des Teilnehmers (`ebicsRequest/header/static/SecurityMedium`)
- Authentifikationssignatur des technischen Teilnehmers, falls ein solcher vorhanden ist, ansonsten die Authentifikationssignatur des einreichenden Teilnehmers selbst (`ebicsRequest/AuthSignature`)
  Die Authentifikationsignatur umfasst alle XML-Elemente des EBICS-Requests, deren Attributwert für `@authenticate` gleich „true“ ist. Die Definition des XML-Schemas „ebics_request_H004.xsd“ gewährleistet, dass der Wert des Attributs `@authenticate` für genau die Elemente gleich „true“ ist, die auch unterschrieben werden müssen
- (optional) Daten für die Vorabprüfung des Auftrags (`ebicsRequest/body/PreValidation`)
- Information zur Verschlüsselung der EUs und Auftragsdaten (`ebicsRequest/body/DataTransfer/DataEncryptionInfo`), die insbesondere auch den asymmetrisch verschlüsselten Transaktionsschlüssel (`ebicsRequest/body/DataTransfer/DataEncryptionInfo/TransactionKey`) enthält
- EUs der Nutzdaten des Auftrags (`ebicsRequest/body/DataTransfer/SignatureData`)
  SignatureData enthält ein Instanzdokument, das zu „ebics_orders_H004.xsd“ konform ist und `UserSignatureData` als Toplevel-Element enthält. Vor der Einbettung in den EBICS-Request wurde dieses Instanzdokument mit ZIP komprimiert, für das Kreditinstitut verschlüsselt und schließlich base64-kodiert (siehe Anhang (Kapitel 11.2.2)). Ein Beispiel eines solchen Instanzdokuments, welches eine einzige EU enthält, ist in Abbildung 43 enthalten. Die Belegung für das Attribut `PartnerID` im `UserSignatureData` Dokument muss identisch sein zur Kunden-ID des Einreichers im Element `ebicsRequest/header/static/PartnerID`.

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
xmlns="urn:org:ebics:H004"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:org:ebics:H004 ebics_request_H004.xsd"
Version="H004" Revision="1">
  <header authenticate="true">
    <static>
      <HostID>EBIXHOST</HostID>
      <Nonce>BDA2312973890654FAC9879A89794E65</Nonce>
      <Timestamp>2005-01-30T15:30:45.123Z</Timestamp>
      <PartnerID>PARTNER1</PartnerID>
      <UserID>USER0001</UserID>
```

```
      <Product Language="de" InstituteID="Institutskennung">Kundenproduktkennung</Product>
      <OrderDetails>
        <OrderType>IZV</OrderType>
        <OrderAttribute>OZHNN</OrderAttribute>
        <StandardOrderParams/>
      </OrderDetails>
      <BankPubKeyDigests>
        <Authentication Version="X002"
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256">1H/rQr2Axe9hYTV2n/tCp+3UIQQ=</Authenticati
on>
        <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> …hier Hashwert Authentifikation..</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue> …hier Signaturwert Authentifikation..</ds:SignatureValue>
  </AuthSignature>
  <body>
    <PreValidation authenticate="true">
      <DataDigest SignatureVersion="A004">bTTeUGiVqOUjVfJviMo97LHEDmQ=</DataDigest>
    </PreValidation>
    <DataTransfer>
      <DataEncryptionInfo authenticate="true">
        <EncryptionPubKeyDigest Version="E002"
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256">..hier Hashwert des öffentlichen
Bankschlüssels für die Verschlüsselung ..</EncryptionPubKeyDigest>
        <TransactionKey>EIGI4En6KEB6ArEzw+iq4N1wm6EptcyxXxStA…</TransactionKey>
       </DataEncryptionInfo>
      <SignatureData authenticate="true">n6KEB6ArEzw+iq4N1wm6EptcyxXxStAO…</SignatureData>
    </DataTransfer>
  </body>
</ebicsRequest>
```

Abbildung 42: EBICS-Request für die Transaktionsinitialisierung für die Auftragsart IZV

```
<?xml version="1.0" encoding="UTF-8"?>
<UserSignatureData
xmlns="http://www.ebics.org/S001"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ebics.org/S001 http://www.ebics.org/S001/ebics_signature.xsd">
<UserSignatureData>
    <OrderSignatureData>
       <SignatureVersion>A005</SignatureVersion>
```

```
        <SignatureValue>EUXkQa……</SignatureValue>
        <PartnerID>PARTNER1</PartnerID>
        <UserID>User1</UserID>
    </OrderSignatureData>
</UserSignatureData>
```

Abbildung 43: XML-Dokument, das die EUs der Unterzeichner des IZV-Auftrags enthält

- Übertragung der folgenden Daten in der EBICS-Response (siehe auch Beispiel in Abbildung 44)

bankfachlicher Returncode (`ebicsResponse/body/ReturnCode`)
Auftragsnummer (`ebicsResponse/header/mutable/OrderId`)
technischer Returncode (`ebicsResponse/header/mutable/ReturnCode`)
technischer Reporttext (`ebicsResponse/header/mutable/ReportText`)
(bedingt) banksystemweit eindeutige TransaktionsID (`ebicsResponse/header/static/TransactionID`), falls die folgenden Bedingungen erfüllt sind:

- Während der Transaktionsinitialisierung sind keine Fehler technischer oder bankfachlicher Art aufgetreten
- Innerhalb der aktuellen Transaktion werden in weiteren darauf folgenden Transaktionsschritten Nutzdatensegmente übertragen, d.h. die Auftrags-attribute sind „OZHNN“ oder „DZHNN“.

Transaktionsphase (`ebicsResponse/header/mutable/TransactionPhase`) mit der Belegung „Initialisation“
Authentifikationssignatur des Kreditinstituts (`ebicsResponse/AuthSignature`)
Die Authentifikationsignatur umfasst alle XML-Elemente der EBICS-Response, deren Attributwert für `@authenticate` gleich „true“ ist. Die Definition des XML-Schemas „ebics_response_H004.xsd“ gewährleistet, dass der Wert des Attributs `@authenticate` für genau die Elemente gleich „true“ ist, die auch unterschrieben werden müssen
(optional) Zeitstempel der letzten Aktualisierung der Bankparameter (`ebicsResponse/body/TimestampBankParameter`).

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsResponse
xmlns="urn:org:ebics:H004"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:org:ebics:H004 ebics_response_H004.xsd"
Version="H004" Revision="1">
  <header authenticate="true">
    <static>
      <TransactionID>ABCDEF41394644363445313243ABCDEF</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> …hier Hashwert Authentifikation..</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue> …hier Signaturwert Authentifikation..</ds:SignatureValue>
  </AuthSignature>
  <body>
    <ReturnCode authenticate="true">000000</ReturnCode>
  </body>
</ebicsResponse>
```

Abbildung 44: EBICS-Response der Transaktionsinitialisierung für die Auftragsart IZV

5.5.1.1.2 EBICS-Nachrichten der Phase Datentransfer eines Nutzdatensegments

- Übertragung der folgenden Daten im EBICS-Request (siehe Beispiel in Abbildung 45):

Host-ID des EBICS Bankrechners (`ebicsRequest/header/static/HostID`)

Daten zur Identifikation des aktuellen Transaktionsschritts:

- TransaktionsID (ebicsRequest/header/static/TransactionID)
- Transaktionsphase (`ebicsRequest/header/mutable/TransactionPhase`) mit der Belegung „Transfer“
- laufende Nummer des Nutzdatensegments (`ebicsRequest/header/mutable/SegmentNumber`)
  Das Attribut ebicsRequest/header/mutable/SegmentNumber@lastSegment gibt an, ob es sich dabei um das letzte Datensegment handelt.

Authentifikationssignatur des technischen Teilnehmers, falls ein solcher für die aktuelle Transaktion definiert wurde, ansonsten die Authentifikationssignatur des einreichenden Teilnehmers selbst (`ebicsRequest/AuthSignature`)
Die Authentifikationsignatur umfasst alle XML-Elemente des EBICS-Requests, deren Attributwert für `@authenticate` gleich „true“ ist. Die Definition des XML-Schemas „ebics_request_H004.xsd“ gewährleistet, dass der Wert des Attributs `@authenticate` für genau die Elemente gleich „true“ ist, die auch unterschrieben werden müssen

das eigentliche Nutzdatensegment (`ebicsRequest/body/DataTransfer/OrderData`) (siehe Kapitel 3.3 sowie Kapitel 7 für Einzelheiten der Segmentierung von Nutzdaten).

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
 xmlns="urn:org:ebics:H004"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_request_H004.xsd"
 Version="H004" 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> …hier Hashwert Authentifikation..</ds:DigestValue>
     </ds:Reference>
   </ds:SignedInfo>
   <ds:SignatureValue> …hier Signaturwert Authentifikation..</ds:SignatureValue>
 </AuthSignature>
 <body>
   <DataTransfer>
     <OrderData>RUJJQ1MtUmVxdWVzdCBm/HIgZGllINxiZXJ0…</OrderData>
   </DataTransfer>
 </body>
</ebicsRequest>
```

Abbildung 45: EBICS-Request für die Übertragung des letzten Nutzdatensegments für die Auftragsart IZV

- Übertragung der folgenden Daten in der EBICS-Response (siehe auch Beispiel in Abbildung 46)

bankfachlicher Returncode (`ebicsResponse/body/ReturnCode`)

Auftragsnummer (`ebicsResponse/header/mutable/OrderId`)

technischer Returncode (`ebicsResponse/header/mutable/ReturnCode`)

technischer Reporttext (`ebicsResponse/header/mutable/ReportText`)

Daten zur Identifikation eines Transaktionsschritts:
Falls der technische Returncode den Wert EBICS_TX_RECOVERY_SYNC hat, identifiziert dieser Transaktionsschritt den Wiederaufsetzpunkt der Upload-Transaktion. Sind jedoch wie

im vorliegenden Beispiel weder technische noch fachliche Fehler aufgetreten, so spiegelt dieser Transaktionsschritt den gerade aktuellen Transaktionsschritt wider

- TransaktionsID (`ebicsResponse/header/static/TransactionID`)
- Transaktionsphase (`ebicsResponse/header/mutable/TransactionPhase`)
- (bedingt) laufende Nummer des Nutzdatensegments (`ebicsResponse/header/mutable/SegmentNumber`), falls der Wert für `TransactionPhase` ungleich „Initialisation“ ist
  Das Attribut `ebicsResponse/header/mutable/SegmentNumber@lastSegment` gibt an, ob es sich dabei um das letzte Nutzdatensegment handelt.

Authentifikationssignatur des Kreditinstituts (`ebicsResponse/AuthSignature`)
Die Authentifikationsignatur umfasst alle XML-Elemente der EBICS-Response, deren Attributwert für `@authenticate` gleich „true“ ist. Die Definition des XML-Schemas „ebics_response_H004.xsd“ gewährleistet, dass der Wert des Attributs `@authenticate` für genau die Elemente gleich „true“ ist, die auch unterschrieben werden müssen.

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsResponse
 xmlns=" rn:org:ebics:H004"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_response_H004.xsd"
 Version="H004" Revision="1">
 <header authenticate="true">
   <static>
     <TransactionID>ABCDEF41394644363445313243ABCDEF</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> …hier Hashwert Authentifikation..</ds:DigestValue>
     </ds:Reference>
   </ds:SignedInfo>
   <ds:SignatureValue> …hier Signaturwert Authentifikation..</ds:SignatureValue>
 </AuthSignature>
 <body>
   <ReturnCode authenticate="true">000000</ReturnCode>
 </body>
```

```
</ebicsResponse>
```

Abbildung 46: EBICS-Response der Übertragung des letzten Nutzdatensegments für die Auftragsart IZV

5.5.1.2 Verarbeitung der EBICS-Nachrichten

Kapitel 5.5.1.1 beschreibt den Inhalt der EBICS-Nachrichten, die im Rahmen einer Upload-Transaktion ausgetauscht werden. Gegenstand dieses Kapitels ist die bankseitige Verarbeitung dieser EBICS-Nachrichten. Im Sequenzdiagramm in Abbildung 41 sind Aktionssequenzen hervorgehoben, deren Ablauf hier näher beschrieben wird.

Um die Beschreibung der Abläufe zu vereinfachen, wird angenommen, dass jeder Verarbeitungsschritt einen Returncode (RC) liefert, dessen Wert gleich 0 („000000“, EBICS_OK) ist, wenn dieser Schritt erfolgreich durchgeführt werden konnte. In Abhängigkeit von RC werden der technische Returncode (RCT) und der bankfachliche Returncode (RCF) gesetzt, deren Werte dann in die EBICS-Nachrichten einfließen.

Die Überprüfung der Validität der EBICS-Requests erfolgt auf Grundlage der XML-Schema Definitionsdatei „ebics_request_H004.xsd“ sowie unter Berücksichtigung der Einschränkungen, die in Kapitel 5.5.1.1 zu den einzelnen Requests festgelegt wurden. Normalerweise findet die Validitätsprüfung parallel und/oder verzahnt mit den anderen Ablaufschritten der Verarbeitung der EBICS-Requests statt. Die nachfolgenden Abläufe verzichten auf die Darstellung eines Ablaufschritts vom Typ „Validitätsprüfung des EBICS-Requests“ zugunsten einer möglichst einfachen Darstellung. Demnach können diese Abläufe durch folgende zusätzliche technische Fehler abgebrochen werden:

- EBICS_INVALID_XML
  Die empfangene EBICS XML-Nachricht entspricht syntaktisch nicht den Vergaben des EBICS XML-Schemas: das XML ist nicht wohlgeformt bzw. nicht valide gemäß Schema,
  z.B. Upload-Request ohne das Element `HostId` (laut Schema erforderlich).
- EBICS_INVALID_REQUEST
  Die empfangene EBICS XML-Nachricht entspricht syntaktisch nicht den EBICS Vorgaben,
  z.B. Upload-Request ohne Element `NumSegments` (laut Schema optional aber laut Kapitel 5.5.1.1.1 erforderlich).
- EBICS_INVALID_REQUEST_CONTENT
  Die empfangene EBICS XML-Nachricht entspricht semantisch nicht den EBICS Vorgaben, obwohl es gemäß Schema syntaktisch korrekt ist,
  z.B. Upload-Request mit `OrderAttribute` „UZHNN“ und `NumSegments` > 0
  (laut Schema valide aber laut Kapitel 5.5.1.1.1 unzulässig).

5.5.1.2.1 Verarbeitung in der Initialisierungsphase

In Abbildung 50 ist die bankseitige Verarbeitung des EBICS-Requests dargestellt, welcher in der Initialisierungsphase einer Upload-Transaktion vom Kundensystem an das Banksystem versendet wird. Die einzelnen Verarbeitungsschritte werden im Folgenden näher erläutert:

I. Erzeugung einer EBICS-Transaktion (siehe Abbildung 49)

Dieser Verarbeitungsschritt ist sowohl für Upload- als auch für Download-Transaktionen relevant. Die nachfolgende Beschreibung berücksichtigt beide Transaktionstypen, um dann in den nachfolgenden Kapiteln zum Thema Download-Transaktionen auf diese Beschreibung verweisen zu können.

I.a. Überprüfung der Auftragsart

Die Überprüfung der Autragsart liefert den technischen Returncode EBICS_INVALID_ORDER_TYPE im Falle einer ungültigen Auftragsart bzw. den technischen Returncode EBICS_UNSUPPORTED_ORDER_TYPE im Fall einer gültigen, jedoch optionalen Auftragsart, die vom Banksystem nicht unterstützt wird.

I.b. Replay Test

Der Replay-Test liefert den technischen Returncode EBICS_TX_MESSAGE_REPLAY, falls es sich bei dem EBICS-Request um einen wieder eingespielten Request handelt. Details zur Replay-Vermeidung sind im Anhang (Kapitel 11.4) beschrieben.

I.c. Prüfung der Authentizität des EBICS-Requests (siehe Abbildung 47):

Die Authentifikationssignatur wird von einem technischen Teilnehmer geleistet, wenn ein solcher Bestandteil der Steuerdaten ist. Andernfalls wird die Authentifikationssignatur vom (nichttechnischen) Teilnehmer der EBICS-Transaktion erzeugt, welcher den Auftrag einreicht bzw. bankfachliche EUs zu einem bereits bestehenden Auftrag nachliefert. Um die Authentifikationssignatur eines beliebigen (technischen oder nichttechnischen) Teilnehmers prüfen zu können, muss die entsprechende Kombination aus Kunden- und TeilnehmerID im Banksystem registriert sein und der Zustand des Teilnehmers auf „Bereit“ stehen. In Fehlersituationen, die aus einer ungültigen Kombination aus Kunden-/TeilnehmerID oder einem unzulässigen Teilnehmerzustand resultieren, erhält der Sender den technischen Returncode EBICS_AUTHENTICATION_FAILED.

Die Verifikation der Authentifikationssignatur beinhaltet:

- die Prüfung, ob mit der Authentifikationssignatur alle erforderlichen Elemente der EBICS-Nachricht unterschrieben wurden: Das sind alle XML-Elemente des EBICS-Requests, deren Attributwert für `@authenticate` gleich „true“ ist
- die Verifikation der Authentifikationsunterschrift selbst.

Dieser Verarbeitungsschritt bricht mit dem technischen Fehler EBICS_AUTHENTICATION_FAILED ab, wenn die Authentifikationssignatur nicht erfolgreich verifiziert werden kann.

Stammt die erfolgreich verifizierte Authentifikationssignatur von einem technischen Teilnehmer, so wird zusätzlich die Gültigkeit und der Zustand des (nichttechnischen) Teilnehmers überprüft. Fehler, die aus einer ungültigen Kombination aus Kunden-/TeilnehmerID oder einem unzulässigen Teilnehmerzustand resultieren, werden dem Sender des EBICS-Requests mit Hilfe der technischen Fehlercodes EBICS_USER_UNKNOWN bzw. EBICS_INVALID_USER_STATE mitgeteilt.

Begründung: Wenn die Authentifikationssignatur nicht erfolgreich verifiziert werden kann, stammt der EBICS-Request potenziell von einem Angreifer. Fehler wie beispielsweise „Unbekannter Teilnehmer“ oder „Unzulässiger Teilnehmerzustand“ werden in diesem Fall nicht direkt an den Sender des EBICS-Requests weitergereicht, um potenziellen Angreifern keine genaue Information über die Gültigkeit von TeilnehmerIDs oder den Zustand von Teilnehmern zu geben. Nachdem die Authentifikationssignatur des technischen Teilnehmers jedoch erfolgreich verifiziert werden konnte, werden die Fehler EBICS_USER_UNKNOWN bzw. EBICS_INVALID_USER_STATE, die sich auf den nichttechnischen Teilnehmer der EBICS-Transaktion beziehen, an den authentisierten Sender weitergegeben.

I.d. Überprüfung der Hashwerte der Bankschlüssel

Mit dieser Überprüfung soll verhindert werden, dass ein Teilnehmer Aufträge einreicht, obwohl er nicht über die aktuellen öffentlichen Schlüssel des Kreditinstituts verfügt. In der Version „H004“ von EBICS ist die EU der Kreditinstitute nur vorgesehen (siehe Kapitel 3.5.2). Deshalb werden in der Version „H004“ nur die Hashwerte des öffentlichen Authentifikationsschlüssels und des öffentlichen Verschlüsselungsschlüssels überprüft. Nachfolgende EBICS-Versionen, die die EU der Kreditinstitute unterstützen, müssen in diesem Verarbeitungsschritt auch den Hashwert des öffentlichen bankfachlichen Schlüssels des Kreditinstituts überprüfen. Zu diesem Zweck übermittelt der Teilnehmer die Hashwerte der öffentlichen Schlüssel des Kreditinstituts, über die er verfügt. Das Banksystem überprüft diese Hashwerte. Sollten sie nicht mit den Hashwerten der aktuellen öffentlichen Schlüssel übereinstimmen, wird die Transaktionsinitialisierung mit dem technischen Returncode EBICS_BANK_PUBKEY_UPDATE_REQUIRED abgebrochen.

Verfügt der Teilnehmer nicht über den aktuellen Authentifikationsschlüssel des Kreditinstituts, so kann er die Authentifikationssignatur der EBICS-Response des Kreditinstituts nicht erfolgreich verifizieren. Dennoch sollte beim Eintreffen des Fehlers EBICS_BANK_PUBKEY_UPDATE_REQUIRED die Aktualität der Bankschlüssel überprüft werden und notfalls die neuesten Schlüssel mit Hilfe der systembedingten Auftragsart HPB abgeholt werden.

I.e. Teilnehmerbezogene Prüfungen des Auftrags (siehe Abbildung 48)

I.e.a. Überprüfung der Auftragsartberechtigung

Hier wird überprüft, ob der Teilnehmer das Recht hat, Aufträge der gegebenen Auftragsart einzureichen. Schlägt diese Überprüfung fehl, dann wird die Transaktionsinitialisierung mit dem fachlichen Fehler EBICS_AUTHORISATION_FAILED abgebrochen.

Bei Sendeaufträgen ist die Auftragsartberechtigung erfolgreich, wenn der Teilnehmer für die gegebene Auftragsart zumindest eine EU-Berechtigung der Klasse „T“ besitzt.

Hinweis: Hier wird nicht die EU-Berechtigung der eigentlichen Unterzeichner des Auftrags überprüft. Diese Prüfung ist Teil der (optionalen) Vorabprüfung eines Auftrags.

Bei Downloadaufträgen ist die Auftragsartberechtigung nicht an eine EU-Berechtigung gekoppelt. Hier wird überprüft, ob der Teilnehmer eine Berechtigung für die gegebene Auftragsart besitzt.

I.e.b. Bankfachliche Vorabprüfung

Diese Prüfung betrifft nur Sendeaufträge, Details der Vorabprüfung sind in Kapitel 5.3 beschrieben.

Wenn das Banksystem die optionale Vorabprüfung nicht unterstützt, das Kundensystem jedoch Daten zur Vorabprüfung liefert, wird der Hinweis EBICS_NO_ONLINE_CHECKS an das Kundensystem zurückgegeben. Diese fachliche Information hat keinen Einfluss auf die laufende Transaktion, der Auftrag wird fortgesetzt.

Die bankfachliche Vorabprüfung eines Sendeauftrags liefert im Fehlerfall die folgenden fachlichen Returncodes:

- EBICS_SIGNATURE_VERIFICATION_FAILED
  Dieser fachliche Fehler tritt dann auf, wenn die EU eines Unterzeichners des Auftrags nicht erfolgreich verifiziert werden konnte

- EBICS_INVALID_SIGNATURE_FILE_FORMAT
  Die übergebenen EU-Dateien entsprechen nicht dem festgelegten Format

- EBICS_ PARTNER_ID_MISMATCH
  Die PartnerID (=KundenID) der EU-Datei stimmt nicht mit der PartnerID (=KundenID) des Einreichers überein

- EBICS_ACCOUNT_AUTHORISATION_FAILED
  Dieser fachliche Fehlercode wird zurückgegeben, wenn für einen der Unterzeichner die Kontoberechtigungsprüfung fehlschlägt

- EBICS_AMOUNT_CHECK_FAILED
  Dieser fachliche Fehler tritt auf, wenn für einen der Unterzeichner die Limitprüfung fehlschlägt

- EBICS_SIGNER_UNKNOWN
  Dieser fachliche Fehler tritt auf, wenn einer der Unterzeichner kein gültiger Teilnehmer ist

- EBICS_INVALID_SIGNER_STATE
  Dieser fachliche Fehler tritt auf, wenn der Zustand eines Unterzeichners ungleich „Bereit“ ist.

- Für Return Codes, die Zertifikate betreffen, wird auf Anhang 1 verwiesen.

I.e.c. Überprüfung von Auftragsattributen und Auftragsnummer

Die folgenden Überprüfungen betreffen nur die bankfachlichen Sendeaufträge:
Ein Auftrag mit Auftragsattributen "DZHNN" ist nur ohne Auftragsnummer zulässig.
Ein Auftrag mit Auftragsattributen "OZHNN" ist nur ohne Auftragsnummer zulässig und muss vor einer (möglichen) korrespondierenden Unterschriftsdatei mit Auftragsattribute “UZHNN” übertragen werden.
Ein Upload-Request mit Auftragsattributen “UZHNN” ist nur mit Auftragsnummer zulässig, wenn die Auftragsart “SPR” ist.
Ein Upload-Request mit der Auftragsart “SPR” hat immer die Auftragsattribute "UZHNN" und ist nur ohne Auftragsnummer zulässig.
Folglich sind mögliche Fehlermeldungen für Upload-Requests:

- Bei Upload-Requests mit Auftragsattributen “DZHNN” oder “OZHNN”, die mit Auftragsnummer übertragen werden, wird die Fehlermeldung EBICS_INCOMPATIBLE_ORDER_ATTRIBUTE zurückgegeben

- Bei einem Upload-Request mit Auftragsattributen “UZHNN”, der ohne Auftragsnummer übertragen wird, wird die Fehlermeldung EBICS_INCOMPATIBLE_ORDER_ATTRIBUTE zurückgegeben

- Bei einem Upload-Request mit Auftragsattributen “UZHNN”, der mit einer Auftragsnummer übertragen wird, die bereits vom Bankrechner an einen Auftrag mit Auftragsattributen “DZHNN” vergeben wurde, wird die Fehlermeldung EBICS_INCOMPATIBLE_ORDER_ATTRIBUTE zurückgegeben

- Bei einem SPR Upload-Request , der mit einer Auftragsnummer übertragen wird, wird die Fehlermeldung EBICS_INVALID_REQUEST_CONTENT zurückgegeben

- Bei einem Upload-Request mit Auftragsattributen “UZHNN”, der mit einer dem Bankrechner unbekannten Auftragsnummer übertragen wird, wird die Fehlermeldung EBICS_ORDER_ID_UNKNOWN zurückgegeben

- Bei einem Upload-Request mit Auftragsattributen “UZHNN”, der mit einer Auftragsnummer übertragen wird, die bereits vom Bankrechner für einen passenden Auftrag mit Auftragsattributen “OZHNN” vergeben wurde, dieser jedoch einen unzulässigen Verarbeitungsstatus hat (weil der Auftrag voll autorisiert oder zurückgewiesen wurde) wird die Fehlermeldung EBICS_ORDER_ID_ALREADY_EXISTS zurückgegeben.
  Dies gilt sowohl für Aufträge mit einer Verwaltung innerhalb als auch außerhalb der VEU

Regel für die Upload-Response: Jede Upload-Response enthält eine TransactionID und die durch den Bankrechner vergebene Auftragsnummer (=OrderId, dies gilt auch für Fehlerfälle)

Die folgenden Überprüfungen betreffen nur die bankfachlichen Abholaufträge.

Setzt der Teilnehmer die Auftragsattribute gleich „DZHNN“, so fordert er die Abholdaten ohne die EU des Kreditinstituts an. Die Transaktionsinitialisierung wird mit dem fachlichen Fehler EBICS_DOWNLOAD_SIGNED_ONLY abgebrochen, wenn für den Teilnehmer vereinbart wurde, dass er die Abholdaten für die gegebene Auftragsart nur mit der EU des Kreditinstituts abrufen darf.

Setzt der Teilnehmer die Auftragsattribute gleich „OZHNN“, so fordert er die Abholdaten mit der EU des Kreditinstituts an. Die Transaktionsinitialisierung wird mit dem fachlichen Returncode EBICS_DOWNLOAD_UNSIGNED_ONLY abgebrochen, wenn für den Teilnehmer vereinbart wurde, dass er die Abholdaten der gegebenen Auftragsart nur ohne EU des Kreditinstituts abrufen darf.
In der Version „H004“ des EBICS-Protokolls ist die EU der Kreditinstitute nur vorgesehen (siehe dazu Kapitel 3.5.2). Deshalb sind in der Version „H004“ für Abholaufträge nur die Auftragsattribute „DZHNN“ zulässig. Die Belegung “OZHNN” wird hier lediglich in Vorbereitung zukünftiger EBICS-Versionen berücksichtigt.

I.f. Erzeugen einer neuen EBICS-Transaktion mit eindeutiger TransaktionsID

Wenn alle vorhergehenden Prüfungen erfolgreich durchgeführt wurden und noch weitere Transaktionsschritte folgen, erzeugt die Transaktionsverwaltung für EBICS-Transaktionen auf Bankseite eine neue EBICS-Transaktion mit einer banksystemweit eindeutigen TransaktionsID. Details zur Generierung der TransaktionsID sind im Anhang (Kapitel 11.6) enthalten.

II. Vorverarbeitung

Die Vorverarbeitung betrifft hier die übermittelten EUs sowie die Auftragsparameter von Aufträgen mit Auftragsattributen „DZHNN“ oder „OZHNN“. Die Vorverarbeitung ist nicht Bestandteil der EBICS Spezifikation und somit abhängig von der Implementierung des Banksystems. Beispielsweise ist die Zwischenspeicherung der EUs und Auftragsparameter ein Teil dieser Vorverarbeitung.

III. Weiterleitung an die Verwaltung offener Aufträge

Sind die Auftragsattribute gleich “UZHNN”, werden also nur EUs zu einem gegebenen Auftrag übertragen, so besteht die EBICS-Transaktion aus einem einzigen Request-/Response-Paar. In diesem Fall werden die übermittelten Auftragsparameter und EUs direkt an die Verwaltung offener Aufträge weitergeleitet, und die Transaktion wird beendet. Die Komponente Verwaltung offener Aufträge ist nicht Bestandteil des EBICS-Standards.

IV. Erzeugen der EBICS-Response

Dieser Verarbeitungsschritt erzeugt die EBICS-Response, die anschließend an das Kundensystem geschickt wird. Im Fehlerfall enthält diese EBICS-Nachricht den ent-

sprechenden technischen oder fachlichen Fehlercode vorausgegangener Verarbeitungsschritte. Der Inhalt dieser EBICS-Nachricht ist in Kapitel 5.5.1.1.1 genauer beschrieben.

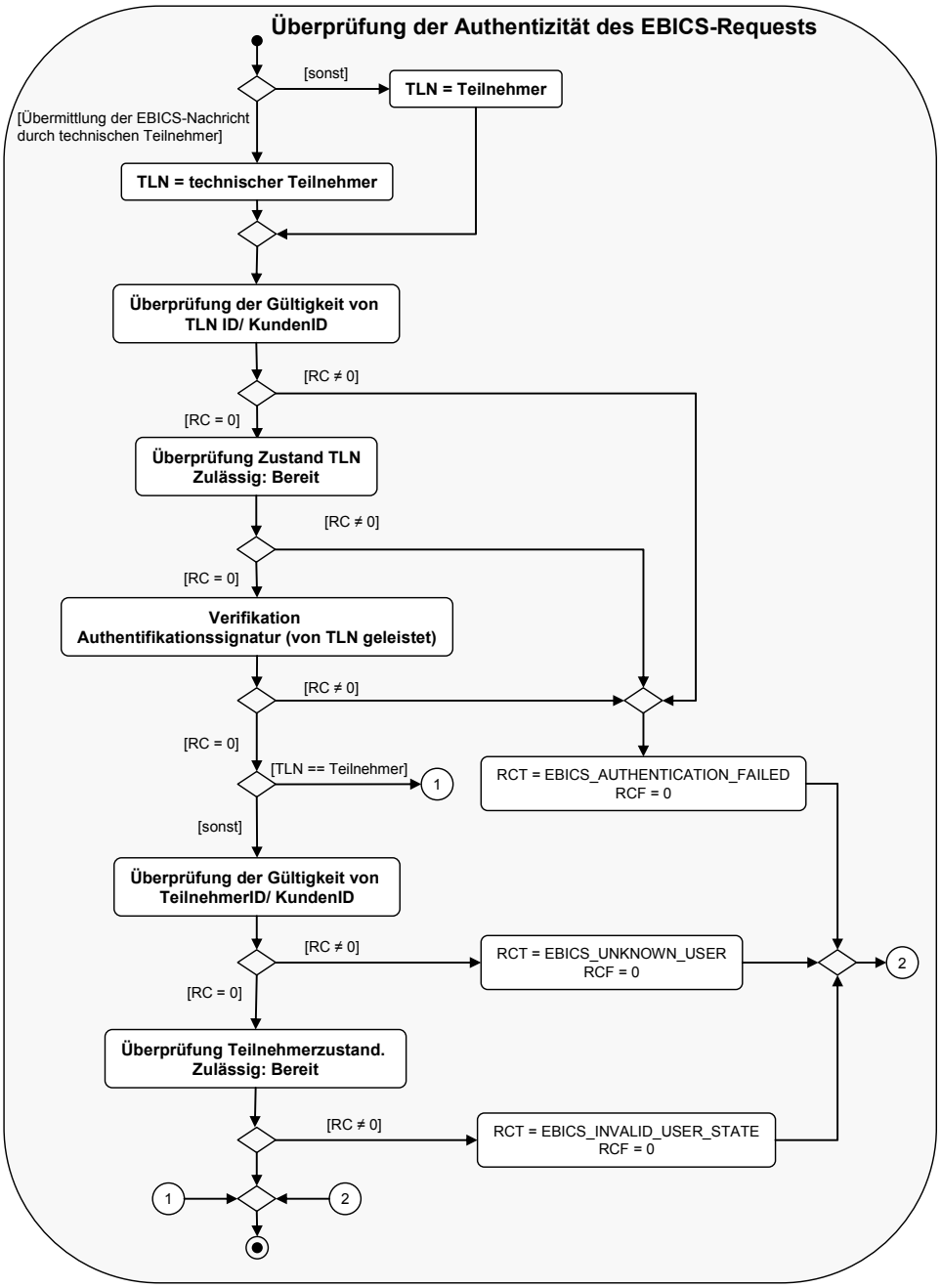


Abbildung 47: Detaillierte Beschreibung des Ablaufschritts „Überprüfung der Authentizität des EBICS-Requests“

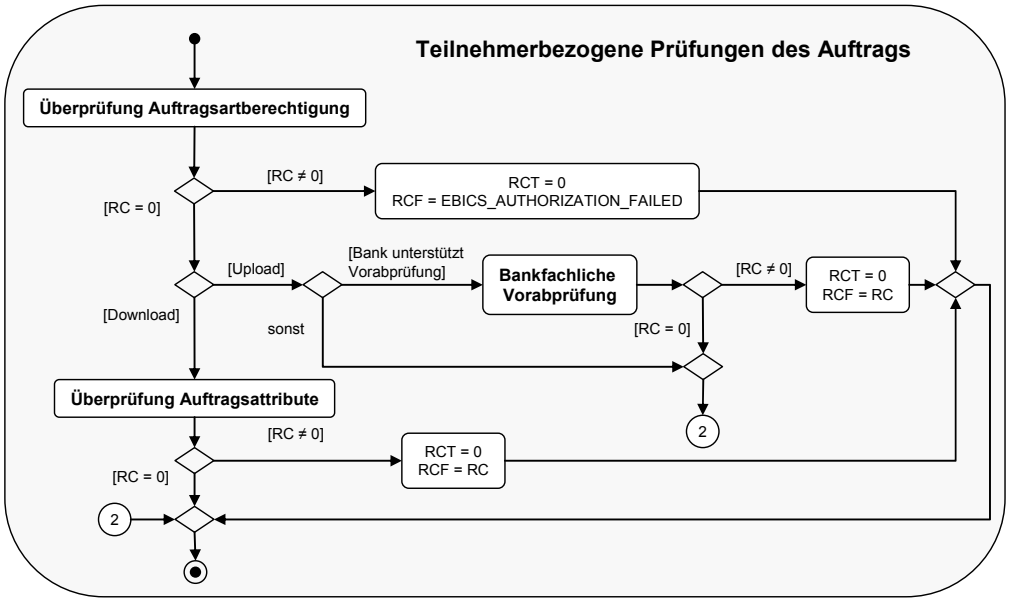


Abbildung 48: Detaillierte Beschreibung des Ablaufschritts „Teilnehmerbezogene Prüfungen des Auftrags“

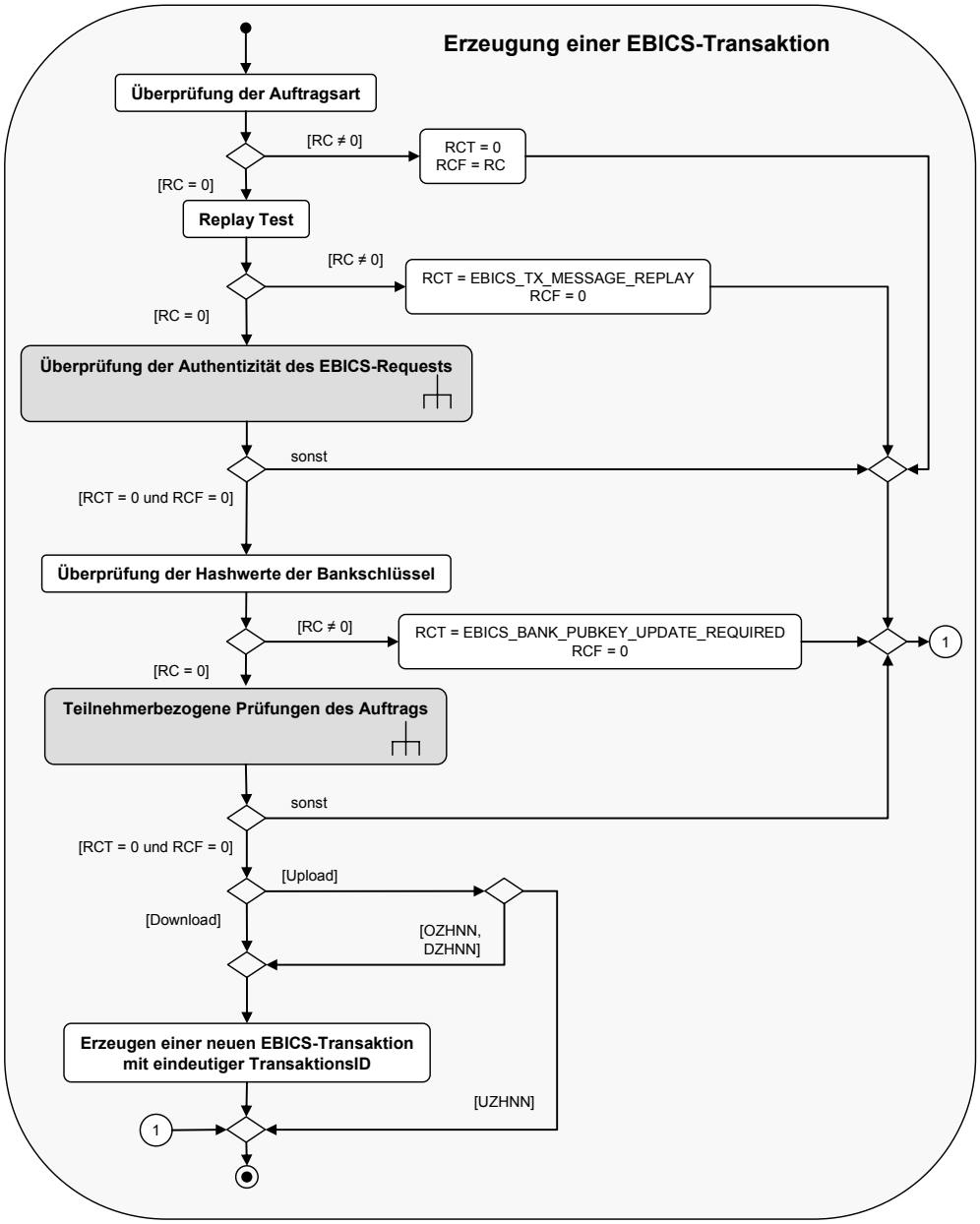


Abbildung 49: Detaillierte Beschreibung des Ablaufschritts „Erzeugung einer EBICS-Transaktion“

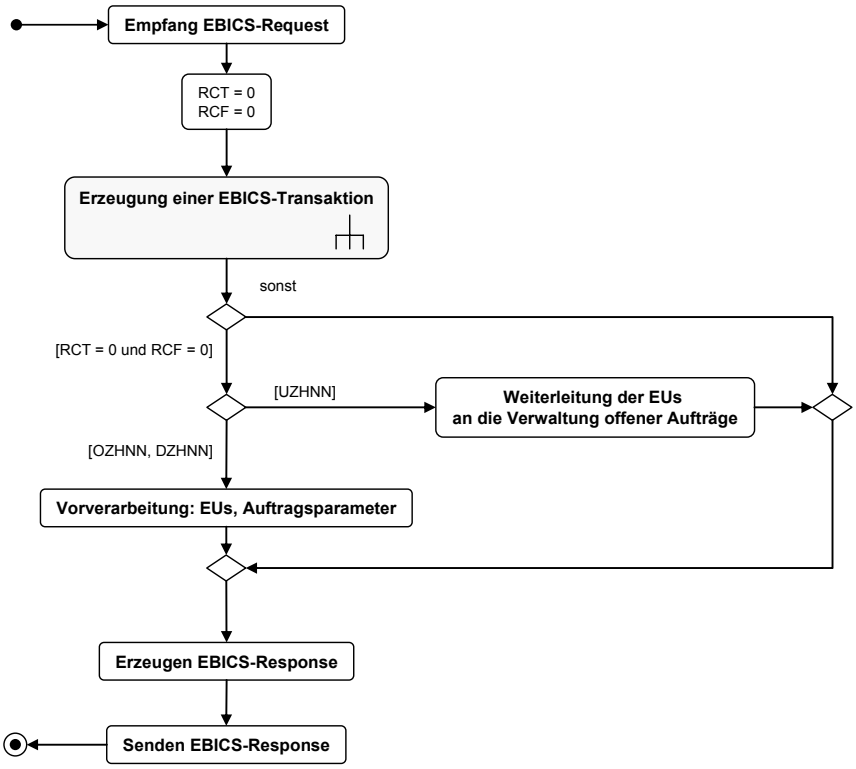


Abbildung 50: Verarbeitung des EBICS-Requests aus der Transaktionsinitialisierung

5.5.1.2.2 Verarbeitung in der Phase des Datentransfers

Die bankseitige Verarbeitung des EBICS-Requests, welcher in der Phase des Datentransfers einer EBICS-Transaktion vom Kundensystem an das Banksystem versendet wird, ist in Abbildung 52 sowie Abbildung 53 dargestellt. Die einzelnen Verarbeitungsschritte werden im Folgenden näher erläutert:

I. Überprüfung der EBICS-Transaktion (siehe Abbildung 51)

Dieser Verarbeitungsschritt ist sowohl für Upload- wie auch Download-Transaktionen relevant. Die nachfolgende Beschreibung berücksichtigt beide Transaktionstypen, um dann in den nachfolgenden Kapiteln zum Thema Download Transaktionen auf diese Beschreibung verweisen zu können.

I.a. Überprüfung der TransaktionsID

Hier wird überprüft, ob die EBICS-Transaktion mit der entsprechenden ID in der EBICS-Transaktionsverwaltung des Banksystems als offene, noch nicht abgeschlossene Transaktion existiert. Ist das nicht der Fall, so wird der technische

Fehlercode EBICS_TX_UNKNOWN_TXID an den Sender des EBICS-Request geliefert.

I.b. Überprüfung der Authentizität des EBICS-Request (siehe Abbildung 47)

Diese Prüfung der Authentizität des EBICS-Requests erfolgt genau wie in der Initialisierungsphase der Transaktion (siehe Kapitel 5.5.1.2.1, Punkt I.c) – mit dem Unterschied, dass die benötigten Daten (z.B. Kunden-/TeilnehmerID) nicht Teil der Headerdaten des EBICS-Requests sind, sondern in der Transaktionsverwaltung des Kreditinstituts zu der gegebenen TransaktionsID gespeichert werden. Kann die Prüfung nicht erfolgreich durchgeführt werden, so enthält die EBICS-Response einen entsprechenden Fehlercode gemäß dem Ablauf aus Abbildung 47. Es handelt sich dabei um einen die Fehler EBICS_AUTHENTICATION_FAILED, EBICS _USER_UNKNOWN oder EBICS_INVALID_USER_STATE. Nicht authentisierte Requests haben keine Auswirkung auf den Zustand der Transaktion in der Transaktionsverwaltung des Banksystems. Daten, die den Zustand einer Transaktion ausmachen, wie etwa der der nächste erwartete Transaktionsschritt oder der aktuelle Recovery-Zähler, werden nicht verändert. Dadurch wird verhindert, dass Angreifer mit Hilfe nicht authentisierter EBICS-Requests Einfluss auf eine Transaktion nehmen können. Die Transaktion kann durch den Teilnehmer fortgesetzt werden, als ob der EBICS-Request mit der ungültigen Authentifikationssignatur nicht empfangen worden wäre.

I.c. Überprüfung TxPhase/ TxSchritt aus dem EBICS-Request

An dieser Stelle wird überprüft, ob der Transaktionsschritt aus dem EBICS-Request zum aktuellen Zustand der EBICS-Transaktion auf Banksystemseite passt, wenn man eine bestimmte sequenzielle Abfolge der Transaktionsschritte voraussetzt.

Im Falle einer Upload-Transaktion wird von der sequenziellen Abfolge gemäß Abbildung 41 ausgegangen. Die Überprüfung der Transaktionsphase/ des Transaktionsschritts ist genau dann erfolgreich, wenn:

- der zuletzt vom Teilnehmer initiierte Transaktionsschritt erfolgreich abgeschlossen wurde, d.h. die Initialisierung bzw. die Übermittlung des n-ten Datensegments erfolgreich war
- der Transaktionsschritt aus dem EBICS-Request der nächste Transaktionsschritt in der sequenziellen Abfolge der Transaktionsschritte ist, d.h. es sich um die Übermittlung des 1-ten bzw. (n+1)-ten Datensegments handelt.

Die normale sequenzielle Abfolge von Transaktionsschritten einer Download-Transaktion ist in Abbildung 57 dargestellt. Die Überprüfung der Transaktionsphase/ des Transaktionsschritts ist genau dann erfolgreich, wenn die beiden folgenden Bedingungen erfüllt sind:

- Der zuletzt vom Teilnehmer initiierte Transaktionsschritt wurde erfolgreich durchgeführt, d.h. die Initialisierung (und somit Übermittlung des ersten Datensegments)

bzw. die Anforderung des n-ten Datensegments im Rahmen des Datentransfers waren erfolgreich

- Der Transaktionsschritt aus dem EBICS-Request ist der nächste Transaktionsschritt in der sequenziellen Abfolge der Transaktionsschritte, d.h. es handelt sich um die Anforderung des (n+1)-ten Datensegments oder um die Quittierung der abgeholten Daten, wenn n das letzte Datensegment darstellt.

II. Auswertung des Ergebnisses der Überprüfung der EBICS-Transaktion

Ist die Überprüfung des Transaktionsschritts nicht erfolgreich verlaufen, dann:

- wird überprüft, ob die Upload-Transaktion wiederaufgesetzt werden kann, falls das Banksystem Recovery von Transaktionen unterstützt. Diese Prüfung erfolgt gemäß der Beschreibung aus Kapitel 5.5.2.Kann die Transaktion wiederaufgesetzt werden, so wird der technische Fehlercode EBICS_TX_RECOVERY_SYNC zurückgeliefert, andernfalls wird die Transaktion mit dem technischen Fehlercode EBICS_TX_ABORT abgebrochen
- wird die Upload-Transaktion mit dem fachlichen Fehlercode EBICS_RECOVERY_NOT_SUPPORTED abgebrochen, falls das Banksystem Recovery von Transaktionen nicht unterstützt. Setzt man MAX gleich 0, so wird im Ablaufdiagramm auch der Fall berücksichtigt, in dem Recovery nicht unterstützt wird.

III. Überprüfung der Segmentnummer und Segmentgröße

Die laufende Nummer des übermittelten Nutzdatensegments (`ebicsRequest/header/mutable/SegmentNumber`) muss kleiner oder gleich der Gesamtanzahl zu übermittelnder Datensegmente sein. Falls die Nummer des übermittelten Nutzdatensegments mit der Gesamtanzahl übereinstimmt, muss zudem der Wert des Attributs `ebicsRequest/header/mutable/SegmentNumber@lastSegment` gleich „true“ sein. Ist eine der beiden Bedingungen nicht erfüllt, so wird die Transaktion mit dem technischen Fehlercode EBICS_TX_SEGMENT_NUMBER_EXCEEDED abgebrochen.

Ist die laufende Nummer des übermittelten Nutzdatensegments kleiner als die Gesamtanzahl zu übermittelnder Nutzdatensegmente und der Wert des Attributs `ebicsRequest/header/mutable/SegmentNumber@lastSegment` dennoch „true“, dann wird der technische Returncode der Fehlerklasse „Hinweis“ EBICS_TX_SEGMENT_NUMBER_UNDERRUN zurückgeliefert.

Die Größe des übermittelten Nutzdatensegments darf die in EBICS „H004“ fest vorgegebene Segmentgröße von 1 MB nicht überschreiten. Sonst wird die Transaktion mit dem technischen Fehlercode EBICS_SEGMENT_SIZE_EXCEEDED abgebrochen.

IV. Vorverarbeitung

Die Vorverarbeitung betrifft hier das übermittelte Nutzdatensegment. Die Vorverarbeitung von Nutzdatensegmenten ist nicht Bestandeil des EBICS Spezifikation. Sie ist abhängig

von der Implementierung des Banksystems, die Zwischenspeicherung des Nutzdatensegments kann ein Teil der Vorverarbeitung sein.

V. Weiterleitung an die Verwaltung offener Aufträge

Wenn das übermittelte Nutzdatensegment das letzte war, und es sich um einen bankfachlichen Sendeauftrag handelt, dann werden sämtliche im Rahmen der EBICS-Transaktion übermittelten Auftragsparameter, EUs sowie Auftragsdaten an die Verwaltung offener Aufträge weitergeleitet. Die EBICS-Transaktion kann danach beendet werden. Die Komponente Verwaltung offener Aufträge ist nicht Bestandteil des EBICS-Standards.

VI. Überprüfung und Durchführung des Auftrags

Wenn das übermittelte Nutzdatensegment das letzte war, und es sich um einen systembedingten Sendeauftrag handelt, dann wird auf der Basis der übermittelten Nutzdaten der Auftrag synchron geprüft und durchgeführt. Die gelieferten technischen oder fachlichen Fehlercodes sind abhängig von der Auftragsart und sind in den Kapiteln definiert, die diese Auftragsarten beschreiben.

VII. Erzeugen der EBICS-Response

Dieser Verarbeitungsschritt erzeugt die EBICS-Response, die anschließend an das Kundensystem geschickt wird. Im Fehlerfall enthält diese EBICS-Nachricht den technischen/ fachlichen Fehlercode vorausgegangener Verarbeitungsschritte. Der Inhalt dieser EBICS-Nachricht ist in Kapitel 5.5.1.1.2 genauer beschrieben.

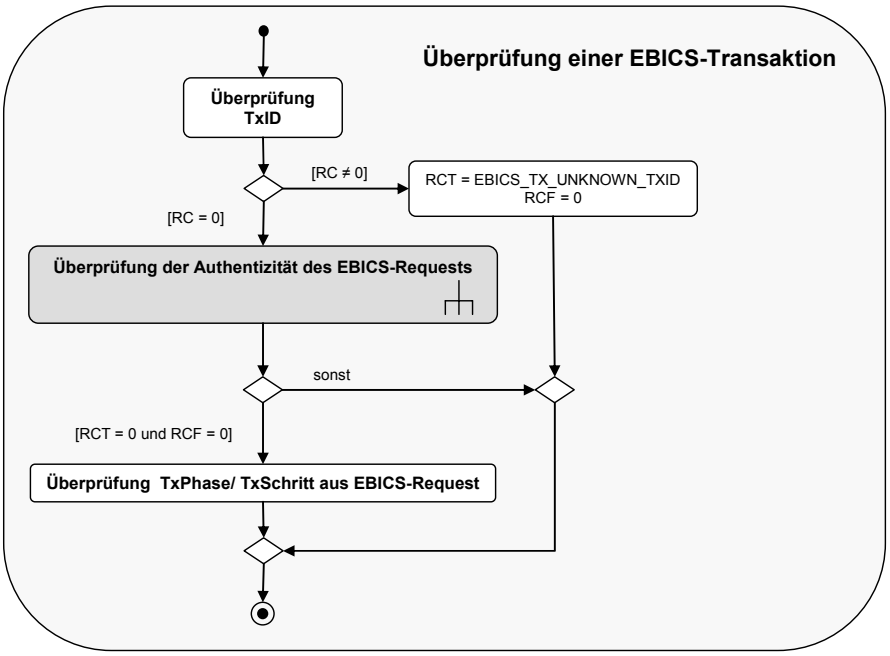


Abbildung 51: Detaillierte Beschreibung des Ablaufschritts „Überprüfung einer EBICS-Transaktion“

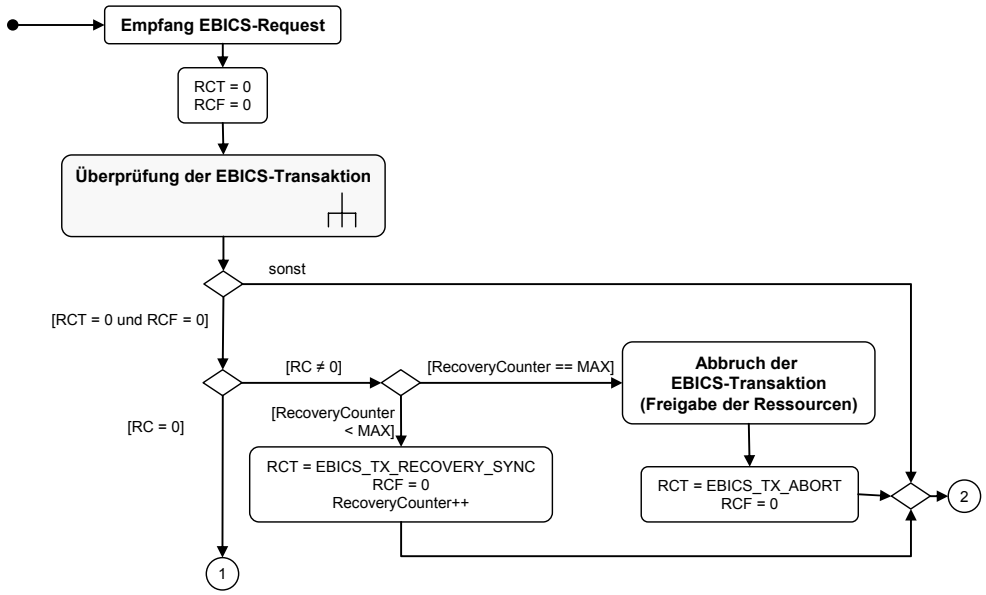


Abbildung 52: Verarbeitung eines EBICS-Requests zur Übermittlung eines Nutzdatensegments (1.Teil)

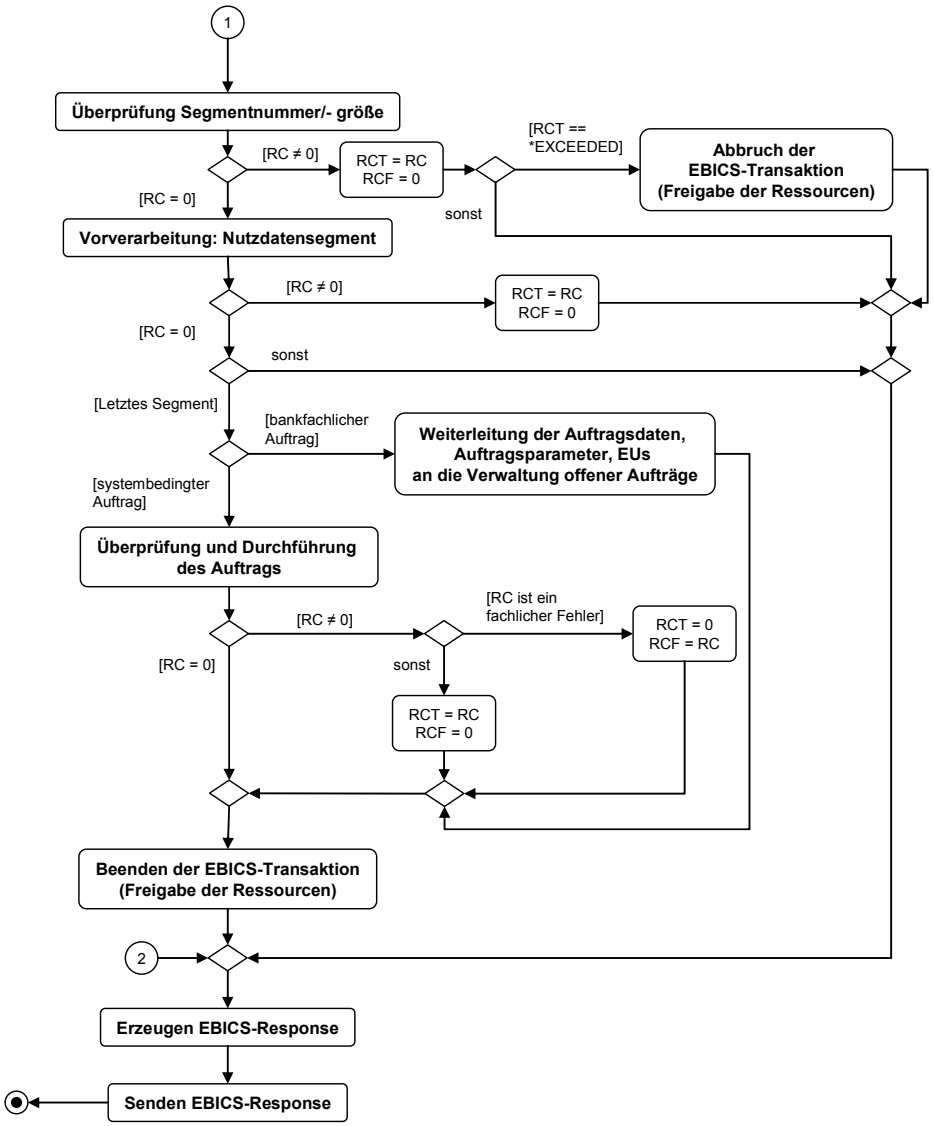


Abbildung 53: Verarbeitung eines EBICS-Requests zur Übermittlung eines Nutzdatensegments (2.Teil)

5.5.2 Wiederaufsetzen von Upload-Transaktionen

Das Kundensystem kann den Recovery-Mechanismus einsetzen, wenn eine der folgenden Fehlersituationen eintritt:

- Transportfehler bei der Übertragung eines EBICS-Requests in der Datentransfer-Phase der Transaktion

- Timeout bzw. Transportfehler beim Empfang einer EBICS-Response in der Datentransfer-Phase der Transaktion
- Verlust des Zustands der Transaktion auf Teilnehmerseite.

Auf Bankseite kann die fehlerhafte Verarbeitung eines EBICS-Requests im Rahmen der Datentransferphase, beispielsweise verursacht durch Fehler in der Vorverarbeitung eines übermittelten Nutzdatensegments, die erneute Übertragung dieses Requests erfordern. Es handelt sich hier um einen Sonderfall von Recovery, da das Kundensystem die Notwendigkeit, die Übertragung zu wiederholen, nicht ohne weiteres erkennt. Mit dem Recovery-Mechanismus von EBICS kann man diesen Sonderfall abhandeln.

EBICS verwendet einen optimistischen Ansatz beim Wiederaufsetzen einer Upload-Transaktion und verzichtet auf einen separaten Synchronisationsschritt mit dem Banksystem. Tritt eine der obigen Fehlersituationen ein, geht das Kundensystem zunächst davon aus, den Wiederaufsetzpunkt der Transaktion aufgrund der im Kundensystem (eventuell persistent) gespeicherten Transaktionsdaten zu kennen.

Wenn das Kundensystem annimmt, der Wiederaufsetzpunkt sei die Übermittlung des n-ten Nutzdatensegments, dann ist der nächste initiierte Transaktionsschritt die Übermittlung des (n+1)-ten Nutzdatensegments. EBICS-Requests im Rahmen des Recovery von Upload-Transaktionen unterscheiden sich nicht von den EBICS-Requests eines normalen, fehlerfreien Ablaufs einer Upload-Transaktion.

In Abbildung 54 ist beispielhaft der Ablauf einer Transaktion dargestellt, die mehrfach ein Wiederaufsetzen erfordert. Das Wiederaufsetzen geschieht jeweils ohne explizite Synchronisation zwischen Kunden- und Banksystem. Das 2. Nutzdatensegment wird dreimal übertragen, da das Kundensystem die entsprechende EBICS-Response wegen eines Timeout- oder eines Transportfehlers nicht empfangen konnte. Bei der zweiten und dritten Übertragung des 2. Nutzdatensegments geht das Kundensystem davon aus, dass der Wiederaufsetzpunkt die Übertragung des 1. Nutzdatensegments ist. Der Wert des RecoveryCounters ist nach der dritten und erfolgreichen Übertragung des 2. Nutzdatensegments gleich 2, da die letzten beiden Übertragungen des 2. Nutzdatensegments vom Banksystem als Recovery-Versuch gewertet werden. Die Transaktion scheitert schließlich an der zu hohen Zahl von Wiederaufsetzversuchen.

Ist die Annahme bzgl. des Wiederaufsetzpunktes falsch, so enthält die EBICS-Response der Übermittlung des (n+1)-ten Datensegments neben dem technischen Returncode EBICS_TX_RECOVERY_SYNC den eigentlichen Wiederaufsetzpunkt der Transaktion. Ist dieser Wiederaufsetzpunkt beispielsweise die Übermittlung des Nutzdatensegments k, so kann die Transaktion nach dieser Synchronisation einfach mit der Übermittlung der Segmente k+1, k+2, usw. fortgesetzt werden.

In Abbildung 55 ist der erfolgreiche Ablauf einer Transaktion dargestellt, der ein Wiederaufsetzen der Transaktion nach einer expliziten Synchronisation zwischen Kunden- und Bank-

system enthält. Hier überträgt das Kundensystem das Nutzdatensegment 1 in einem Zustand, in dem das Banksystem eigentlich das Segment 3 erwartet. Die EBICS-Response des Kreditinstituts (siehe Abbildung 56) enthält somit den Wiederaufsetzpunkt der Transaktion und der ist in diesem Fall die Übertragung des 2. Nutzdatensegments. Danach fährt das Kundensystem mit der Übertragung des Nutzdatensegments 3 fort und beendet die Transaktion mit der Übertragung des letzten Segments 4.

Formatiert: Deutsch (Deutschland)

Unabhängig davon, ob ein Kundensystem Fehler im Ablauf einer Transaktion feststellt, kann das Banksystem die erneute Übertragung eines EBICS-Request erzwingen. Das wird dadurch erreicht, dass die zugehörige EBICS-Response analog zu den obigen Recovery-Situationen neben dem Wiederaufsetzpunkt der Transaktion den technischen Returncode EBICS_TX_RECOVERY_SYNC enthält.

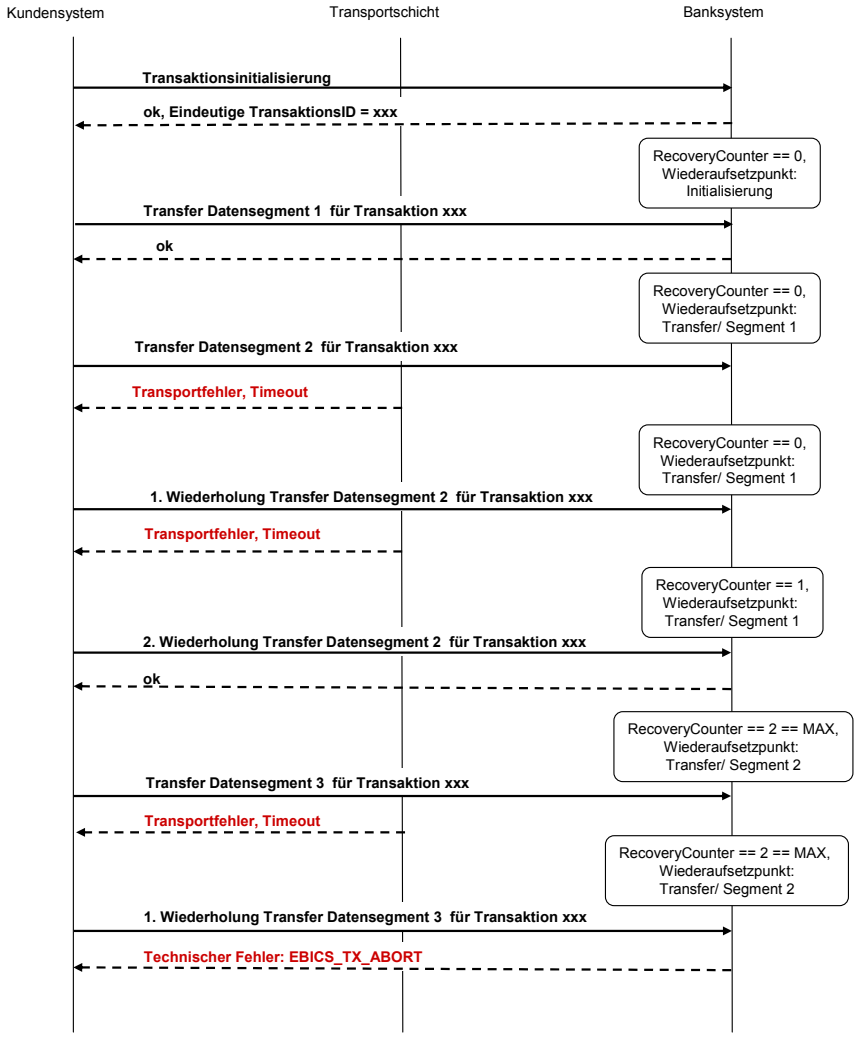


Abbildung 54: Abbruch des Recovery einer Upload-Transaktion aufgrund der Überschreitung der maximal erlaubten Zahl von Recovery-Versuchen

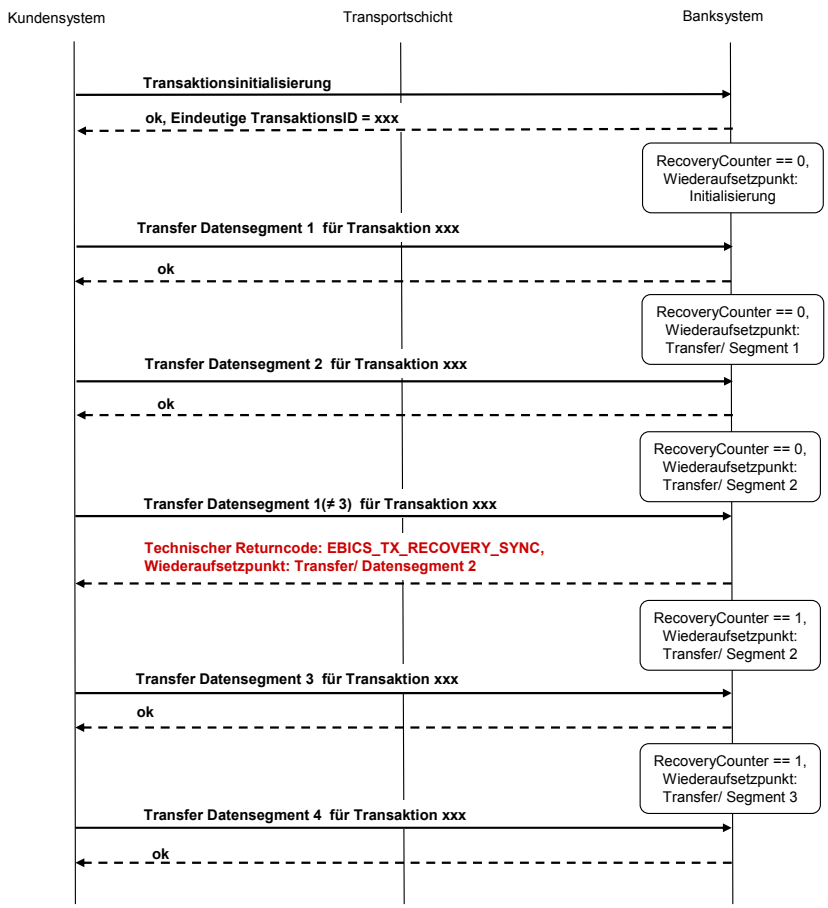


Abbildung 55: Recovery einer Upload-Transaktion mit expliziter Synchronisation zwischen Kunden- und Banksystem

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsResponse
 xmlns="urn:org:ebics:H004"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation=" urn:org:ebics:H004 ebics_response_H004"
 Version="H004" Revision="1">
 <header authenticate="true">
   <static>     <TransactionID>ABCDEF4139464436344531324 3ABCDEF</TransactionID>
   </static>
   <mutable>
     <TransactionPhase>Transfer</TransactionPhase>
     <SegmentNumber>2</SegmentNumber>
     <OrderId>OR01</OrderId>
     <ReturnCode>061101</ReturnCode>
     <ReportText>[EBICS_TX_RECOVERY_SYNC] Synchronisation erforderlich</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> …hier Hashwert Authentifikation..</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue> …hier Signaturwert Authentifikation..</ds:SignatureValue>
  </AuthSignature>
  <body>
    <ReturnCode authenticate="true">000000</ReturnCode>
  </body>
</ebicsResponse>
```

Abbildung 56: EBICS-Response mit technischem Fehler EBICS_TX_RECOVERY_SYNC

Formatiert: Deutsch (Deutschland)

Feldfunktion geändert

Formatiert: Deutsch (Deutschland)

5.6 Download-Transaktionen

5.6.1 Ablauf von Download-Transaktionen

Der Ablauf einer Download-Transaktion ist in Abbildung 57 anhand eines Sequenzdiagramms dargestellt. Dieses Sequenzdiagramm zeigt den Austausch von EBICS-Nachrichten in den einzelnen Phasen einer Download-Transaktion. Das erste Nutzdatensegment ist bereits in der EBICS-Response der Transaktionsinitialisierung enthalten. Alle weiteren Nutzdatensegmente werden in einer Schleife übertragen, die abbricht, sobald das letzte Datensegment vom Kundensystem empfangen wurde. (Siehe Abbruchbedingung der Schleife „[letztes Datensegment wurde empfangen]“). Schließlich wird der erfolgreiche Empfang aller Nutzdatensegmente vom Kundensystem quittiert.

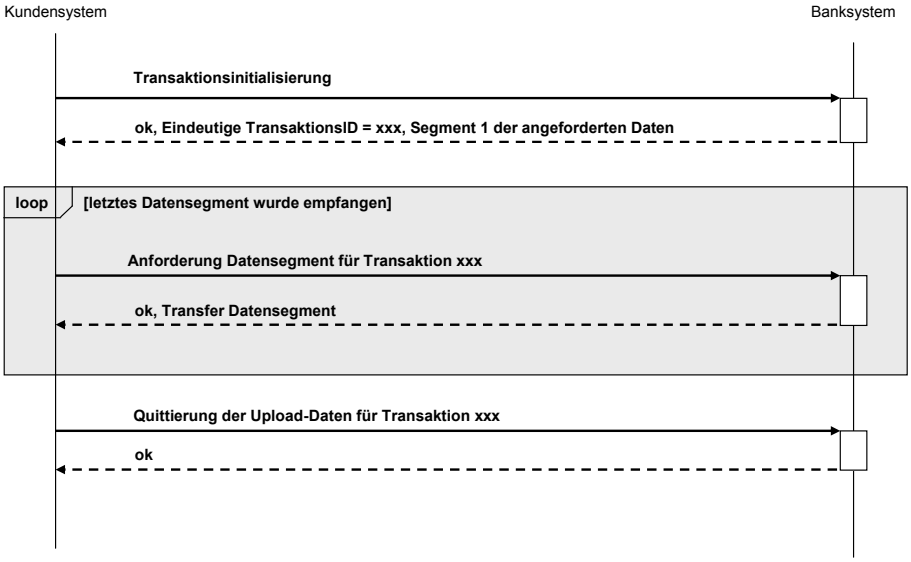


Abbildung 57: Fehlerfreier Ablauf einer Download-Transaktion

5.6.1.1 Beschreibung der EBICS-Nachrichten

Die nachfolgende Beschreibung der Transaktionsschritte einer Download-Transaktion verwendet zur Verdeutlichung Beispielnachrichten für die Durchführung eines STA-Auftrags. Sie nimmt Bezug auf Elemente dieser Beispielnachrichten unter Verwendung der XPath-Notation.

Die folgenden Kapitel beschreiben die EBICS-Nachrichten der einzelnen Phasen einer Download-Transaktion. Es werden die Daten aufgezählt, die Bestandteil dieser Nachrichten sind. Daten, die grundsätzlich optional sind, werden mit „(optional)“ gekennzeichnet. Daten, die nur unter bestimmten Voraussetzungen fehlen dürfen, werden stattdessen mit „(bedingt)“ gekennzeichnet. Optionale XML-Elemente, die in der Beschreibung einer EBICS-Nachricht zu einer bestimmten Transaktionsphase fehlen, dürfen in dieser EBICS-Nachricht nicht vorkommen. Optionale XML-Elemente, die in der Beschreibung einer EBICS-Nachricht zu einer bestimmten Transaktionsphase vorkommen, MÜSSEN für diese EBICS-Nachricht immer entsprechend gesetzt sein.

EBICS-Requests von Download-Transaktionen sind zu ebics_request_H004.xsd konforme (XML-)Instanzdokumente bestehend aus dem Toplevel-Element `ebics`, welches in ebics_request_H004.xsd deklariert ist. EBICS-Responses von Download-Transaktionen sind zu ebics_response_H004.xsd konforme Instanzdokumente, die aus dem Toplevel-Element `ebics` bestehen, das wiederum in ebics_response_H004.xsd deklariert ist.

5.6.1.1.1 EBICS-Nachrichten der Transaktionsinitialisierung

- Übertragung der folgenden Daten im EBICS-Request (siehe auch Beispiel in Abbildung 58):

  - Host-ID des EBICS Bankrechners (`ebicsRequest/header/static/HostID`)
  - Transaktionsphase (`ebicsRequest/header/mutable/TransactionPhase`) mit der Belegung „Initialisation“
  - Kombination aus Nonce und Timestamp für die Vermeidung von Replay alter EBICS-Nachrichten (`ebicsRequest/header/static/Nonce`, `ebicsRequest/header/static/Timestamp`)
  - Teilnehmer (`ebicsRequest/header/static/PartnerID`, `ebicsRequest/header/static/UserID`), welcher den Auftrag einreicht oder bankfachliche EUs zu einem bestehenden Auftrag nachliefert
  - (bedingt) technischer Teilnehmer (`ebicsRequest/header/static/PartnerID`, `ebicsRequest/header/static/SystemID`)
    `SystemID` muss genau dann vorhanden sein, wenn es sich bei dem Kundensystem um ein Multi-User-System handelt. Der technische Teilnehmer ist für das Erstellen der EBICS-Requests (inklusive der Authentifkationssignaturen) zuständig, die zu Aufträgen gehören, die vom Teilnehmer eingereicht oder bankfachlich unterzeichnet werden
  - (optional) Information zum Kundenprodukt (`ebicsRequest/header/static/Product`)
  - Auftragsart (ebicsRequest/header/static/OrderDetails/OrderType)
  - Auftragsattribute (`ebicsRequest/header/static/OrderDetails/OrderAttribute`)
    Setzt der Teilnehmer die Auftragsattribute gleich „DZHNN“, so fordert er die Abholdaten ohne die EU des Kreditinstituts an. Setzt er sie hingegen gleich „OZHNN“, so fordert er die Abholdaten mit der EU des Kreditinstituts an
  - Auftragsparameter (`ebicsRequest/header/static/OrderDetails/StandardOrderParams`).
    Die Ausprägung der Auftragsparameter ist abhängig von der Auftragsart. Für STA sind die Auftragsparameter vom Typ `StandardOrderParamsType`
  - Hashwerte der öffentlichen Schlüssel des Kreditinstituts, über die der Teilnehmer verfügt (ebicsRequest/header/static/BankPubKeyDigests/Authentication, ebicsRequest/header/static/BankPubKeyDigests/Encryption,

Formatiert: Deutsch (Deutschland)

ebicsRequest/header/static/BankPubKeyDigests/Signature).
Zu jedem dieser Hashwerte wird sowohl der verwendete Hashalgorithmus als auch die Version des entsprechenden Authentifikations- bzw. Verschlüsselungs- bzw. Signaturverfahrens angegeben.
In der Version „H004“ des EBICS-Protokolls ist die EU der Kreditinstitute nur vorgesehen (siehe dazu Kapitel 3.5.2). Das Element BankPubKeyDigests/Signature ist in Vorbereitung zukünftiger EBICS-Versionen bereits in dieser Beschreibung enthalten, obwohl seine maximale Häufigkeit (maxOccurs) in der Version „H004“ gleich 0 ist

- Sicherheitsmedium für den bankfachlichen Schlüssel des Teilnehmers (`ebicsRequest/header/static/SecurityMedium`)
- Authentifikationssignatur des technischen Teilnehmers, falls ein solcher vorhanden ist, ansonsten die Authentifikationssignatur des Teilnehmers selbst (`ebicsRequest/AuthSignature`)
  Die Authentifikationsignatur umfasst alle XML-Elemente des EBICS-Requests, deren Attributwert für `@authenticate` gleich „true“ ist. Die Definition des XML-Schemas „ebics_request_H004.xsd“ gewährleistet, dass der Wert des Attributs `@authenticate` für genau die Elemente gleich „true“ ist, die auch unterschrieben werden müssen

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
 xmlns="urn:org:ebics:H004"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_request_H004.xsd"
 Version="H004" Revision="1">
 <header authenticate="true">
   <static>
     <HostID>EBIXHOST</HostID>
     <Nonce>98498A65465C645E645F64565462C645</Nonce>
     <Timestamp>2005-01-30T15:40:45.123Z</Timestamp>
     <PartnerID>PARTNER1</PartnerID>
     <UserID>USER0001</UserID>
     <Product Language="de" InstituteID="Institutskennung">Kundenproduktkennung</Product>
     <OrderDetails>
       <OrderType>STA</OrderType>
       <OrderAttribute>DZHNN</OrderAttribute>
       <StandardOrderParams>
         <DateRange>
           <Start>2005-01-01</Start>
           <End>2005-01-30</End>
         </DateRange>
       </StandardOrderParams>
     </OrderDetails>
     <BankPubKeyDigests>
       <Authentication Version="X002"
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256">1H/rQr2Axe9hYTV2n/tCp+3UIQQ=</Authenticati
on>
       <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> …hier Hashwert Authentifikation..</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue> …hier Signaturwert Authentifikation..</ds:SignatureValue>
  </AuthSignature>
  <body/>
</ebicsRequest>
```

Abbildung 58: EBICS-Request für die Transaktionsinitialisierung für die Auftragsart STA

Formatiert: Deutsch (Deutschland)

Feldfunktion geändert

Formatiert: Deutsch (Deutschland)

- Übertragung der folgenden Daten in der EBICS-Response (siehe Beispiel in Abbildung 59):
  - bankfachlicher Returncode (`ebicsResponse/body/ReturnCode`)
  - technischer Returncode (ebicsResponse/header/mutable/ReturnCode)
  - technischer Reporttext (ebicsResponse/header/mutable/ReportText)
  - (bedingt) banksystemweit eindeutige TransaktionsID (`ebicsResponse/header/static/TransactionID`), falls während der Transaktionsinitialisierung keine Fehler technischer oder bankfachlicher Art aufgetreten sind
  - Transaktionsphase (`ebicsResponse/header/mutable/TransactionPhase`) mit der Belegung „Initialisation“
  - (bedingt) Gesamtzahl zu übertragender Nutzdatensegmente (`ebicsResponse/header/static/NumSegments`), falls keine Fehler technischer oder bankfachlicher Art aufgetreten sind
  - (bedingt) laufende Nummer des in dieser Response übermittelten Nutzdaten-segments (`ebicsResponse/header/mutable/SegmentNumber`), falls keine Fehler technischer oder bankfachlicher Art aufgetreten sind. `SegmentNumber` ist in der Initialisierungsphase stets mit 1 belegt. Das Attribut `ebicsResponse/header/mutable/SegmentNumber@lastSegment` gibt an, ob es sich dabei um das letzte Datensegment handelt

- Authentifikationssignatur des Kreditinstituts (`ebicsResponse/AuthSignature`)
  Die Authentifikationsignatur umfasst alle XML-Elemente der EBICS-Response, deren Attributwert für `@authenticate` gleich „true“ ist. Die Definition des XML-Schemas „ebics_response_H004.xsd“ gewährleistet, dass der Wert des Attributs `@authenticate` für genau die Elemente gleich „true“ ist, die auch unterschrieben werden müssen
- (bedingt) Information zur Verschlüsselung der Auftragsdaten und eventuell der EU der Auftragsdaten (`ebicsResponse/body/DataTransfer/DataEncryptionInfo`), falls keine Fehler technischer oder bankfachlicher Natur aufgetreten sind. `DataEncryptionInfo` enthält insbesondere auch den asymmetrisch verschlüsselten Transaktionsschlüssel (`ebicsResponse/body/DataTransfer/DataEncryptionInfo/TransactionKey`)
- (bedingt) das erste Nutzdatensegment (`ebicsResponse/body/DataTransfer/OrderData`), falls keine Fehler technischer oder bankfachlicher Natur aufgetreten sind
- (bedingt) die bankfachliche EU der Nutzdaten durch das Kreditinstitut (`ebicsResponse/body/DataTransfer/SignatureData`), falls keine Fehler technischer oder bankfachlicher Natur aufgetreten sind und die Auftragsattribute gleich „OZHNN“ sind.
  In der Version „H004“ des EBICS-Protokolls ist die EU der Kreditinstitute nur vorgesehen (siehe dazu Kapitel 3.5.2). Die Belegung „OZHNN“ ist in der EBICS-Version „H004“ nicht zulässig und somit die Bedingung für das Vorhandensein des Elements `SignatureData` nicht erfüllt. Das Element `SignatureData` ist in dieser Beschreibung in Vorbereitung zukünftiger EBICS-Versionen dennoch enthalten
- (optional) Zeitstempel der letzten Aktualisierung der Bankparameter (`ebicsResponse/body/TimestampBankParameter`).

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsResponse
xmlns="urn:org:ebics:H004"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:org:ebics:H004 ebics_response_H004.xsd "
Version="H004" Revision="1">
<header authenticate="true">
  <static>
    <TransactionID>FEDCBA4139464436345313243FEDCBA</TransactionID>
    <NumSegments>2</NumSegments>
  </static>
  <mutable>
    <TransactionPhase>Initialisation</TransactionPhase>
    <SegmentNumber>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> …hier Hashwert Authentifikation..</ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
    <ds:SignatureValue> …hier Signaturwert Authentifikation..</ds:SignatureValue>
</AuthSignature>
<body>
  <DataTransfer>
    <DataEncryptionInfo authenticate="true">
      <EncryptionPubKeyDigest Version="E002"
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256">..hier Hashwert des öffentlichen
Bankschlüssels für die Verschlüsselung ..</EncryptionPubKeyDigest>
      <TransactionKey>En6KEB6ArEzw+iq4N1wm6Eptcyx…</TransactionKey>
      </DataEncryptionInfo>
    <OrderData>…</OrderData>
  </DataTransfer>
  <ReturnCode authenticate="true">000000</ReturnCode>
</body>
</ebicsResponse>
```

Abbildung 59: EBICS-Response der Transaktionsinitialisierung für die Auftragsart STA

5.6.1.1.2 EBICS-Nachrichten der Phase des Datentransfers

- Übertragung der folgenden Daten im EBICS-Request (siehe auch Beispiel in Abbildung 60):
  - Host-ID des EBICS Bankrechners (`ebicsRequest/header/static/HostID`)
  - Daten zur Identifikation des aktuellen Transaktionsschritts:
    - TransaktionsID (ebicsRequest/header/static/TransactionID)
    - Transaktionsphase (`ebicsRequest/header/mutable/TransactionPhase`) mit der Belegung „Transfer“
    - laufende Nummer des Nutzdatensegments, das in diesem Transaktionsschritt abgeholt werden soll (`ebicsRequest/header/mutable/SegmentNumber`)
      Das Attribut

`ebicsRequest/header/mutable/SegmentNumber@lastSegment` hat für diesen EBICS-Request keine Bedeutung

- Authentifikationssignatur des technischen Teilnehmers, falls ein solcher vorhanden ist, ansonsten die Authentifikationssignatur des Teilnehmers selbst (`ebicsRequest/AuthSignature`)
  Die Authentifikationsignatur umfasst alle XML-Elemente des EBICS-Requests, deren Attributwert für `@authenticate` gleich „true“ ist. Die Definition des XML-Schemas „ebics_request_H004.xsd“ gewährleistet, dass der Wert des Attributs `@authenticate` für genau die Elemente gleich „true“ ist, die auch unterschrieben werden müssen.

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
 xmlns="urn:org:ebics:H004"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_request_H004.xsd"
 Version="H004" Revision="1">
 <header authenticate="true">
   <static>
     <HostID>EBIXHOST</HostID>
     <TransactionID>FEDCBA41394644363445313243FEDCBA</TransactionID>
   </static>
   <mutable>
     <TransactionPhase>Transfer</TransactionPhase>
     <SegmentNumber>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> …hier Hashwert Authentifikation../ds:DigestValue>
     </ds:Reference>
   </ds:SignedInfo> <ds:SignatureValue> …hier Signaturwert
Authentifikation..</ds:SignatureValue>
 </AuthSignature>
 <body/>
</ebicsRequest>
```

Abbildung 60: EBICS-Request für die Übertragung des nächsten Nutzdatensegments für die Auftragsart STA

- Übertragung der folgenden Daten in der EBICS-Response (siehe Beispiel in Abbildung 61)
  - bankfachlicher Returncode (`ebicsResponse/body/ReturnCode`)
  - technischer Returncode (ebicsResponse/header/mutable/ReturnCode)

- technischer Reporttext (ebicsResponse/header/mutable/ReportText)
- Daten zur Identifikation eines Transaktionsschritts
  Falls der technische Returncode den Wert EBICS_TX_RECOVERY_SYNC hat, identifiziert dieser Transaktionsschritt den letzten Wiederaufsetzpunkt der Download-Transaktion. Sind jedoch weder technische noch fachliche Fehler aufgetreten, so spiegelt dieser Transaktionsschritt den gerade aktuellen Transaktionsschritt wider:
  - TransaktionsID (ebicsResponse/header/static/TransactionID)
  - Transaktionsphase
    (ebicsResponse/header/mutable/TransactionPhase)
  - laufende Nummer des Nutzdatensegments
    (`ebicsResponse/header/mutable/SegmentNumber`).
    Es handelt sich hierbei um die Nummer des Nutzdatensegments, die vom EBICS-Request angefordert wurde oder im Falle des Fehlers EBICS_TX_RECOVERY_SYNC um die Nummer des letzten Nutzdatensegments, welches aus Sicht des Banksystems erfolgreich an das Kundensystem übermittelt wurde. Im Fall des Fehlers EBICS_TX_RECOVERY_SYNC ist der Wert für `SegmentNumber` immer gleich 1, falls der Wert für `TransactionPhase` „Initialisation“ ist.
    Das Attribut
    `ebicsResponse/header/mutable/SegmentNumber@lastSegment` gibt jeweils an, ob es sich um das letzte Nutzdatensegment handelt.
- Authentifikationssignatur des Kreditinstituts
  (`ebicsResponse/AuthSignature`)
  Die Authentifikationsignatur umfasst alle XML-Elemente der EBICS-Response, deren Attributwert für `@authenticate` gleich „true“ ist. Die Definition des XML-Schemas „ebics_response_H004.xsd“ gewährleistet, dass der Wert des Attributs `@authenticate` für genau die Elemente gleich „true“ ist, die auch unterschrieben werden müssen
- (bedingt) das angeforderte Nutzdatensegment
  (`ebicsResponse/body/DataTransfer/OrderData`), falls keine Fehler technischer oder bankfachlicher Natur aufgetreten sind.

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsResponse
 xmlns="urn:org:ebics:H004"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_response_H004.xsd"
 Version="H004" Revision="1">
 <header authenticate="true">
   <static>
```

```
      <TransactionID>FEDCBA4139464436344531324 3FEDCBA</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> …hier Hashwert Authentifikation..</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue> …hier Signaturwert Authentifikation..</ds:SignatureValue>
  </AuthSignature>
  <body>
    <DataTransfer>
      <OrderData>…</OrderData>
    </DataTransfer>
    <ReturnCode authenticate="true">000000</ReturnCode>
  </body>
</ebicsResponse>
```

Abbildung 61: EBICS-Response der Übertragung des letzten Nutzdatensegments für die Auftragsart STA

5.6.1.1.3 EBICS-Nachrichten der Quittierungsphase

- Übertragung der folgenden Daten im EBICS-Request (siehe auch Beispiel in Abbildung 62):
  - Host-ID des EBICS Bankrechners (`ebicsRequest/header/static/HostID`)
  - Daten zur Identifikation des aktuellen Transaktionsschritts:
    - TransaktionsID (ebicsRequest/header/static/TransactionID)
    - Transaktionsphase (`ebicsRequest/header/mutable/TransactionPhase`) mit der Belegung „Receipt“
  - Authentifikationssignatur des technischen Teilnehmers, falls ein solcher vorhanden ist, ansonsten die Authentifikationssignatur des Teilnehmers selbst (`ebicsRequest/AuthSignature`)
    Die Authentifikationsignatur umfasst alle XML-Elemente des EBICS-Requests, deren Attributwert für `@authenticate` gleich „true“ ist. Die

Definition des XML-Schemas „ebics_request_H004.xsd“ gewährleistet, dass der Wert des Attributs `@authenticate` für genau die Elemente gleich „true“ ist, die auch unterschrieben werden müssen

- Quittung (`ebicsRequest/body/TransferReceipt/ReceiptCode`): Der Wert der Quittung ist 0 („positive Quittung“), wenn die Abholung und Verarbeitung der Nutzdaten erfolgreich war. Andernfalls ist der Wert der Quittung 1 („negative Quittung“).

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
 xmlns="urn:org:ebics:H004"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_request_H004.xsd"
 Version="H004" 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> …hier Hashwert Authentifikation..</ds:DigestValue>
     </ds:Reference>
   </ds:SignedInfo>
   <ds:SignatureValue> …hier Signaturwert Authentifikation..</ds:SignatureValue>
 </AuthSignature>
 <body>
   <TransferReceipt authenticate="true">
     <ReceiptCode>0</ReceiptCode>
   </TransferReceipt>
 </body>
</ebicsRequest>
```

Abbildung 62: EBICS-Request für die Quittierung der Download-Daten

- Übertragung der folgenden Daten in der EBICS-Response (siehe Beispiel in Abbildung 63)
  - bankfachlicher Returncode (`ebicsResponse/body/ReturnCode`)
  - technischer Returncode (ebicsResponse/header/mutable/ReturnCode)
  - technischer Reporttext (ebicsResponse/header/mutable/ReportText)

- Daten zur Identifikation eines Transaktionsschritts:
  Falls der technische Returncode den Wert EBICS_TX_RECOVERY_SYNC hat, identifiziert dieser Transaktionsschritt den letzten Wiederaufsetzpunkt der Download-Transaktion. Sind jedoch weder technische noch fachliche Fehler aufgetreten, so spiegelt dieser Transaktionsschritt den gerade aktuellen Transaktionsschritt wider, also die Quittierung der Abholdaten.
- TransaktionsID (ebicsResponse/header/static/TransactionID)
- Transaktionsphase (ebicsResponse/header/mutable/TransactionPhase)
- (bedingt) laufende Nummer des Nutzdatensegments (`ebicsResponse/header/mutable/SegmentNumber`), falls der Fehler EBICS_TX_RECOVERY_SYNC aufgetreten ist und demzufolge der Wert für `TransactionPhase` „Initialisation“ oder „Transfer“ ist.
  Es handelt sich hierbei um die Nummer des Nutzdatensegments, welches aus Sicht des Banksystems zuletzt erfolgreich an das Kundensystem übermittelt wurde. Der Wert für `SegmentNumber` ist immer gleich 1, falls der Wert für `TransactionPhase` „Initialisation“ ist.
  Das Attribut `ebicsResponse/header/mutable/SegmentNumber@lastSegment` gibt jeweils an, ob es sich um das letzte Nutzdatensegment handelt.
- Authentifikationssignatur des Kreditinstituts (`ebicsResponse/AuthSignature`)
  Die Authentifikationsignatur umfasst alle XML-Elemente der EBICS-Response, deren Attributwert für `@authenticate` gleich „true“ ist. Die Definition des XML-Schemas „ebics_response_H004.xsd“ gewährleistet, dass der Wert des Attributs `@authenticate` für genau die Elemente gleich „true“ ist, die auch unterschrieben werden müssen.

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsResponse
 xmlns="urn:org:ebics:H004"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_response_H004.xsd"
 Version="H004" Revision="1">
 <header authenticate="true">
   <static>
     <TransactionID>FEDCBA4139464436344531324 3FEDCBA</TransactionID>
   </static>
   <mutable>
     <TransactionPhase>Receipt</TransactionPhase>
     <ReturnCode>011000</ReturnCode>
     <ReportText>[EBICS_POSTPROCESS_DONE] positive Quittung erhalten </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> …hier Hashwert Authentifikation..</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue> …hier Signaturwert Authentifikation..</ds:SignatureValue>
  </AuthSignature>
  <body>
    <ReturnCode authenticate="true">000000</ReturnCode>
  </body>
</ebicsResponse>
```

Abbildung 63: EBICS-Response der Quittierung der Download-Daten

5.6.1.2 Verarbeitung der EBICS-Nachrichten

Kapitel 5.6.1.1 beschreibt den Inhalt der EBICS-Nachrichten, die im Rahmen einer Download-Transaktion ausgetauscht werden. Gegenstand dieses Kapitels ist die Verarbeitung der EBICS-Nachrichten. Im Sequenzdiagramm in Abbildung 57 sind Aktionssequenzen hervorgehoben, deren Ablauf hier näher beschrieben wird.

Um die Beschreibung der Abläufe zu vereinfachen, wird angenommen, dass jeder Verarbeitungsschritt einen Returncode (RC) liefert, dessen Wert gleich EBICS_OK (000000) ist, wenn dieser Schritt erfolgreich durchgeführt werden konnte. In Abhängigkeit von RC werden der technische Returncode (RCT) und der bankfachliche Returncode (RCF) gesetzt, deren Werte dann in EBICS-Nachrichten einfließen.

- Die Überprüfung der Validität der EBICS-Requests erfolgt auf Grundlage der XML-Schema-Definitionsdatei „ebics_request_H004.xsd“ sowie unter Berücksichtigung der Einschränkungen, die in Kapitel 5.6.1.1 zu den einzelnen Requests festgelegt wurden. Normalerweise findet die Validitätsprüfung parallel und/oder verzahnt mit den anderen Schritten der Verarbeitung der EBICS-Requests statt. Die nachfolgenden Abläufe verzichten auf die Darstellung eines Ablaufschritts vom Typ „Validitätsprüfung des EBICS-Requests“ zugunsten einer möglichst einfachen Darstellung. Demnach können diese Abläufe durch folgende zusätzlichen technischen Fehler abgebrochen werden: EBICS_INVALID_XML, EBICS_INVALID_REQUEST oder EBICS_INVALID_REQUEST_CONTENT. Für Return Codes, die Zertifikate betreffen, wird auf Anhang 1 verwiesen.

5.6.1.2.1 Verarbeitung in der Initialisierungsphase

In Abbildung 64 ist die bankeitige Verarbeitung des EBICS-Requests dargestellt, welcher in der Initialisierungsphase einer Download-Transaktion vom Kundensystem an das Banksystem versendet wird. Die einzelnen Verarbeitungsschritte werden im Folgenden näher erläutert:

I. Erzeugung einer EBICS-Transaktion (siehe 5.5.1.2.1 Punkt 1 sowie Abbildung 49)

II. Beenden der EBICS-Transaktion

Sollten die angefordeten Abholdaten nicht verfügbar sein, so wird die EBICS-Transaktion mit dem fachlichen Returncode EBICS_NO_DOWNLOAD_DATA_AVAILABLE beendet.

III. Bereitstellen von Daten

In diesem Verarbeitungsschritt wird das erste Nutzdatensegment zwecks Einbettung in die EBICS-Response bereitgestellt. Falls das Kreditinstitut die bankfachliche EU für die aktuelle Auftragsart und den aktuellen Teilnehmer (Einreicher) einsetzt, wird zusätzlich die bankfachliche EU des Kreditinstituts über die Nutzdaten bereitgestellt.

In der Version „H004“ des EBICS-Protokolls ist die EU der Kreditinstitute nur vorgesehen (siehe dazu Kapitel 3.5.2). Sie wird hier nur in Vorbereitung zukünftiger EBICS-Versionen berücksichtigt.

Die Bereitstellung der Abholdaten ist nicht Bestandteil von EBICS, sie ist vielmehr abhängig von der Implementierung des Banksystems.

IV. Erzeugen der EBICS-Response

Dieser Verarbeitungsschritt erzeugt die EBICS-Response, die anschließend an das Kundensystem gesendet wird. Wenn alle bisherigen Verarbeitungsschritte erfolgreich waren, enthält diese EBICS-Nachricht das erste Segment der Nutzdaten und eventuell auch die bankfachliche Unterschrift der (gesamten) Nutzdaten. Im Fehlerfall transportiert diese EBICS-Nachricht den entsprechenden technischen oder fachlichen Fehlercode. Einzelheiten zum Inhalt dieser EBICS-Nachricht sind in Kapitel 5.6.1.1.1 beschrieben.

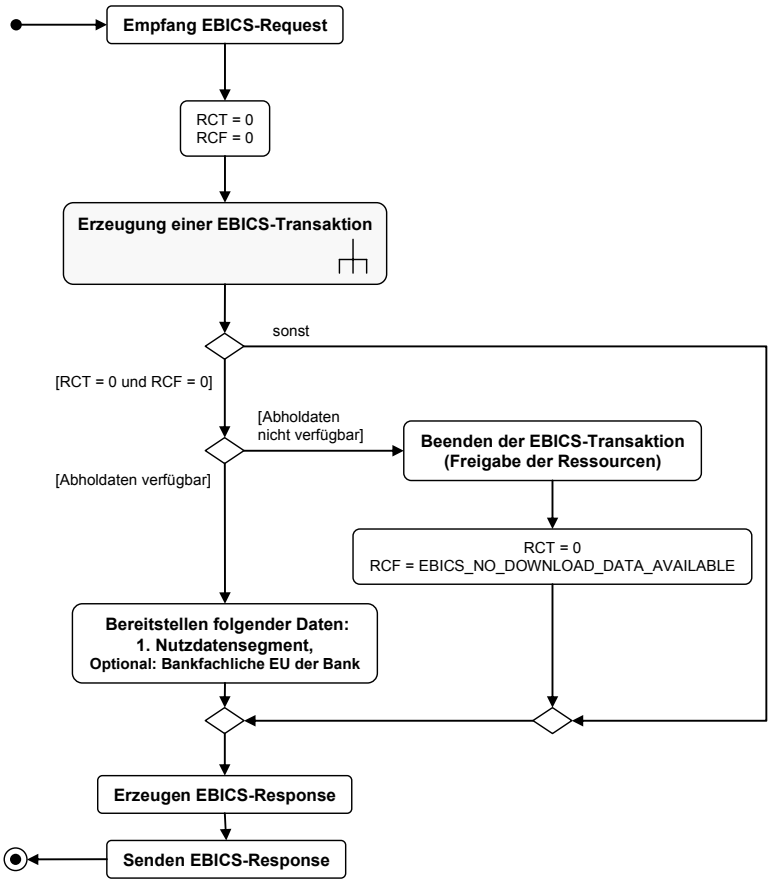


Abbildung 64: Verarbeitung des EBICS-Requests der Initialisierungsphase einer Download-Transaktion

5.6.1.2.2 Verarbeitung in der Phase des Datentransfers

In Abbildung 66 ist die bankseitige Verarbeitung des EBICS-Request dargestellt, welcher in der Datentransferphase einer EBICS-Transaktion vom Kundensystem an das Banksystem versendet wird. Die einzelnen Verarbeitungsschritte werden im Folgenden näher erläutert:

I. Überprüfung der Download-Transaktion (siehe Abbildung 65)

I.a. Überprüfung der EBICS-Transaktion (siehe 5.5.1.2.2, Punkt 1 sowie Abbildung 51)

I.b. Auswertung des Ergebnisses der Überprüfung der EBICS-Transaktion

Ist die Überprüfung des Transaktionsschritts nicht erfolgreich, dann:

- wird überprüft, ob die Download-Transaktion wiederaufgesetzt werden kann, falls das Banksystem Recovery von Transaktionen unterstützt. Die Prüfung erfolgt

gemäß der Beschreibung aus Kapitel 5.6.2. Ist diese Prüfung erfolgreich, so wird der technische Returncode EBICS_TX_RECOVERY_SYNC zurückgeliefert, andernfalls wird die Transaktion mit dem technischen Returncode EBICS_TX_ABORT abgebrochen.

- wird die Download-Transaktion mit dem fachlichen Fehlercode EBICS_RECOVERY_NOT_SUPPORTED abgebrochen, falls das Banksystem Recovery von Transaktionen nicht unterstützt. Wenn man im Ablaufdiagramm MAX gleich 0 setzt, wird auch der Fall berücksichtigt, in dem Recovery nicht unterstützt wird.

II. Bereitstellen von Daten

In diesem Verarbeitungsschritt wird das angeforderte Nutzdatensegment zwecks Einbettung in die EBICS-Response bereitgestellt. Die Bereitstellung der Abholdaten ist nicht Bestandteil von EBICS, sie ist vielmehr abhängig von der Implementierung des Banksystems.

III. Erzeugen der EBICS-Response

Dieser Verarbeitungsschritt erzeugt die EBICS-Response, die anschließend an das Kundensystem gesendet wird. Wenn alle bisherigen Verarbeitungsschritte erfolgreich waren, enthält diese EBICS-Nachricht das im entsprechenden EBICS-Request angeforderte Segment der Nutzdaten. Im Fehlerfall transportiert diese EBICS-Nachricht den entsprechenden technischen oder fachlichen Fehlercode. Der Inhalt dieser EBICS-Nachricht ist in Kapitel 5.6.1.1.2 detailliert beschrieben.

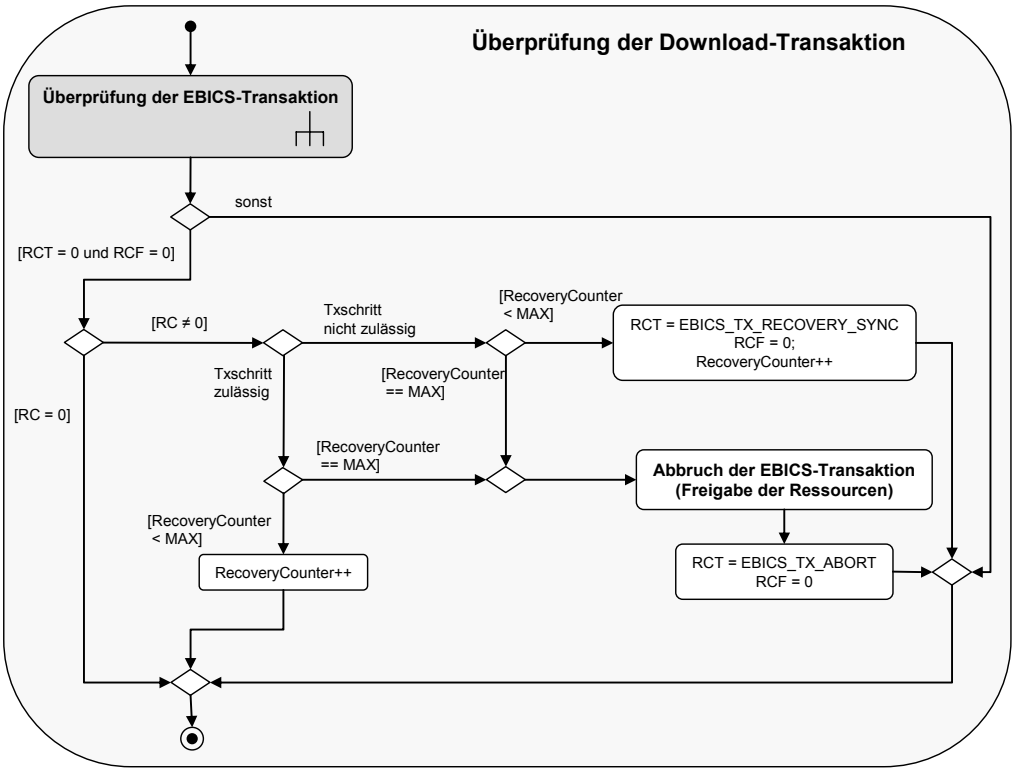


Abbildung 65: Detaillierte Beschreibung des Ablaufschritts „Überprüfung der Download-Transaktion“

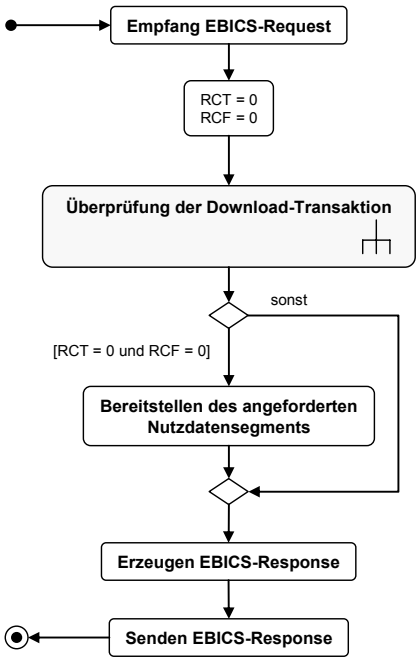


Abbildung 66: Verarbeitung eines EBICS-Requests zur Anforderung eines Nutzdatensegments

5.6.1.2.3 Verarbeitung in der Quittierungsphase

In Abbildung 67 ist die bankseitige Verarbeitung des EBICS-Request dargestellt, welcher in der Quittierungsphase einer EBICS-Transaktion vom Kundensystem an das Banksystem versendet wird.

Die einzelnen Verarbeitungsschritte werden im Folgenden näher erläutert:

I. Überprüfung der Download-Transaktion (siehe Beschreibung aus Kapitel 5.6.1.2.2, Punkt 1)

II. Download Nachbearbeitung

Die positive Quittung bedeutet, dass die Auftragsdaten vom Kundensystem erfolgreich abgeholt und verarbeitet werden konnten. Im Unterschied zur negativen Quittung hat sie zur Folge, dass auf dem Banksystem nun abschließende Arbeiten durchgeführt werden können, wie beispielsweise die Markierung der Auftragsdaten als „abgeholt. Unabhängig von der Art der Quittung wird die EBICS-Transaktion vom Banksystem beendet.

III. Beenden der EBICS-Transaktion

IV. Erzeugen der EBICS-Response

Dieser Verarbeitungsschritt erzeugt die EBICS-Response, die anschließend an das Kundensystem gesendet wird. Im Falle der positiven Quittung wird der technische Returncode EBICS_DOWNLOAD_POSTPROCESS_DONE, im Falle der negativen Quittung der technische Returncode EBICS_DOWNLOAD_POSTPROCESS_SKIPPED zurückgeliefert. Im Fehlerfall transportiert diese EBICS-Nachricht den entsprechenden technischen oder fachlichen Fehlercode. Der Inhalt dieser EBICS-Nachricht ist in Kapitel 5.6.1.1.3 genauer beschrieben.

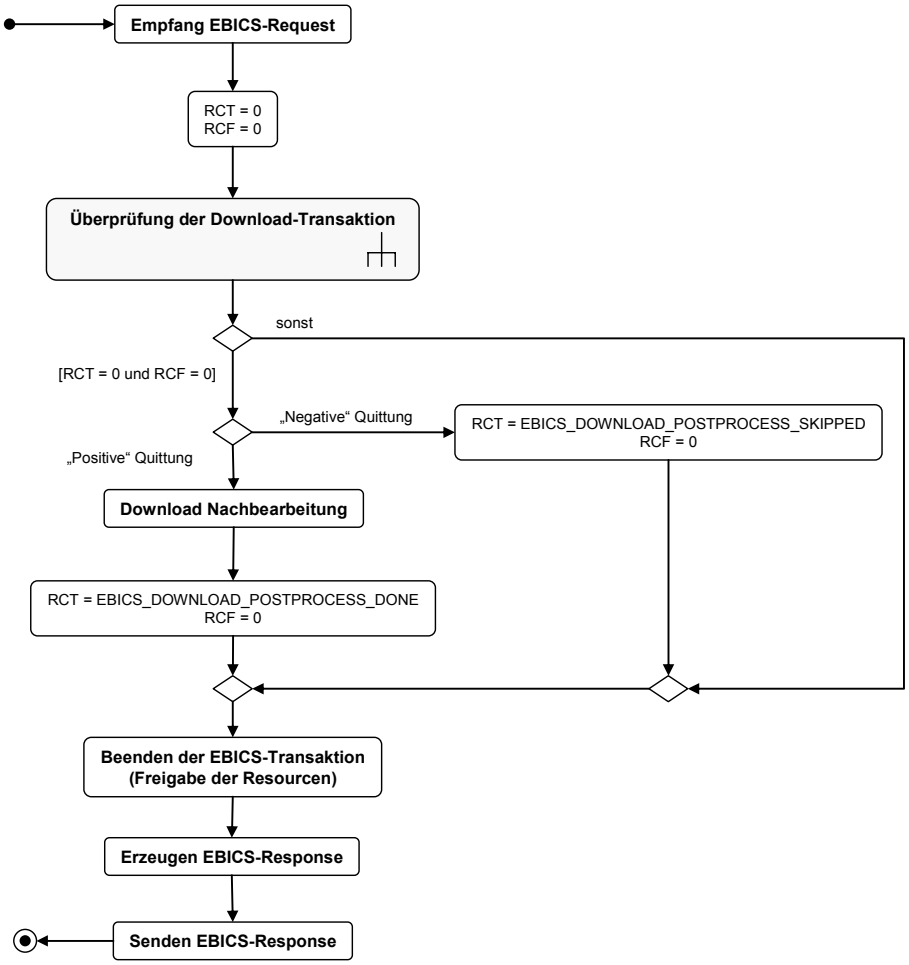


Abbildung 67: Verarbeitung eines EBICS-Requests zur Quittierung im Rahmen einer Download-Transaktion

5.6.2 Wiederaufsetzen von Download-Transaktionen

Recovery von Download-Transaktionen wird immer vom Kundensystem angestoßen. Die Ursachen für ein Recovery sind analog zu denen für Upload-Transaktionen:

- Transportfehler bei der Übertragung eines EBICS-Request in der Datentransfer- oder Quittierungsphase der Transaktion
- Timeout bzw. Transportfehler beim Empfang einer EBICS-Response in der Datentransfer- oder Quittierungsphase der Transaktion
- Verlust von bereits empfangenen Nutzdatensegmenten auf Teilnehmerseite
- Temporärer Fehler in der Verarbeitung einer empfangenen EBICS-Response, der eine erneute Übertragung erfordert.

Tritt eine der obigen Fehlersituationen ein, dann wählt das Kundensystem in Abhängigkeit von der Anzahl der vorhandenen Nutzdatensegmente auf Teilnehmerseite den geeigneten Wiederaufsetzpunkt aus. Ist der gewählte Wiederaufsetzpunkt die Anforderung des n-ten Nutzdatensegments, dann ist der nächste vom Teilnehmer initiierte Transaktionsschritt die Anforderung des (n+1)-ten Nutzdatensegments oder die Quittierung der Abholung sämtlicher Nutzdatensegmente, falls n das letzte Nutzdatensegment ist. EBICS-Requests im Rahmen des Recovery von Download-Transaktionen unterscheiden sich nicht von den EBICS-Requests eines normalen fehlerfreien Ablaufs einer Download-Transaktion.

In Abbildung 68 wird der beispielhafte Ablauf einer Transaktion dargestellt, die ein mehrfaches Wiederaufsetzen erfordert. Das Wiederaufsetzen geschieht jeweils ohne explizite Synchronisation zwischen Kunden- und Banksystem. Das 3. Nutzdatensegment wird dreimal angefordert, da das Kundensystem die entsprechende EBICS-Response wegen eines Timeout oder eines Transportfehlers nicht empfangen konnte. Bei der zweiten und dritten Anforderung des 3. Nutzdatensegments geht das Kundensystem davon aus, dass der Wiederaufsetzpunkt die Anforderung des 2. Nutzdatensegments ist. Der Wert des RecoveryCounters ist nach der dritten (und erfolgreichen) Anforderung des 3. Nutzdatensegments gleich 2, da die letzten beiden Anforderungen des 3. Nutzdatensegments vom Banksystem als Recovery-Versuch gewertet werden. Die Transaktion scheitert schließlich an der zu hohen Zahl von Wiederaufsetzversuchen.

Ist der gewählte Wiederaufsetzpunkt aus Sicht des Banksystems nicht zulässig, so enthält die EBICS-Response neben dem technischen Returncode EBICS_TX_RECOVERY_SYNC den letzten möglichen Wiederaufsetzpunkt der Download-Transaktion. Die zulässigen Wiederaufsetzpunkte einer Download-Transaktion sind in Kapitel 5.4 definiert. Ist der gewählte Wiederaufsetzpunkt beispielsweise die Anforderung des Nutzdatensegments mit der laufenden Nummer k, so kann die Transaktion mit der Anforderung der Nutzdatensegmente mit den laufenden Nummern i+1, i+2, usw. fortgesetzt werden, wobei i <= k gelten muss. Gilt allerdings i < k, wird also das i-te Nutzdatensegment wiederholt angefordert, dann wird der Zähler für die Anzahl der durchgeführten Recovery-Versuche um eins erhöht.

In Abbildung 69 ist der erfolgreiche Ablauf einer Transaktion dargestellt, der Wiederaufsetzen der Transaktion nach einer expliziten Synchronisation zwischen Kunden- und Bank-

system enthält. Hier fordert das Kundensystem das 5. Nutzdatensegment in einem Zustand an, ohne vorher das 4. Nutzdatensegment angefordert zu haben. Die EBICS-Response des Kreditinstituts (siehe Abbildung 70) enthält somit den Wiederaufsetzpunkt der Transaktion, der in diesem Fall die Anforderung des 3. Nutzdatensegments darstellt. Danach fährt das Kundensystem mit der Anforderung des 4. Nutzdatensegments fort und beendet die Transaktion nach dem Erhalt des letzten Segments 5.

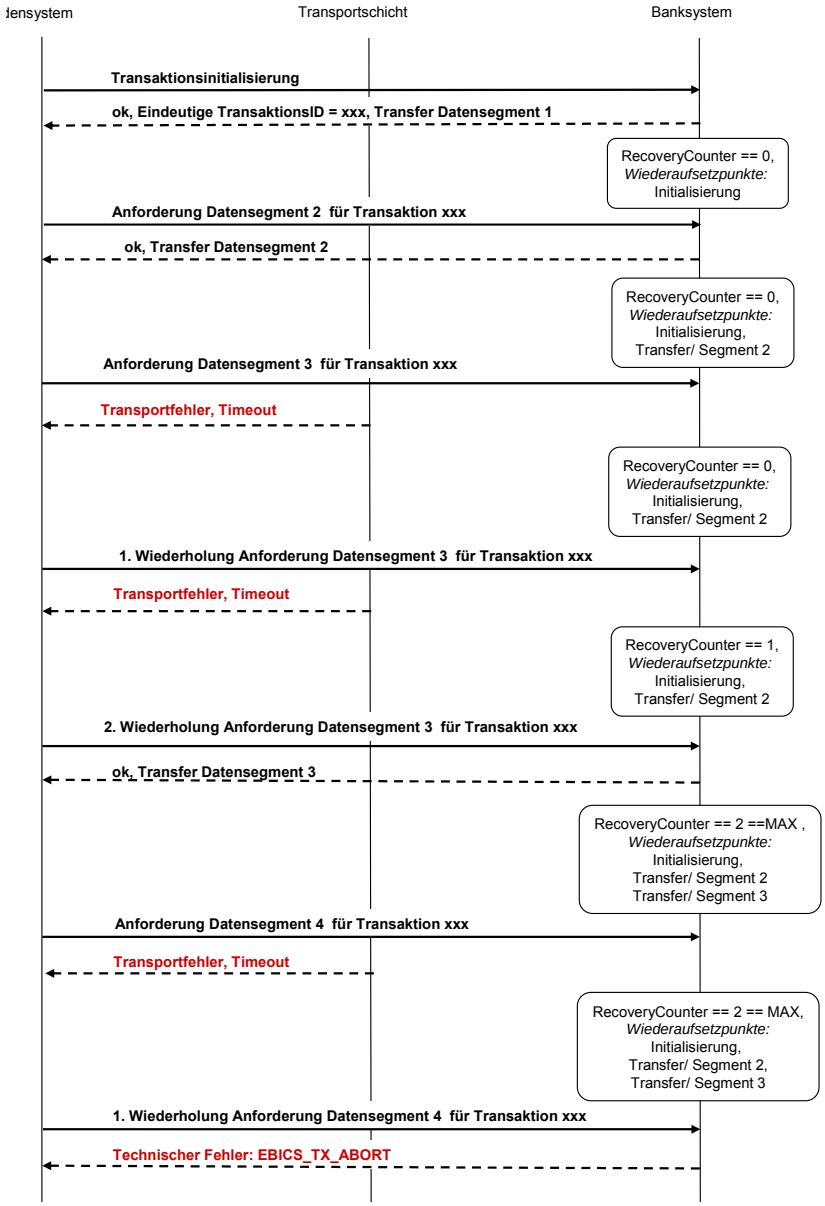


Abbildung 68: Abbruch des Recovery einer Download-Transaktion aufgrund der Überschreitung der maximal erlaubten Zahl von Recovery-Versuchen

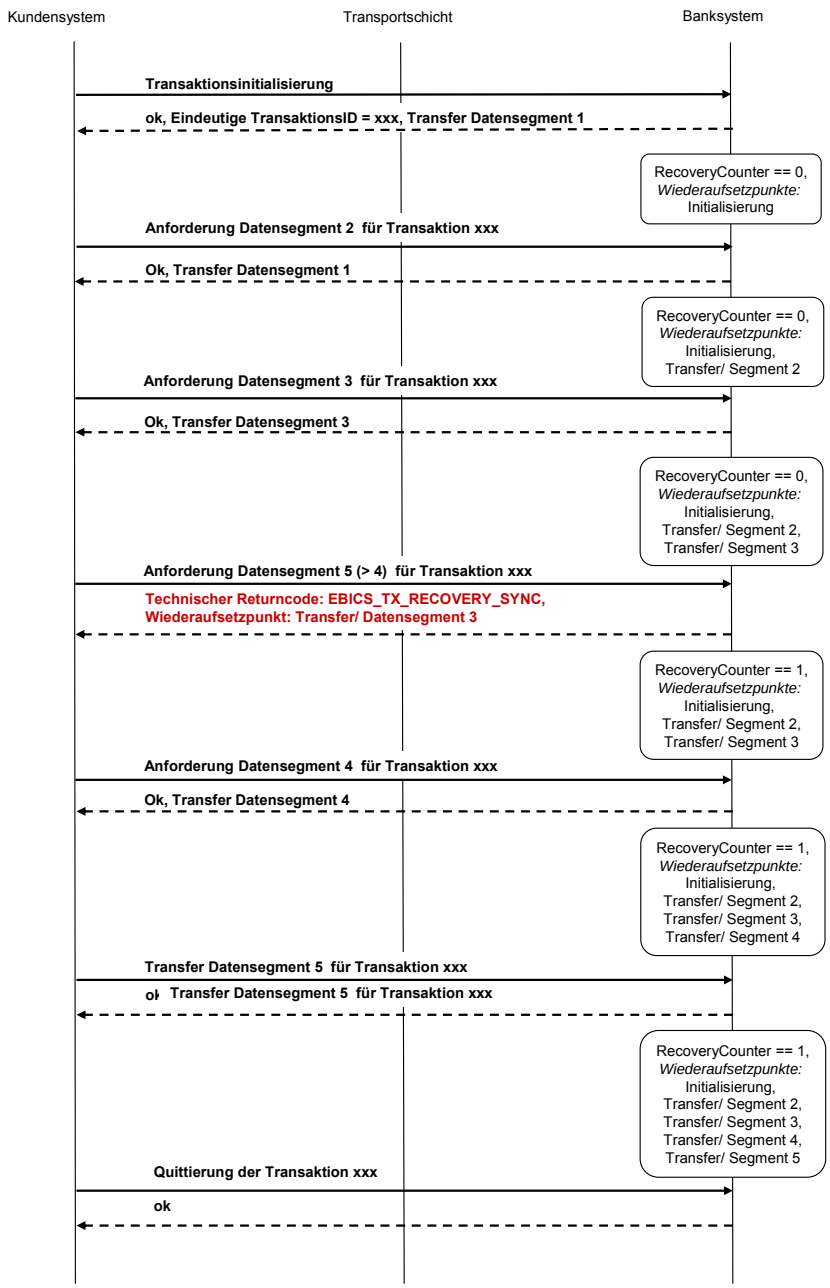


Abbildung 69: Recovery einer Download-Transaktion mit expliziter Synchronisation zwischen Kunden- und Banksystem

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsResponse
 xmlns=" urn:org:ebics:H004"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation=" urn:org:ebics:H004 ebics_response_H004.xsd "
 Version="H004" Revision="1">
 <header authenticate="true">
   <static>     <TransactionID>FEDCBA4139464436344531324 3FEDCBA</TransactionID>
   </static>
   <mutable>
     <TransactionPhase>Transfer</TransactionPhase>
     <SegmentNumber>3</SegmentNumber>
     <ReturnCode>061101</ReturnCode>
     <ReportText>[EBICS_TX_RECOVERY_SYNC] Synchronisation erforderlich</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> …hier Hashwert Authentifikation..</ds:DigestValue>
     </ds:Reference>
   </ds:SignedInfo>
   <ds:SignatureValue> …hier Signaturwert Authentifikation..</ds:SignatureValue>
 </AuthSignature>
 <body>
   <ReturnCode authenticate="true">000000</ReturnCode>
 </body>
</ebicsResponse>
```

Abbildung 70: EBICS-Response mit technischem Fehler EBICS_TX_RECOVERY_SYNC

6 Verschlüsselung

EBICS sieht Verschlüsselung auf zwei unterschiedlichen Protokoll-Ebenen vor: auf der Ebene des XML-basierten Anwendungsprotokolls von EBICS sowie auf einer Ebene zwischen Anwendungs- und Transportebene, nämlich auf der TLS-Ebene.

6.1 Verschlüsselung auf TLS-Ebene

Der Einsatz von TLS auf den externen Übertragungsstrecken zwischen Kunden- und Banksystem sichert die Geheimhaltung und die Integrität der EBICS-Nachrichten auf diesen Strecken. Die kryptographischen Verfahren, die zum Aufbau einer TLS-Session zwischen Kunden- und Banksystem verwendet werden, sind im Anhang (Kapitel 11.3.1) beschrieben.

6.2 Verschlüsselung auf Anwendungsebene

Die Auftragsdaten bankfachlicher Aufträge werden grundsätzlich als sensibel eingestuft und deshalb in verschlüsselter Form in die EBICS-Nachrichten eingebettet. Das ermöglicht ihre Geheimhaltung auf banksysteminternen sowie kundensysteminternen Übertragungsstrecken, auf denen die Kommunikation nicht notwendigerweise auf TLS basiert.

Die Auftragsdaten systembedingter Aufträge des Schlüsselmanagements werden verschlüsselt, sobald der Sender der Auftragsdaten über den (hinreichend überprüften) Verschlüsselungsschlüssel des Empfängers verfügt. Die Auftragsdaten von INI-, HIA- oder HSA-Aufträgen werden somit unverschlüsselt in die EBICS-Nachricht eingebettet, die Auftragsdaten von HPB-, PUB- , HCS oder HCA-Aufträgen hingegen verschlüsselt.

Die Auftragsdaten systembedingter Aufträge der Verteilten Elektronischen Unterschrift werden ebenfalls in verschlüsselter Form in die EBICS-Nachricht eingebettet.

Die elektronischen Unterschriften eines Auftrags, d.h. die Transportunterschrift bzw. die bankfachlichen EUs werden analog zu den Auftragsdaten bankfachlicher Aufträge immer verschlüsselt.

Außer den Auftragsdaten und den EUs werden auf Anwendungsebene keine weiteren Daten verschlüsselt.

Zu verschlüsselnde Auftragsdaten sowie EUs eines Auftrags werden zunächst mittels ZIP komprimiert, anschließend verschlüsselt und schließlich base64-kodiert in die EBICS-Nachricht eingebettet. Dabei erfolgt die Komprimierung und anschließende Verschlüsselung der Auftragsdaten vor deren Segmentierung. Das eingesetzte Verschlüsselungsverfahren ist ein Hybridverfahren: die Daten werden symmetrisch verschlüsselt, der eingesetzte symmetrische Schlüssel wird dem Empfänger der Daten asymmetrisch verschlüsselt mitgeteilt. Details des Verschlüsselungsverfahrens werden im Anhang (Kapitel 11.3.2) beschrieben.

Im Falle einer Upload-Transaktion wird im Kundensystem ein zufälliger symmetrischer Schlüssel generiert, welcher ausschließlich im Rahmen dieser Transaktion sowohl für die Verschlüsselung der EUs als auch für die Verschlüsselung der Nutzdaten eingesetzt wird. Dieser Schlüssel wird asymmetrisch mit dem öffentlichen Verschlüsselungsschlüssel des Kreditinstituts verschlüsselt und während der Initialisierungsphase der Transaktion vom Kundensystem an das Banksystem übertragen.

Analog wird im Falle einer Download-Transaktion im Banksystem ein zufälliger symmetrischer Schlüssel generiert, der für die Verschlüsselung der abzuholenden Nutzdaten und für die Verschlüsselung der vorgesehenen bankfachlichen Unterschrift des Kreditinstituts verwendet wird. Dieser Schlüssel wird asymmetrisch verschlüsselt und während der Initialisierungsphase der Transaktion vom Banksystem an das Kundensystem übertragen. Die asymmetrische Verschlüsselung erfolgt mit dem öffentlichen Verschlüsselungsschlüssel des technischen Teilnehmers, falls die EBICS-Nachrichten der Transaktion durch einen technischen Teilnehmer versendet werden. Andernfalls erfolgt die asymmetrische Verschlüsselung mit dem öffentlichen Verschlüsselungsschlüssel des nichttechnischen Teilnehmers, d.h. des Einreichers des Auftrags.

Ab EBICS 2.4 muss das Kundensystem in einen Request den Hashwert des öffentlichen Bankschlüssels E002 verwenden. Dieser Hashwert wird vom Kundensystem gemäß E002-Verfahren mittels SHA-256 gebildet.
Die Transaktion wird mit dem Returncode EBICS_INVALID_REQUEST_CONTENT abgebrochen, wenn noch E001 in einem Request verwendet wird.

7 Segmentierung der Auftragsdaten

7.1 Verfahrensbeschreibung

In der Version H004 des EBICS-Standards MÜSSEN Auftragsdaten, die in komprimierter, verschlüsselter und base64-kodierter Form mehr als 1 MB Speicherplatz benötigen, unabhängig von der Transferrichtung (Upload/Download) vor dem Versand segmentiert werden.

Bei der Segmentierung ist folgendes Vorgehen einzuhalten:

1. Die Auftragsdaten werden ZIP-komprimiert
2. Die komprimierten Auftragsdaten werden gemäß Kapitel 6.2 verschlüsselt
3. Die komprimierten, verschlüsselten Auftragsdaten werden base64-kodiert. Hierbei sind im resultierenden kodierten Datenblock bei EBICS nur die 65 druckbaren Zeichen des Base64-Alphabets aus RFC 2045 erlaubt. Insbesondere sind sogenannte „white space characters“ wie Leerzeichen, Tabs, Wagenrückläufe und Zeilenvorschübe („CR/LF“) unzulässig
4. Das Resultat ist hinsichtlich des Datenvolumens zu prüfen:

4i. Bleibt das resultierende Datenvolumen unterhalb der Grenze von 1 MB = 1.048.576 Bytes, so können die Auftragsdaten komplett innerhalb eines Versandschritts als ein Datensegment übermittelt werden

4ii. Übersteigt das resultierende Datenvolumen 1.048.576 Bytes, so sind die Daten sequenziell in Segmente von jeweils maximal 1.048.576 Bytes base64-konform aufzuteilen.

Mit Schritt 4i ist sichergestellt, dass auch diejenigen Auftragsdaten einheitlich im Rahmen der Segmentierung behandelt werden, welche in komprimierter, verschlüsselter und kodierter Form die zulässige maximale Segmentgröße von 1 MB nicht überschreiten.

Der Empfänger vollzieht die algorithmischen Berechnungen in umgekehrter Reihenfolge, um die ursprünglichen Auftragsdaten wieder herzustellen:

1. Das gerade empfangene Datensegment wird an die bereits empfangenen Datensegmente angehängt (konkateniert)
2. Der komplette Datenblock wird base64-dekodiert
3. Das Ergebnis der base64-Dekodierung wird gemäß Kapitel 6.2 entschlüsselt
4. Das Ergebnis der Entschlüsselung wird ZIP-dekomprimiert, um die ursprünglichen Auftragsdaten zu erhalten.

7.2 Umsetzung in den EBICS-Nachrichten

Der Sender der Auftragsdaten nummeriert die gemäß Kapitel 7.1 erzeugten Datensegmente sequenziell aufsteigend, beginnend mit 1.

Der Server bricht die Transaktion mit dem technischen Fehlercode EBICS_TX_SEGMENT_NUMBER_EXCEEDED ab, sofern der Client bei einer **Upload-Transaktion** die in der Initialisierungsphase angegebene Gesamtanzahl zu übertragender Segmente im Feld `ebicsRequest/header/static/NumSegments` zu niedrig angegeben hat, d.h. falls für den aktuellen Transaktionsschritt gilt:

- `ebicsRequest/header/mutable/SegmentNumber = ebicsRequest/header/static/NumSegments` (aus der Initialisierungsphase) und `ebicsRequest/header/mutable/SegmentNumber@lastSegment≠"true"`, oder
- `ebicsRequest/header/mutable/SegmentNumber > ebicsRequest/header/static/NumSegments` (aus der Initialisierungsphase).

Der Server beendet die Transaktion regulär mit dem technischen Hinweis-Returncode EBICS_TX_SEGMENT_NUMBER_UNDERRUN, sofern der Client bei einer **Upload-Transaktion** die in der Initialisierungsphase angegebene Gesamtanzahl zu übertragender Segmente im Feld `ebicsRequest/header/static/NumSegments` zu hoch angesetzt hat, d.h. falls für den aktuellen Transaktionsschritt gilt:

- `ebicsRequest/header/mutable/SegmentNumber < ebicsRequest/header/static/NumSegments` (aus der Initialisierungsphase) und `ebicsRequest/header/mutable/SegmentNumber@lastSegment="true"`.

Der Server bricht die Transaktion mit dem technischen Fehlercode EBICS_SEGMENT_SIZE_EXCEEDED ab, sofern der Client bei einer **Upload-Transaktion** die zulässige Segmentgröße von 1 MB im aktuellen Transaktionsschritt überschritten hat.

Bei **Download-Transaktionen** obliegt es dem Kundensystem, auf Unstimmigkeiten bzgl. der Segmentanzahl oder -größe zu reagieren:

- Ist die tatsächliche Anzahl übertragener Segmente bis zur Attributbelegung `ebicsResponse/header/mutable/SegmentNumber@lastSegment="true"` seitens des Servers niedriger als die Angabe aus der Initialisierungsphase, so SOLLTE der Client die laufende Transaktion dennoch ordnungsgemäß mit der Quittierungsphase fortsetzen.
- Überschreitet der Server die in der Initialisierungsphase postulierte Gesamtanzahl Segmente, so KANN der Client die Transaktion trotzdem durch Anfragen für weitere Segmente fortführen. Alternativ bzw. bei unverhältnismäßig großer Abweichung der

aktuellen Segmentanzahl zur Vorgabe KANN der Client die Transaktion unterbrechen, indem er keine weiteren Anfragen sendet.

- Überschreitet der Server die zulässige Segmentgröße von 1 MB, so SOLLTE der Client die Transaktion abbrechen.

8 Verteilte Elektronische Unterschrift (VEU)

Die bankseitige Unterstützung für die Verteilte Elektronische Unterschrift ist verbindlich für EBICS-konforme Implementationen, d.h. jedes Kreditinstitut MUSS VEU anbieten.
Die Auftragsarten der Verteilen Elektronischen Unterschrift sind ab der EBICS-Version 2.4 nur von deutschen Kreditinstituten verpflichtend zu unterstützen.

8.1 Beschreibung des Verfahrens

Die Verteilte Elektronische Unterschrift (VEU) erlaubt es, Aufträge orts- und zeitunabhängig von mehreren Teilnehmern – auch kundenübergreifend – zu autorisieren. Ein zugrunde liegender Auftrag bleibt hierbei in der VEU-Prozessverarbeitung gespeichert, bis über die VEU-Aufträge entweder die erforderliche Anzahl Unterschriften mit der passenden Berechtigung eingegangen sind, ein bankrechnerseitiges Zeitlimit überschritten wurde oder ein Auftragsstorno erfolgt. Das VEU-Verfahren ist somit nicht nur eine Alternative zur kundeninternen nachträglichen Einreichung von EUs zu einem bestehenden Auftrag, sondern bietet überdies auch eine kundenübergreifende verteilte EU mit reichhaltigen Informationsmöglichkeiten zum VEU-Status und zum Auftrag.

Unterschriftsberechtigte eines Kunden können voneinander abweichende Signaturverfahren verwenden. Diese können verschiedene Hashverfahren unterstützen, woraus sich auch unterschiedliche Hashwerte ergeben. Im Falle des VEU-Verfahrens wird bei den Auftragsarten HVD und HVZ immer der Hashwert der Auftragsdaten zur Verfügung gestellt, der sich aus der Unterschriftsversion ergibt, die der Teilnehmer, der HVZ bzw. HVD durchführt, verwendet. Die verwendete Unterschriftsversion wird in einem Attribut mitgegeben.

Ein kompletter Auftragsvorgang zur VEU läuft generell wie folgt ab:

1. Der Auftraggeber initiiert den zugrunde liegenden Auftrag (z.B. IZV), indem er die Auftragsdaten in einer EBICS-Transaktion mit den Auftragsattributen „OZHNN“ überträgt. Für die Unterschrift kann der Auftraggeber entweder den Auftrag sofort bankfachlich signieren (Unterschriftsklasse A, B oder E) oder zunächst nur die Übertragung mittels Transportunterschrift durchführen (Unterschriftsklasse T).
2. Das Banksystem analysiert Auftragsart und bereits geleistete Unterschriften samt ihrer Klasse. Falls zur Ausführung des Auftrags weitere Unterschriften erforderlich sind, wird der Auftrag mitsamt dem Hashwert für das VEU-Verfahren zwischengespeichert. Den Hashwert der Auftragsdaten extrahiert das Banksystem hierbei mittels des öffentlichen Signaturschlüssels des Unterzeichners aus der EU.
3. Möchte ein anderer Teilnehmer das VEU-Verfahren für diesen Auftrag nutzen, so hat er möglicherweise bereits die zur Autorisation erforderlichen Daten – Hashwert des Auftrags, Auftragsart und Auftragsnummer – außerhalb von

EBICS (über einen dritten Kommunikationsweg) erhalten. In diesem Fall geht es bei Punkt 4 weiter. Benötigt er hingegen noch die Auftragsdaten, so kann er folgendermaßen vorgehen:

3i. Zunächst fragt er mittels der Auftragsart HVU oder HVZ ab, für welche Aufträge im Rahmen der VEU er unterschriftsberechtigt ist. Die Antwort enthält u.a. Informationen zu Auftragsart, Auftragsnummer, zur Anzahl erforderlicher und bereits geleisteter Unterschriften (einschließlich dem Hinweis, ob die eigene Unterschrift noch aussteht oder schon geleistet wurde), zum ursprünglichen Auftraggeber und zur Größe der unkomprimierten Auftragsdaten. Die HVZ-Antwort enthält zusätzliche Informationen, insbesondere den Hashwert der Auftragsdaten. Somit kann Schritt 3ii übersprungen werden, wenn HVZ verwendet wurde.

3ii. Als nächstes ermittelt er mit der Auftragsart HVD den Status zu einem dieser Aufträge, z.B. zu dem im Rahmen der VEU eingestellten IZV-Auftrag. Neben dem Hashwert der Auftragsdaten, den das Banksystem aus der EU des Unterzeichners des Auftrags extrahiert hat, und einem Begleitzettel bekommt er eine Auflistung der bisherigen Unterzeichner samt ihrer jeweiligen Berechtigungsklasse.

3iii. Der Teilnehmer kann zusätzliche Auftragsdetails mit der Auftragsart HVT einholen: Je nach Anfrageparameter erhält er entweder Informationen zu den Einzeltransaktionen des Auftrags (Kontodaten, Betragsinformationen, Ausführungsdatum, Verwendungszweckdaten und weitere Beschreibungen) oder die kompletten Auftragsdaten.

4. Der Teilnehmer hat nun alle Informationen, um den Originalauftrag zu unterschreiben oder zu stornieren:

4i. Will er dem ursprünglichen Auftrag eine Unterschrift hinzufügen, so benutzt er die Auftragsart HVE. Hierzu signiert er den Hashwert der Auftragsdaten, den er mittels HVD übermittelt bekommen oder mittels HVT aus den kompletten Auftragsdaten selbst berechnet hat. Die HVE-Steuerdaten enthalten dabei die Auftragsparameter zum ursprünglichen Auftrag (z.B. dem IZV-Auftrag).

4ii. Möchte er den ursprünglichen Auftrag stornieren, so verwendet er die Auftragsart HVS. Wie bei HVE wird die Autorisation mittels der bankfachlichen Signatur über den Hashwert der Auftragsdaten bestätigt, nur dass im Falle von HVS die Unterschrift als Bestätigung des Stornos, nicht des Auftrags selbst gilt. Die HVS-Steuerdaten enthalten wie bei HVE die Auftragsparameter zum ursprünglichen (zu stornierenden) Auftrag.

Abbildung 71 dokumentiert die Abläufe bei Verwendung von VEU. Die Abbildung verdeutlicht die logische Verkettung der VEU-Auftragsarten, wobei reine Kommunikationsverbindungen (z.B. Datenversand vom Bank- zum Kundensystem bei Abruf der VEU-Details mit HVD), Fehlerfälle und die Informationsbeschaffung über alternative Kommunikationswege (z.B. Auftragshashwert per E-Mail vom Einreicher statt HVD) der Übersichtlichkeit halber nicht dargestellt werden.

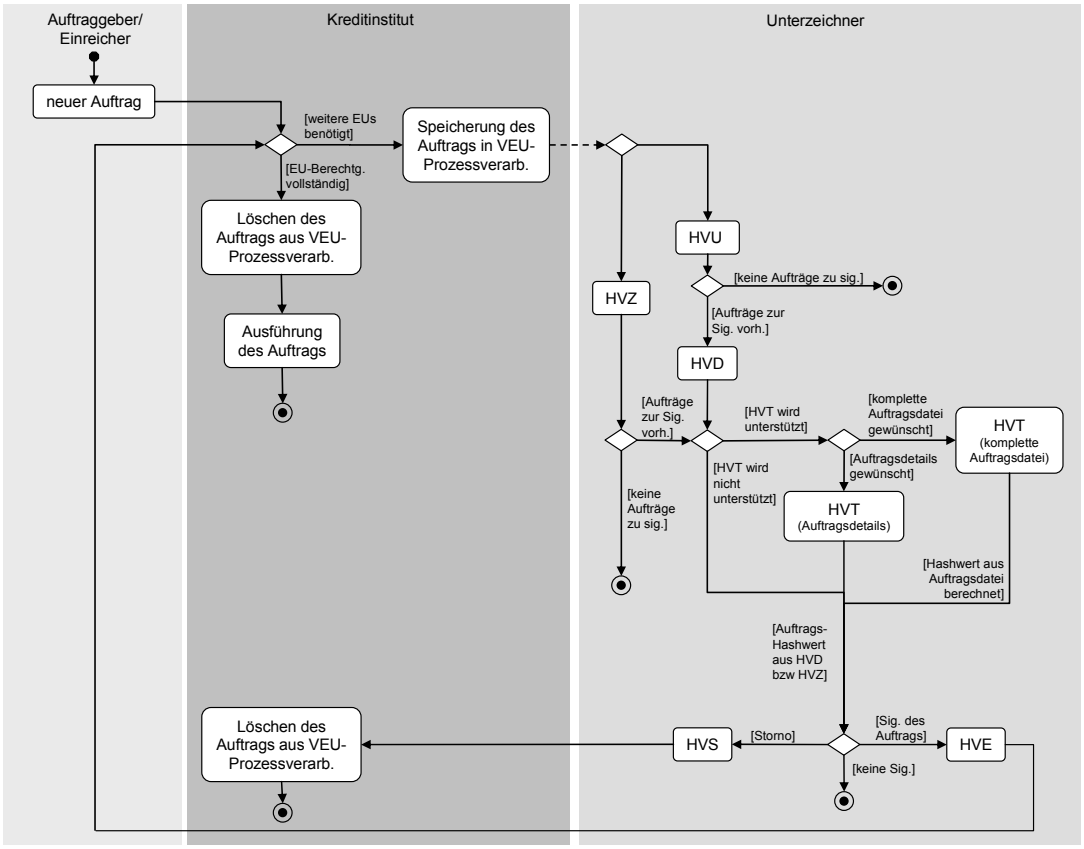


Abbildung 71: Ablaufdiagramm zur VEU

8.2 Technische Umsetzung der VEU

- Ein Teilnehmer stößt die VEU-Prozessverarbeitung an, indem er einen Auftrag mit einer unzureichenden Anzahl bankfachlicher Unterschriften der benötigten Berechtigungsklassen einreicht. Die Einreichung des Auftrags erfolgt in einer EBICS-Transaktion mit den Auftragsattributen „OZHNN“. In jedem Fall muss dieser Auftrag signiert eingereicht werden (entweder mit bankfachlicher Unterschrift der Klassen „A“, „B“ oder „E“ oder mit Transportunterschrift = Unterschriftsklasse „T“).
- Das Banksystem verifiziert zunächst die mitgelieferte(n) EU(s) und die Autorisation des Teilnehmers für die gegebene Auftragsart. Dann gleicht es die Anzahl und Unterschriftsklassen der gelieferten EU(s) mit den lokal hinterlegten EU-Anforderungen für die gegebene Auftragsart ab. Falls noch Unterschriften ausstehend sind, wird der Auftrag mitsamt den bereits übermittelten EUs in die VEU-Prozessverarbeitung eingestellt.

- Informationen über Aufträge, die sich derzeit in der VEU-Prozessverarbeitung befinden, können mit den VEU-Auftragsarten „HVU“, „HVZ“, „HVD“ und „HVT“ abgerufen werden. Notwendige Parameter zur Eingrenzung der Originalaufträge werden über die zusätzlichen Auftragsparameter `HVUOrderParams`, `HVZOrderParams`, `HVDOrderParams` bzw. `HVTOrderParams` übermittelt, die Bestandteil der Steuerdaten für diese Auftragsarten sind. Bei „HVU“, „HVZ“, „HVD“ und „HVT“ handelt es sich um Download-Transaktionen, bei denen die Informationen der Antwort in Form von XML-Dokumenten transparent in das Auftragsdatenfeld eingebettet werden. „Transparent“ bedeutet: Die XML-Strukturen werden binär interpretiert und genauso wie die Auftragsdaten anderer Auftragsarten vor dem Versand komprimiert, verschlüsselt und kodiert.
- Für die Auftragsarten HVE/HVS kann ein Teilnehmer die notwendigen Daten zur Identifikation des Originalauftrags (Hashwert des Auftrags, Auftragsart, Auftragsnummer) über folgende Wege beziehen:
  - HVU & HVD: Mit HVU erfährt er Auftragsart und Auftragsnummer, mit HVD den Hashwert des Auftrags. Der Hashwert stammt dabei aus der EU des Einreichers des Auftrags.
  - Alternativ zu HVU & HVD ist HVZ: Mit HVZ erfährt er Auftragsart und Auftragsnummer und den Hashwert des Auftrags. Der Hashwert stammt dabei aus der EU des Einreichers des Auftrags.
  - HVU & HVT: Mit HVU erfährt der Teilnehmer wie im Fall „HVU & HVD“ Auftragsart und Auftragsnummer. Bei HVT kann er in der Anfrage (`HVTOrderParams`) den Schalter `completeOrderData="true"` setzen und erhält damit die komplette Auftragsdatei. Zu dieser berechnet er selbst den Hashwert.
  - HVZ & HVT: Mit HVZ erfährt der Teilnehmer wie oben beschrieben Auftragsart, Auftragsnummer und Hashwert des Auftrags. Bei HVT kann er in der Anfrage (`HVTOrderParams`) den Schalter `completeOrderData="true"` setzen und erhält damit die komplette Auftragsdatei.
  - über einen alternativen Kommunikationsweg: Dem Teilnehmer steht es frei, sich die Informationen ohne Zuhilfenahme der EBICS-Schnittstelle zu beschaffen. Falls er z.B. Auftragsart und Auftragsnummer bereits kennt, kann er auf den Abruf mittels HVU verzichten. Ist er zusätzlich im Besitz des korrekten Hashwerts des Auftrags, so kann auch ein Aufruf von HVD, HVT bzw. HVZ entfallen.
- Neue EUs können dem Auftrag über die VEU-Auftragsart HVE zugeteilt werden. Die Identifikation des Originalauftrags geschieht hierbei über die zusätzlichen Auftragsparameter `HVEOrderParams`, die Bestandteil der Steuerdaten für einen HVE-Auftrag sind. Die hinzuzufügende EU über die Auftragsdaten des Originalauftrags wird beim Initialisierungsschritt übermittelt. HVE enthält eine oder mehrere EU(s), aber keine Auftragsdaten, und ist daher mit dem Auftragsattribut „UZHNN“ zu kennzeichnen.
- Sobald die geforderte Anzahl der geleisteten EUs mit den passenden Berechtigungen für die gegebene Auftragsart erreicht ist, wird der Originalauftrag aus der VEU-Prozessverarbeitung freigegeben und gelangt in die weitere Auftragsverarbeitung.

Damit taucht dieser Auftrag bei erneuter Ausführung von „HVU“ (bzw „HVZ“) nicht mehr in der Rückgabeliste der zu unterschreibenden Aufträge auf.

- Ein VEU-Storno kann über die Auftragsart HVS ausgelöst werden. Die Identifikation des Originalauftrags geschieht wie bei HVE über zusätzliche Auftragsparameter (hier `HVSOrderParams`), die Bestandteil der Steuerdaten für einen HVS-Auftrag sind. Die autorisierende EU für den Storno über die Auftragsdaten des Originalauftrags wird wie bei HVE beim Initialisierungsschritt übermittelt. Auch HVS enthält eine oder mehrere EU(s), aber keine Auftragsdaten, und ist daher mit dem Auftragsattribut „UZHNN“ zu kennzeichnen.
  Ein Auftragsstorno ist sofort wirksam und bedarf immer nur einer einzigen berechtigten Unterschrift der Klassen „E“, „A“ oder „B“. Ein stornierter Auftrag wird aus der VEU-Prozessverarbeitung entfernt; er gelangt nicht in die weitere Auftragsverarbeitung. Außerdem ist er bei einem erneuten Aufruf von „HVU“ (bzw „HVZ“) nicht mehr in der Liste zu unterschreibender Aufträge enthalten.

8.3 Detailbeschreibung der VEU-Auftragsarten

In diesem Kapitel werden ausschließlich die Unterschiede und Ergänzungen gegenüber EBICS-Standardaufträgen (siehe hierzu Kapitel 5) beleuchtet. Hier werden keine Prozessabläufe mehr erläutert (siehe hierzu die Kapitel 8.1 und 8.2), sondern für jede einzelne VEU-Auftragsart (Request und Response) werden Syntax und Semantik der betroffenen Elemente und Attribute des XML-Schemas definiert und anhand von Beispielen erläutert.

Die Definition der VEU-Auftragselemente (VEU-Auftragsparameter und VEU-Auftragsdaten) erfolgt im XML-Schema „ebics_orders_H004.xsd“. Typdefinitionen sind z.T. im XML-Schema „ebics_types_H004.xsd“ zu finden. Bei den textuellen Darstellungen werden die jeweils relevanten Passagen aus „ebics_orders_H004.xsd“ und „ebics_types_H004.xsd“ zusammengefasst aufgeführt.

8.3.1 HVU (VEU-Übersicht abholen) und HVZ (VEU-Übersicht mit Zusatzinformationen abholen) [verpflichtend]

Mit HVU bzw. HVZ kann sich ein Teilnehmer die Aufträge auflisten lassen, für die er unterschriftsberechtigt ist. Als Filterkriterium kann er die Liste im Request auf bestimmte Auftragsarten eingrenzen (`OrderTypes`). Die Response enthält neben den Auftragskennzeichen auch Größe der Auftragsdaten, Unterschriftskonditionen und Informationen zum Einreicher des Auftrags und zu den bisherigen Unterzeichnern (`OrderDetails`).

Die Antwortnachricht der Auftragsart HVZ enthält neben allen Informationen aus HVU auch Daten aus HVD. Die Auftragsart HVZ ('VEU-Übersicht mit Zusatzinformationen abholen') ist somit vergleichbar mit einer Kombination von HVU mit 1 bis n HVD’s.

HVU und HVZ sind Auftragsarten vom Typ „Download“.

8.3.1.1 HVU-Request

Im HVU-Request übergibt der Teilnehmer optional eine Liste von Auftragsarten als Filterkriterium. Nur Aufträge, deren Auftragsart in der übergebenen Liste enthalten ist, werden zurückgeliefert. Falls der Teilnehmer keine Auftragsartenliste als Eingrenzung übergibt, bekommt er sämtliche Aufträge aufgelistet, für die er unterschriftsberechtigt ist.

Ausprägung der `OrderParams` (Auftragsparameter) für HVU: `HVUOrderParams`

8.3.1.1.1 XML-Schema (grafische Darstellung)

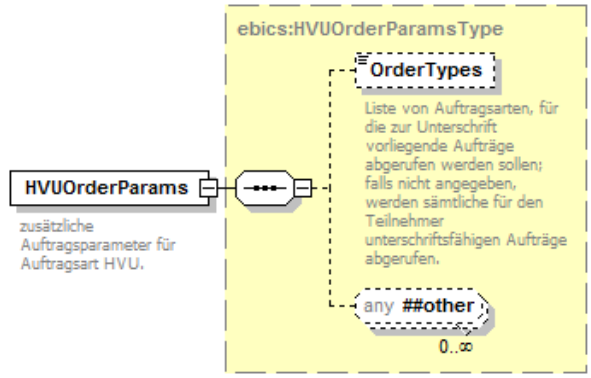


Abbildung 72: HVUOrderParams

8.3.1.1.2 XML-Schema (textuelle Darstellung)

```
 <element name="HVUOrderParams" type="ebics:HVUOrderParamsType"
substitutionGroup="ebics:OrderParams">
   <annotation>
     <documentation xml:lang="de">zusätzliche Auftragsparameter für Auftragsart
HVU.</documentation>
   </annotation>
 </element>
 <complexType name="HVUOrderParamsType">
   <annotation>
     <documentation xml:lang="de">Datentyp für zusätzliche Auftragsparameter für Auftragsart
HVU.</documentation>
   </annotation>
   <sequence>
      <element name="OrderTypes" type="ebics:OrderTListType" minOccurs="0">
       <annotation>
         <documentation xml:lang="de"> Liste von Auftragsarten, für die zur Unterschrift
vorliegende Aufträge abgerufen werden sollen; falls nicht angegeben, werden sämtliche für den
Teilnehmer unterschriftsfähigen Aufträge abgerufen.</documentation>
       </annotation>
     </element>
     <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
 </complexType>
```

8.3.1.1.3 Bedeutung der XML-Elemente/-Attribute

XML-Element/ -Attribut	Datentyp	#	Bedeutung	Beispiel
`HVUOrderParams`	`ebics:HVUOrderParamsType` (komplex)	1	Auftragsparameter für Auftragsart HVU	- (komplex)
`OrderTypes`	`ebics:OrderTListType` (→`list<ebics:OrderTBaseType>` →`list<token, length=3, pattern="[A-Z0-9]{3}">`)	0..1	Liste von Auftragsarten, für die zur Unterschrift vorliegende Aufträge abgerufen werden sollen; falls nicht angegeben, werden sämtliche Aufträge abgerufen, für die der Teilnehmer unterschriftsberechtigt ist	„IZV“

8.3.1.1.4 Beispiel-XML (gekürzt)

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
 xmlns=" urn:org:ebics:H004"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation=" urn:org:ebics:H004 ebics_request_H004.xsd "
 Version="H004" Revision="1">
  <header authenticate="true">
    <static>
      <!-- […] -->
      <OrderDetails>
        <OrderType>HVU</OrderType>
        <OrderAttribute>DZHNN</OrderAttribute>
        <HVUOrderParams>
          <OrderTypes>IZV</OrderTypes>
        </HVUOrderParams>
      </OrderDetails>
      <!-- […] -->
    </static>
    <!-- […] -->
  </header>
  <!-- […] -->
</ebicsRequest>
```

8.3.1.2 HVU-Response

In der HVU-Response werden dem Teilnehmer Informationen zu den Aufträgen zurückgeliefert, für die er unterschriftsberechtigt ist.

Ausprägung der (dekodierten & entschlüsselten & dekomprimierten) `OrderData` (Auftragsdaten) für HVU: `HVUResponseOrderData`

8.3.1.2.1 XML-Schema (grafische Darstellung)

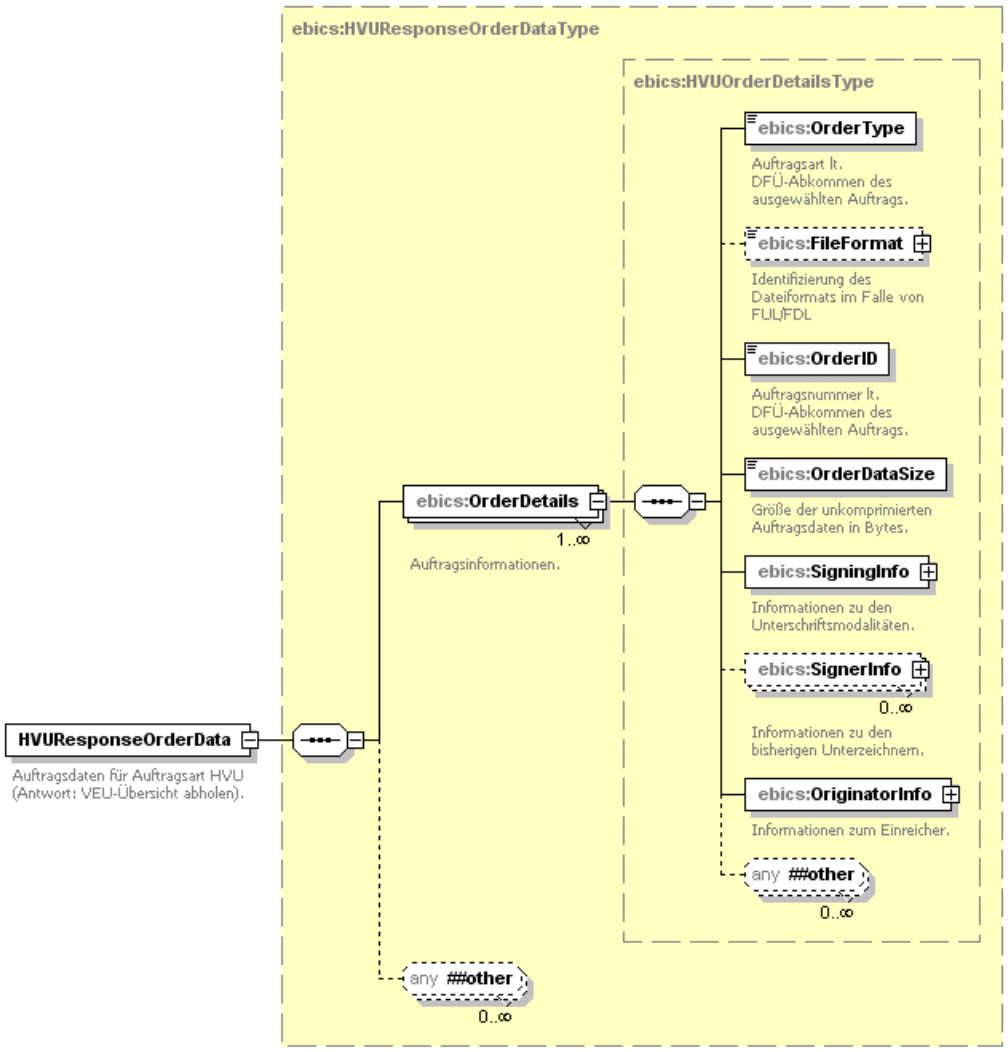


Abbildung 73: HVUResponseOrderData

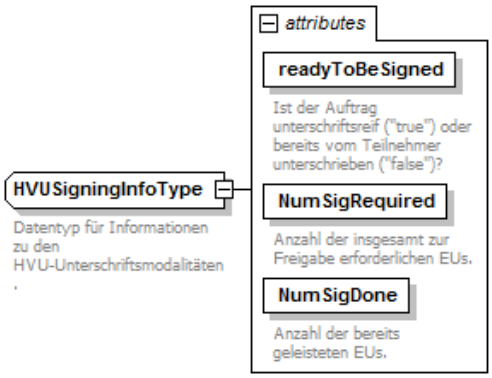


Abbildung 74: HVUSigningInfoType (zu SigningInfo)

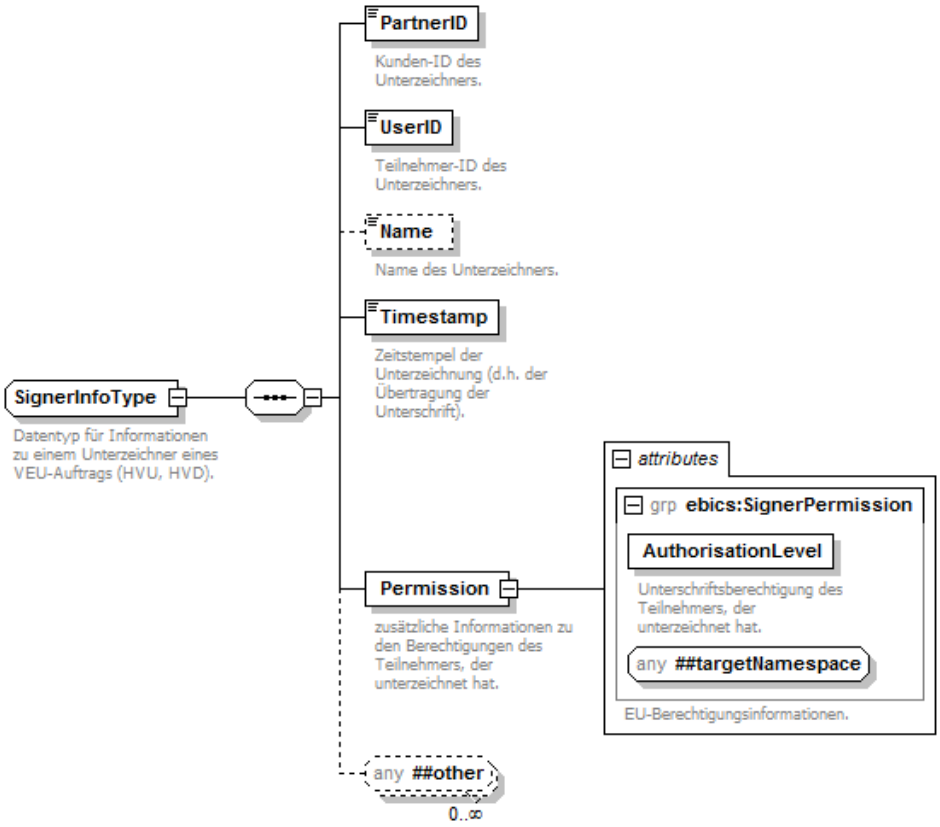


Abbildung 75: SignerInfoType (zu SignerInfo)

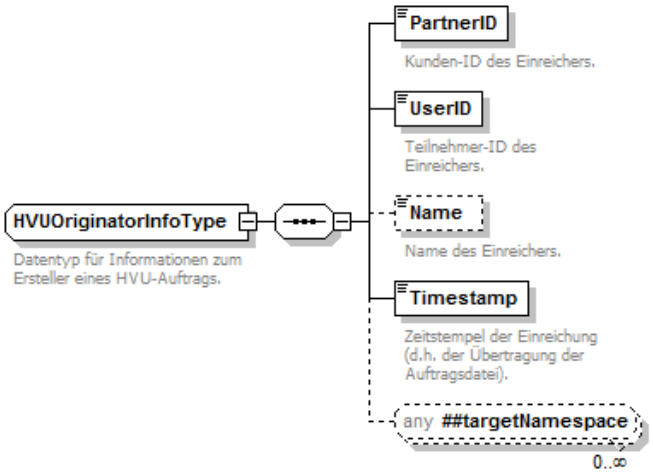


Abbildung 76: HVUOriginatorInfoType (zu OriginatorInfo)

8.3.1.2.2 XML-Schema (textuelle Darstellung)

```
 <element name="HVUResponseOrderData" type="ebics:HVUResponseOrderDataType"
substitutionGroup="ebics:EBICSOrderData">
   <annotation>
     <documentation xml:lang="de">Auftragsdaten für Auftragsart HVU (Antwort: VEU-Übersicht
abholen).</documentation>
   </annotation>
 </element>
 <complexType name="HVUResponseOrderDataType">
   <annotation>
     <documentation xml:lang="de">Datentyp für Auftragsdaten für Auftragsart HVU (Antwort:
VEU-Übersicht abholen).</documentation>
   </annotation>
   <sequence>
     <annotation>
       <documentation xml:lang="de"/>
     </annotation>
     <element name="OrderDetails" type="ebics:HVUOrderDetailsType" maxOccurs="unbounded">
       <annotation>
         <documentation xml:lang="de">Auftragsinformationen.</documentation>
       </annotation>
     </element>
     <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
 </complexType>
 <complexType name="HVUOrderDetailsType">
   <annotation>
     <documentation xml:lang="de">Datentyp für HVU-Auftragsdetails.</documentation>
   </annotation>
   <sequence>
     <element name="OrderType" type="ebics:OrderTBaseType">
       <annotation>
         <documentation xml:lang="de">Auftragsart gemäß Anhang (Kapitel 13) des ausgewählten
Auftrags.</documentation>
       </annotation>
     </element>
```

```
      <element name="OrderID" type="ebics:OrderIDType">
        <annotation>
          <documentation xml:lang="de">Auftragsnummer des ausgewählten Auftrags gemäß Kapitel
10.1.</documentation>
        </annotation>
      </element>
      <element name="OrderDataSize" type="positiveInteger">
        <annotation>
          <documentation xml:lang="de">Größe der unkomprimierten Auftragsdaten in
Bytes.</documentation>
        </annotation>
      </element>
      <element name="SigningInfo" type="ebics:HVUSigningInfoType">
        <annotation>
          <documentation xml:lang="de">Informationen zu den
Unterschriftsmodalitäten.</documentation>
        </annotation>
      </element>
      <element name="SignerInfo" type="ebics:SignerInfoType" minOccurs="0"
maxOccurs="unbounded">
        <annotation>
          <documentation xml:lang="de">Informationen zu den bisherigen
Unterzeichnern.</documentation>
        </annotation>
      </element>
      <element name="OriginatorInfo" type="ebics:HVUOriginatorInfoType">
        <annotation>
          <documentation xml:lang="de">Informationen zum Einreicher.</documentation>
        </annotation>
      </element>
      <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
  <complexType name="HVUSigningInfoType">
    <annotation>
      <documentation xml:lang="de">Datentyp für Informationen zu den HVU-
Unterschriftsmodalitäten.</documentation>
    </annotation>
    <attribute name="readyToBeSigned" type="boolean" use="required">
      <annotation>
        <documentation xml:lang="de">Ist der Auftrag unterschriftsreif ("true") oder bereits
vom Teilnehmer unterschrieben ("false")?</documentation>
      </annotation>
    </attribute>
    <attribute name="NumSigRequired" type="positiveInteger" use="required">
      <annotation>
        <documentation xml:lang="de">Anzahl der insgesamt zur Freigabe erforderlichen
EUs.</documentation>
      </annotation>
    </attribute>
    <attribute name="NumSigDone" type="nonNegativeInteger" use="required">
      <annotation>
        <documentation xml:lang="de">Anzahl der bereits geleisteten EUs.</documentation>
      </annotation>
    </attribute>
  </complexType>
  <complexType name="SignerInfoType">
    <annotation>
      <documentation xml:lang="de">Datentyp für Informationen zu einem Unterzeichner eines VEU-
Auftrags (HVU, HVD).</documentation>
    </annotation>
    <sequence>
      <element name="PartnerID" type="ebics:PartnerIDType">
```

```
        <annotation>
          <documentation xml:lang="de">Kunden-ID des Unterzeichners.</documentation>
        </annotation>
      </element>
      <element name="UserID" type="ebics:UserIDType">
        <annotation>
          <documentation xml:lang="de">Teilnehmer-ID des Unterzeichners.</documentation>
        </annotation>
      </element>
      <element name="Name" type="ebics:NameType" minOccurs="0">
        <annotation>
          <documentation xml:lang="de">Name des Unterzeichners.</documentation>
        </annotation>
      </element>
      <element name="Timestamp" type="ebics:TimestampType">
        <annotation>
          <documentation xml:lang="de">Zeitstempel der Unterzeichnung (d.h. der Übertragung der
Unterschrift).</documentation>
        </annotation>
      </element>
      <element name="Permission">
        <annotation>
          <documentation xml:lang="de">zusätzliche Informationen zu den Berechtigungen des
Teilnehmers, der unterzeichnet hat.</documentation>
        </annotation>
        <complexType>
          <attributeGroup ref="ebics:SignerPermission"/>
        </complexType>
      </element>
      <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
  <complexType name="HVUOriginatorInfoType">
    <annotation>
      <documentation xml:lang="de">Datentyp für Informationen zum Ersteller eines HVU-
Auftrags.</documentation>
    </annotation>
    <sequence>
      <element name="PartnerID" type="ebics:PartnerIDType">
        <annotation>
          <documentation xml:lang="de">Kunden-ID des Einreichers.</documentation>
        </annotation>
      </element>
      <element name="UserID" type="ebics:UserIDType">
        <annotation>
          <documentation xml:lang="de">Teilnehmer-ID des Einreichers.</documentation>
        </annotation>
      </element>
      <element name="Name" type="ebics:NameType" minOccurs="0">
        <annotation>
          <documentation xml:lang="de">Name des Einreichers.</documentation>
        </annotation>
      </element>
      <element name="Timestamp" type="ebics:TimestampType">
        <annotation>
          <documentation xml:lang="de">Zeitstempel der Einreichung (d.h. der Übertragung der
Auftragsdatei).</documentation>
        </annotation>
      </element>
      <any namespace="##targetNamespace" processContents="strict" minOccurs="0"
maxOccurs="unbounded"/>
    </sequence>
```

```
 </complexType>
 <attributeGroup name="SignerPermission">
   <annotation>
     <documentation xml:lang="de">EU-Berechtigungsinformationen.</documentation>
   </annotation>
   <attribute name="AuthorisationLevel" type="ebics:AuthorisationLevelType" use="required">
     <annotation>
       <documentation xml:lang="de">Unterschriftsberechtigung des Teilnehmers, der
unterzeichnet hat.</documentation>
     </annotation>
   </attribute>
   <anyAttribute namespace="##targetNamespace" processContents="strict"/>
 </attributeGroup>
```

8.3.1.2.3 Bedeutung der XML-Elemente/-Attribute

XML-Element/-Attribut	Datentyp	#	Bedeutung	Beispiel
HVUResponse» OrderData	ebics:HVUResponse» OrderDataType (komplex)	1	XML-Auftragsdaten für Auftragsart HVU	- (komplex)
OrderDetails	ebics:HVUOrder» DetailsType (komplex)	1..∞	Auftragsinformationen für Auftragsart HVU	- (komplex)
OrderType	ebics:OrderTBaseType (→token, length=3, pattern="[A-Z0-9]{3}")	1	Auftragsart des zur VEU eingereichten Auftrags	„IZV“
FileFormat	FileFomatType (complex) (→token)	0..1	Dateiformat des Auftrags Bemerkung: Zu verwenden, wenn OrderType = “FUL” oder “FDL”	
FileFormat» @CountryCode	CountryCodeType (→token, length=2, pattern= " [A-Z]{2,2}")		Information zum Anwendungsbereich des Formats (z.B. länderspezifische Formate)	“FR”…
OrderID	ebics:OrderIDType (→token, fixLength=4)	1	Auftragsnummer des zur VEU eingereichten Auftrags	„OR01“
OrderDataSize	positiveInteger	1	Größe der unkomprimier-ten Auftragsdaten des zur VEU eingereichten Auftrags in Bytes	123456
SigningInfo	ebics:HVUSigning» InfoType (komplex)	1	Informationen zu den Unterschriftsmodalitäten	- (komplex)
SigningInfo» @readyToBeSigned	boolean	1	Ist der Auftrag unter-schriftsreif (true) oder bereits vom Teilnehmer unterschrieben (false)?	„true“

SigningInfo» @NumSigRequired	positiveInteger	1	Anzahl der insgesamt zur Freigabe erforderlichen EUs	4
SigningInfo» @numSigDone	nonNegativeInteger	1	Anzahl der bereits geleisteten EUs	2
SignerInfo	ebics:SignerInfo» Type (komplex)	0..∞	Informationen zu den bisherigen Unterzeichnern	- (komplex)
PartnerID (in SignerInfo)	ebics:PartnerIDType (→token, maxLength=35) pattern="[a-zA-Z0-9,=]{1,35}	1	Kunden-ID des Unterzeichners	„PARTNER1“
UserID (in SignerInfo)	ebics:UserIDType (→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})	1	Teilnehmer-ID des Unterzeichners	„USER0001“
Name (in SignerInfo)	ebics:NameType (→normalizedString)	0..1	Name des Unterzeichners	„Max Mustermann“
Timestamp (in SignerInfo)	ebics:TimestampType (→dateTime)	1	Zeitstempel der Unterzeichnung (d.h. der Übertragung der Unterschrift)	„2005-01-31T» 16:30:45.123Z“
Permission	– (komplex)	1	zusätzliche Berechtigungsinformationen zu dem Teilnehmer, der unterzeichnet hat	- (komplex)
Permission» @Authorisation» Level	ebics:Authorisation» LevelType (→token, length=1: "E", "A", "B", "T")	1	Unterschriftsberechtigung des Teilnehmers, der unterzeichnet hat	„A“
OriginatorInfo	ebics:HVUOriginator» InfoType (komplex)	1	Informationen zu dem Einreicher des Auftrags	- (komplex)
PartnerID (in OriginatorInfo)	ebics:PartnerIDType (→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})	1	Kunden-ID des Einreichers	„PARTNER2“
UserID (in OriginatorInfo)	ebics:UserIDType (→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})	1	Teilnehmer-ID des Einreichers	„USER0002“
Name (in OriginatorInfo)	ebics:NameType (→normalizedString)	0..1	Name des Einreichers	„Erich Einreicher“
Timestamp (in OriginatorInfo)	ebics:TimestampType (→dateTime)	1	Zeitstempel der Einreichung (d.h. der Übertragung der Auftragsdatei)	„2005-01-30T» 15:30:45.123Z“

8.3.1.2.4 Beispiel-XML

```
<?xml version="1.0" encoding="UTF-8"?>
<HVUResponseOrderData
 xmlns="urn:org:ebics:H004"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation=" urn:org:ebics:H004 ebics_orders_H004.xsd ">
 <OrderDetails>
   <OrderType>IZV</OrderType>
   <OrderID>OR01</OrderID>
   <OrderDataSize>123456</OrderDataSize>
   <SigningInfo NumSigRequired="4" readyToBeSigned="true" NumSigDone="2"/>
   <SignerInfo>
     <PartnerID>PARTNER1</PartnerID>
     <UserID>USER0001</UserID>
     <Name>Max Mustermann</Name>
     <Timestamp>2005-01-31T16:30:45.123Z</Timestamp>
     <Permission AuthorisationLevel="A"/>
   </SignerInfo>
   <SignerInfo>
     <PartnerID>PARTNER2</PartnerID>
     <UserID>USER0002</UserID>
     <Name>Maxime Musterfrau</Name>
     <Timestamp>2005-01-31T17:30:45.123Z</Timestamp>
     <Permission AuthorisationLevel="B"/>
   </SignerInfo>
   <OriginatorInfo>
     <PartnerID>PARTNER1</PartnerID>
     <UserID>USER0001</UserID>
     <Name>Erich Einreicher</Name>
     <Timestamp>2005-01-30T15:30:45.123Z</Timestamp>
   </OriginatorInfo>
 </OrderDetails>
</HVUResponseOrderData>
```

8.3.1.3 HVZ-Request

Im HVZ-Request übergibt der Teilnehmer optional eine Liste von Auftragsarten als Filterkriterium. Nur Aufträge, deren Auftragsart in der übergebenen Liste enthalten ist, werden zurückgeliefert. Falls der Teilnehmer keine Auftragsartenliste als Eingrenzung übergibt, bekommt er sämtliche Aufträge aufgelistet, für die er unterschriftsberechtigt ist.

Ausprägung der `OrderParams` (Auftragsparameter) für HVZ: `HVZOrderParams`

8.3.1.3.1 XML-Schema (grafische Darstellung)

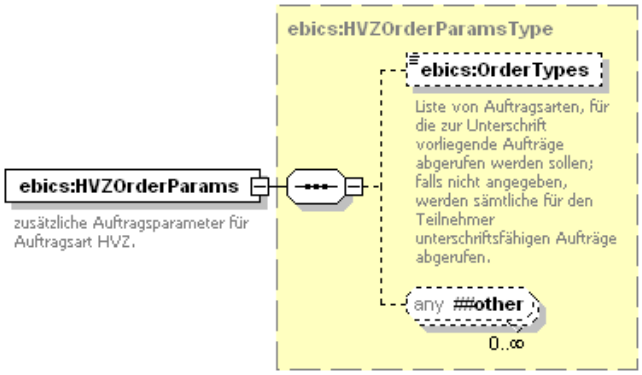


Abbildung: 77 HVZOrderParams

8.3.1.3.2 XML-Schema (textuelle Darstellung)

```
<element name="HVZOrderParams" type="ebics:HVZOrderParamsType"
   substitutionGroup="ebics:OrderParams">
   <annotation>
      <documentation xml:lang="de">
         zusätzliche Auftragsparameter für Auftragsart HVZ.
      </documentation>
   </annotation>
</element>
<complexType name="HVZOrderParamsType">
   <annotation>
      <documentation xml:lang="de">
         Datentyp für zusätzliche Auftragsparameter für Auftragsart HVZ.
      </documentation>
   </annotation>
   <sequence>
      <element name="OrderTypes" type="ebics:OrderTListType"
         minOccurs="0">
         <annotation>
            <documentation xml:lang="de">
               Liste von Auftragsarten, für die zur Unterschrift
               vorliegende Aufträge abgerufen werden sollen; falls nicht
               angegeben, werden sämtliche für den Teilnehmer
               unterschriftsfähigen Aufträge abgerufen.
            </documentation>
         </annotation>
      </element>
      <any namespace="##other" processContents="lax" minOccurs="0"
         maxOccurs="unbounded" />
   </sequence>
</complexType>
```

8.3.1.3.3 Bedeutung der XML-Elemente/-Attribute

XML-Element/ -Attribut	Datentyp	#	Bedeutung	Beispiel
HVZOrderParams	ebics:HVZOrderParamsType (komplex)	1	Auftragsparameter für Auftragsart HVZ	- (komplex)

OrderTypes	ebics:OrderTListType (→list<ebics:OrderTBaseType> →list<token, length=3, pattern="[A-Z0-9]{3}">)	0..1	Liste von Auftragsarten, für die zur Unterschrift vorliegende Aufträge abgerufen werden sollen; falls nicht angegeben, werden sämtliche Aufträge abgerufen, für die der Teilnehmer unterschriftsberechtigt ist	„IZV“

8.3.1.3.4 Beispiel-XML (gekürzt)

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest xmlns="urn:org:ebics:H004"
   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:org:ebics:H004 ebics_request_H004.xsd"
   Version="H004" Revision="1">
   <header authenticate="true">
      <static>
         <!-- […] -->
         <OrderDetails>
            <OrderType>HVZ</OrderType>
            <OrderAttribute>DZHNN</OrderAttribute>
            <HVZOrderParams>
               <OrderTypes>IZV</OrderTypes>
            </HVZOrderParams>
         </OrderDetails>
         <!-- […] -->
      </static>
      <!-- […] -->
   </header>
   <!-- […] -->
</ebicsRequest>
```

8.3.1.4 HVZ-Response

In der HVZ-Response werden dem Teilnehmer Informationen zu den Aufträgen zurückgeliefert, für die er unterschriftsberechtigt ist.

HVZResponseOrderData enthält alle Information aus HVUResponseOrderData und HVDResponseOrderData mit Ausnahme des Elements „DisplayFile“ mit der Dateianzeige. Wie bei HVD wird der Auftragshashwert aus der EU des ersten Unterzeichners des Auftrags extrahiert bzw. ggf. neu berechnet, wenn der Teilnehmer, der HVZ durchführt, ein anderes Signaturverfahren verwendet. Um dies transparent zu gestalten, wird dem Hashwert ein Attribut mit dem verwendeten Signaturverfahren mitgegeben.

Nur für Zahlungsaufträge werden zusätzliche Informationen aus der Dateianzeige geliefert, falls diese verfügbar sind:

- Summe aller Beträge über alle logische Dateien
- Gesamtanzahl der Sätze über alle logischen Dateien
- Währung (nur bei sortenreinen Zahlungen, sonst keine Angabe)
- Für DTAUS/DTAZV: Auftraggeber, Kontonummer / IBAN und Bankleitzahl / BIC der ersten Transaktion der ersten logischen Datei

Ausprägung der (dekodierten & entschlüsselten & dekomprimierten) `OrderData` (Auftragsdaten) für HVZ: `HVZResponseOrderData`

8.3.1.4.1 XML-Schema (grafische Darstellung)

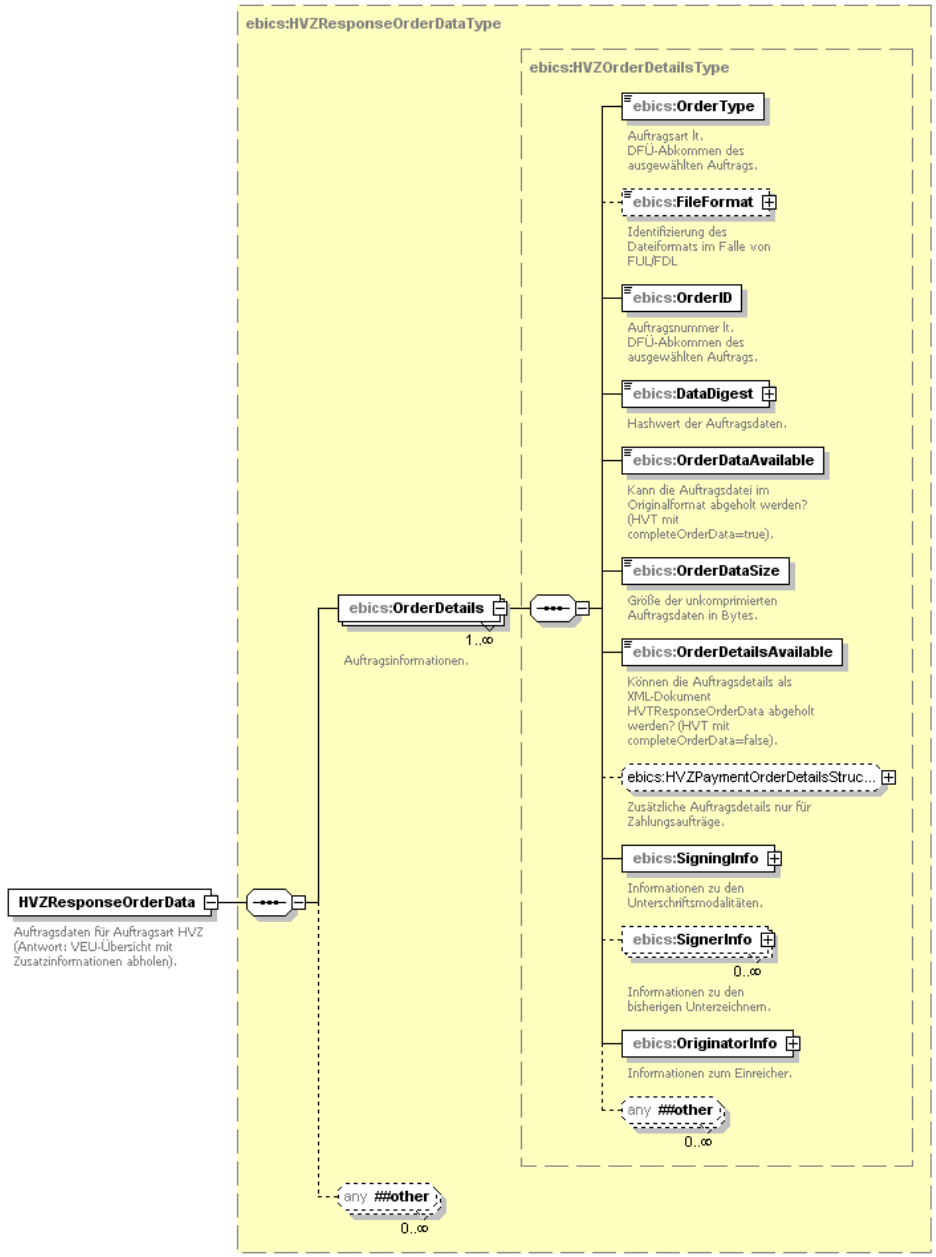


Abbildung 78: HVZResponseOrderData

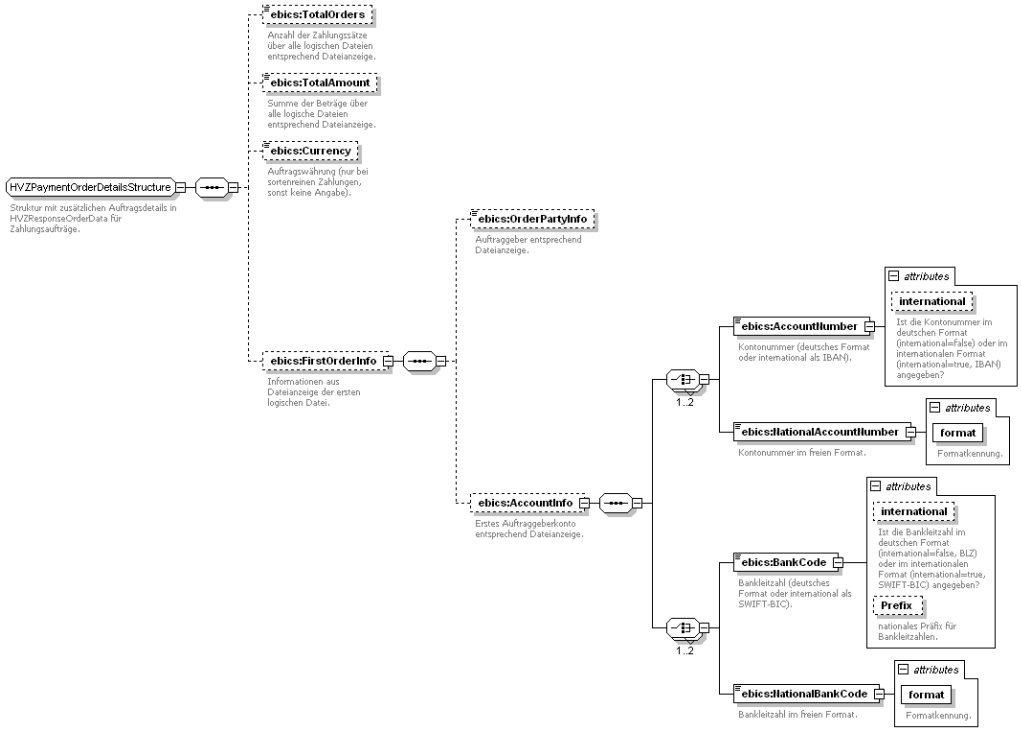


Abbildung 79 HVZPaymentOrderDetailsStructure

8.3.1.4.2 XML-Schema (textuelle Darstellung)

```
<element name="HVZResponseOrderData"
   type="ebics:HVZResponseOrderDataType"
   substitutionGroup="ebics:EBICSOrderData">
   <annotation>
      <documentation xml:lang="de">
         Auftragsdaten für Auftragsart HVZ (Antwort: VEU-Übersicht mit
         Zusatzinformationen abholen).
      </documentation>
   </annotation>
</element>
<complexType name="HVZResponseOrderDataType">
   <annotation>
      <documentation xml:lang="de">
         Datentyp für Auftragsdaten für Auftragsart HVZ (Antwort:
         VEU-Übersicht mit Zusatzinformationen abholen).
      </documentation>
   </annotation>
   <sequence>
      <annotation>
         <documentation xml:lang="de" />
      </annotation>
      <element name="OrderDetails" type="ebics:HVZOrderDetailsType"
         maxOccurs="unbounded">
```

```
         <annotation>
            <documentation xml:lang="de">
               Auftragsinformationen.
            </documentation>
         </annotation>
      </element>
      <any namespace="##other" processContents="lax" minOccurs="0"
         maxOccurs="unbounded" />
   </sequence>
</complexType>
<complexType name="HVZOrderDetailsType">
   <annotation>
      <documentation xml:lang="de">
         Datentyp für HVZ-Auftragsdetails.
      </documentation>
   </annotation>
   <sequence>
      <element name="OrderType" type="ebics:OrderTBaseType">
         <annotation>
            <documentation xml:lang="de">
               Auftragsart lt. DFÜ-Abkommen des ausgewählten Auftrags.
            </documentation>
         </annotation>
      </element>
      <element name="OrderID" type="ebics:OrderIDType">
         <annotation>
            <documentation xml:lang="de">
               Auftragsnummer lt. DFÜ-Abkommen des ausgewählten
               Auftrags.
            </documentation>
         </annotation>
      </element>
      <element name="DataDigest" type="ebics:DigestType">
         <annotation>
            <documentation xml:lang="de">
               Hashwert der Auftragsdaten.
            </documentation>
         </annotation>
      </element>
      <element name="OrderDataAvailable" type="boolean">
         <annotation>
            <documentation xml:lang="de">
               Kann die Auftragsdatei im Originalformat abgeholt werden?
               (HVT mit completeOrderData=true).
            </documentation>
         </annotation>
      </element>
      <element name="OrderDataSize" type="positiveInteger">
         <annotation>
            <documentation xml:lang="de">
               Größe der unkomprimierten Auftragsdaten in Bytes.
            </documentation>
         </annotation>
      </element>
      <element name="OrderDetailsAvailable" type="boolean">
         <annotation>
            <documentation xml:lang="de">
               Können die Auftragsdetails als XML-Dokument
               HVTResponseOrderData abgeholt werden? (HVT mit
               completeOrderData=false).
            </documentation>
         </annotation>
```

```
      </element>
      <group ref="ebics:HVZPaymentOrderDetailsStructure"
         minOccurs="0">
         <annotation>
            <documentation>
               Auftragsdetails Order details related to payment orders
               only.
            </documentation>
         </annotation>
      </group>
      <element name="SigningInfo" type="ebics:HVUSigningInfoType">
         <annotation>
            <documentation xml:lang="de">
               Informationen zu den Unterschriftsmodalitäten.
            </documentation>
         </annotation>
      </element>
      <element name="SignerInfo" type="ebics:SignerInfoType"
         minOccurs="0" maxOccurs="unbounded">
         <annotation>
            <documentation xml:lang="de">
               Informationen zu den bisherigen Unterzeichnern.
            </documentation>
         </annotation>
      </element>
      <element name="OriginatorInfo"
         type="ebics:HVUOriginatorInfoType">
         <annotation>
            <documentation xml:lang="de">
               Informationen zum Einreicher.
            </documentation>
         </annotation>
      </element>
      <any namespace="##other" processContents="lax" minOccurs="0"
         maxOccurs="unbounded" />
   </sequence>
</complexType>
<group name="HVZPaymentOrderDetailsStructure">
   <annotation>
      <documentation xml:lang="de">
         Struktur mit zusätzlichen Auftragsdetails in
         HVZResponseOrderData für Zahlungsaufträge.
      </documentation>
      <documentation xml:lang="en">
         Structure with additional order details in HVZResponseOrderData
         related to payment orders.
      </documentation>
   </annotation>
   <sequence>
      <element name="TotalOrders" type="nonNegativeInteger"
         minOccurs="0">
         <annotation>
            <documentation xml:lang="de">
               Anzahl der Zahlungssätze über alle logischen Dateien
               entsprechend Dateianzeige.
            </documentation>
            <documentation xml:lang="en">
               Total transaction number for all logical files (from
               dispay file).
            </documentation>
         </annotation>
      </element>
```

```
<element name="TotalAmount" minOccurs="0">
   <annotation>
      <documentation xml:lang="de">
         Summe der Beträge über alle logische Dateien entsprechend
         Dateianzeige.
      </documentation>
      <documentation xml:lang="en">
         Total transaction amount for all logical files (from
         dispay file).
      </documentation>
   </annotation>
   <simpleType>
      <restriction base="ebics:AmountValueType" />
   </simpleType>
</element>
<element name="Currency" type="ebics:CurrencyBaseType"
   minOccurs="0">
   <annotation>
      <documentation xml:lang="de">
         Auftragswährung (nur bei sortenreinen Zahlungen, sonst
         keine Angabe).
      </documentation>
      <documentation xml:lang="en">
         Order currency (only if identical across all
         transactions, ship otherwise).
      </documentation>
   </annotation>
</element>
<element name="FirstOrderInfo" minOccurs="0">
   <annotation>
      <documentation xml:lang="de">
         Informationen aus Dateianzeige der ersten logischen
         Datei.
      </documentation>
      <documentation xml:lang="en">
         Order details from display file for first logical file.
      </documentation>
   </annotation>
   <complexType>
      <sequence>
         <element name="OrderPartyInfo" type="normalizedString"
            minOccurs="0">
            <annotation>
               <documentation xml:lang="de">
                  Auftraggeber entsprechend Dateianzeige.
               </documentation>
               <documentation xml:lang="en">
                  Order party information (from display file).
               </documentation>
            </annotation>
         </element>
         <element name="AccountInfo" minOccurs="0">
            <annotation>
               <documentation xml:lang="de">
                  Erstes Auftraggeberkonto entsprechend
                  Dateianzeige.
               </documentation>
               <documentation xml:lang="en">
                  First order party account (from display file).
               </documentation>
            </annotation>
            <complexType>
```

```
<sequence>
   <choice maxOccurs="2">
      <element name="AccountNumber">
         <annotation>
            <documentation xml:lang="de">
               Kontonummer (deutsches Format oder
               international als IBAN).
            </documentation>
            <documentation xml:lang="en">
               Account number (German format or
               international as IBAN).
            </documentation>
         </annotation>
         <complexType>
            <simpleContent>
               <extension
                  base="Q1:AccountNumberType">
                  <attribute name="international"
                     type="boolean" use="optional"
                     default="false">
                     <annotation>
                        <documentation
                           xml:lang="de">
                           Ist die Kontonummer im
                           deutschen Format
                           (international=false)
                           oder im internationalen
                           Format
                           (international=true,
                           IBAN) angegeben?
                        </documentation>
                        <documentation
                           xml:lang="en">
                           Account number given in
                           German format
                           (international=false) or
                           in international format
                           (international=true,
                           IBAN)?
                        </documentation>
                     </annotation>
                  </attribute>
               </extension>
            </simpleContent>
         </complexType>
      </element>
      <element name="NationalAccountNumber">
         <annotation>
            <documentation xml:lang="de">
               Kontonummer im freien Format.
            </documentation>
            <documentation xml:lang="en">
               Account number in free format.
            </documentation>
         </annotation>
         <complexType>
            <simpleContent>
               <extension
                  base="Q1:NationalAccountNumberType">
                  <attribute name="format"
                     type="token" use="required">
                     <annotation>
```

```
                        <documentation
                           xml:lang="de">
                           Formatkennung.
                        </documentation>
                        <documentation
                           xml:lang="en">
                           Format type.
                        </documentation>
                     </annotation>
                  </attribute>
               </extension>
            </simpleContent>
         </complexType>
      </element>
   </choice>
   <choice maxOccurs="2">
      <element name="BankCode">
         <annotation>
            <documentation xml:lang="de">
               Bankleitzahl (deutsches Format oder
               international als SWIFT-BIC).
            </documentation>
            <documentation xml:lang="en">
               Bank sort code (German format or
               international as SWIFT-BIC).
            </documentation>
         </annotation>
         <complexType>
            <simpleContent>
               <extension base="Q1:BankCodeType">
                  <attribute name="international"
                     type="boolean" use="optional"
                     default="false">
                     <annotation>
                        <documentation
                           xml:lang="de">
                           Ist die Bankleitzahl im
                           deutschen Format
                           (international=false,
                           BLZ) oder im
                           internationalen Format
                           (international=true,
                           SWIFT-BIC) angegeben?
                        </documentation>
                        <documentation
                           xml:lang="en">
                           Bank sort code given in
                           German format
                           (international=false) or
                           in international format
                           (international=true,
                           SWIFT-BIC)?
                        </documentation>
                     </annotation>
                  </attribute>
                  <attribute name="Prefix"
                     type="ebics:BankCodePrefixType"
                     use="optional">
                     <annotation>
                        <documentation
                           xml:lang="de">
                           nationales Präfix für
```

```
                                    Bankleitzahlen.
                                 </documentation>
                                 <documentation
                                    xml:lang="en">
                                    National prefix for bank
                                    sort code.
                                 </documentation>
                              </annotation>
                           </attribute>
                        </extension>
                     </simpleContent>
                  </complexType>
               </element>
               <element name="NationalBankCode">
                  <annotation>
                     <documentation xml:lang="de">
                        Bankleitzahl im freien Format.
                     </documentation>
                     <documentation xml:lang="en">
                        Bank sort code in free format.
                     </documentation>
                  </annotation>
                  <complexType>
                     <simpleContent>
                        <extension
                           base="Q1:NationalBankCodeType">
                           <attribute name="format"
                              type="token" use="required">
                              <annotation>
                                 <documentation
                                    xml:lang="de">
                                    Formatkennung.
                                 </documentation>
                                 <documentation
                                    xml:lang="en">
                                    Format type.
                                 </documentation>
                              </annotation>
                           </attribute>
                        </extension>
                     </simpleContent>
                  </complexType>
               </element>
            </choice>
         </sequence>
      </complexType>
   </element>
</sequence>
</complexType>
</element>
</sequence>
</group>
```

8.3.1.4.3 Bedeutung der XML-Elemente/-Attribute

XML-Element/-Attribut	Datentyp	#	Bedeutung	Beispiel
HVZResponse» OrderData	ebics:HVZResponse» OrderDataType	1	XML-Auftragsdaten für Auftragsart HVZ	- (komplex)

	(komplex)			
OrderDetails	ebics:HVZOrder» DetailsType (komplex)	1..∞	Auftragsinformationen für Auftragsart HVZ	- (komplex)
OrderType	ebics:OrderTBaseType (→token, length=3, pattern="[A-Z0-9]{3}")	1	Auftragsart des zur VEU eingereichten Auftrags	„IZV“
FileFormat	FileFomatType (complex) (→token)	0..1	Dateiformat des Auftrags Bemerkung: Zu verwenden, wenn OrderType = “FUL” oder “FDL”	
FileFormat» @CountryCode	CountryCodeType (→token, length=2, pattern= " [A-Z]{2,2}")		Information zum Anwendungsbereich des Formats (z.B. länderspezifische Formate)	“FR”…
OrderID	ebics:OrderIDType (→token, fixLength=4)	1	Auftragsnummer des zur VEU eingereichten Auftrags	„OR01“
DataDigest	ebics:DigestType (→dsig:DigestValue» Type →base64Binary)	1	Hashwert des Auftrags für das Signaturverfahren des Teilnehmers gemäß Request-Element UserID	- (Base64-Daten)
DataDigest» @SignatureVersion	ebics:Signature»VersionType (→token, length=4, pattern="A\d{3}"		Version des Signaturverfahrens des Teilnehmers gemäß Request-Element UserID	z.B. „A005“
OrderDataAvailable	boolean	1	Kann die Auftragsdatei im Originalformat abgeholt werden? (HVT mit completeOrderData=true)	„true“
OrderDataSize	positiveInteger	1	Größe der unkomprimierten Auftragsdaten des zur VEU eingereichten Auftrags in Bytes	123456
OrderDetails» Available	boolean	1	Können die Auftragsdetails als XML-Dokument HVTResponseOrderData abgeholt werden? (HVT mit completeOrderData=false)	„true“
TotalOrders	nonNegativeInteger	0..1	Anzahl der Zahlungssätze über alle logische Dateien	15

			entsprechend Dateianzeige.	
TotalAmount	ebics:AmountValue» Type (→decimal, totalDigits=24, fractionDigits=4)	0..1	Summe der Beträge über alle logische Dateien entsprechend Dateianzeige.	129.00
TotalAmount» @isCredit	boolean	0..1	Kennzeichen zur Unterscheidung von Überweisungen (isCredit="true") und Lastschriften (isCredit="false"). (optional verwendbar, wenn dies über alle Transaktionen identisch ist; andernfalls auszulassen).	"false"
Currency	ebics:CurrencyBase» Type (→token, length=3, pattern="[A-Z]{3}")	0..1	Auftragswährung (nur bei sortenreinen Zahlungen, sonst keine Angabe).	„USD"
FirstOrderInfo	(komplex)	0..1	Informationen aus Dateianzeige der ersten logischen Datei.	- (komplex)
OrderPartyInfo	normalizedString	0..1	Auftraggeber entsprechend Dateianzeige.	„Arnod Auftraggeber"
AccountInfo	komplex	0..1		- (komplex)
-	-	1..2	Angaben zur Kontonummer: AccountNumber und/oder NationalAccountNumber	-
AccountNumber	ebics:AccountNumber» Type (→token, maxLength=40, pattern="\d{3,10}|([A-Z]{2}\d{2}[A-Za-z0-9]{3,30}")	1	Kontonummer (deutsches Format oder international als IBAN)	„12345678"
AccountNumber» @international	boolean	0..1	Ist die Kontonummer im deutschen Format (international=false) oder im internationalen Format (international=true, IBAN) angegeben? Default="false"	„false"

NationalAccount» Number	ebics:National» AccountNumberType (→token, maxLength=40)	1	Kontonummer im freien Format (weder deutsches Format noch IBAN)	„1234567890123456“
NationalAccount» Number@format	token	1	Formatkennung	„other“
-	-	1..2	Angaben zur Bankleitzahl: BankCode und/oder NationalBankCode	-
BankCode	ebics:BankCodeType (→token, maxLength=11, pattern="\d{8}|([A-Z]{6}[A-Z0-9]{2}([A-Z0-9]{3})?)")	1	Bankleitzahl (deutsches Format oder international als SWIFT-BIC).	„50010060“
BankCode» @international	boolean	0..1	Ist die Bankleitzahl im nationalen=deutschen (BankCode» @international= "false") oder internationalen=SWIFT-BIC-Format (BankCode» @international="true") angegeben? Default="false"	„false“
NationalBankCode	ebics:National» BankCodeType (→token, maxLength=30)	1	Bankleitzahl im freien Format. (weder deutsches Format noch SWIFT-BIC)	„123456789012“
NationalBankCode» @format	token	1	Formatkennung	„other“
SigningInfo	ebics:HVUSigning» InfoType (komplex)	1	Informationen zu den Unterschriftsmodalitäten	- (komplex)
SigningInfo» @readyToBeSigned	Boolean	1	Ist der Auftrag unterschriftsreif (true) oder bereits vom Teilnehmer unterschrieben (false)?	„true“
SigningInfo» @NumSigRequired	positiveInteger	1	Anzahl der insgesamt zur Freigabe erforderlichen EUs	4
SigningInfo» @numSigDone	nonNegativeInteger	1	Anzahl der bereits geleisteten EUs	2
SignerInfo	ebics:SignerInfo» Type (komplex)	0..∞	Informationen zu den bisherigen Unterzeichnern	- (komplex)
PartnerID (in SignerInfo)	ebics:PartnerIDType (→token, maxLength=35, pattern="[a-zA-Z0-	1	Kunden-ID des Unterzeichners	„PARTNER1“

	9,=]{1,35})			
UserID (in SignerInfo)	ebics:UserIDType (→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})	1	Teilnehmer-ID des Unterzeichners	„USER0001“
Name (in SignerInfo)	ebics:NameType (→normalizedString)	0..1	Name des Unterzeichners	„Max Mustermann“
Timestamp (in SignerInfo)	ebics:TimestampType (→dateTime)	1	Zeitstempel der Unterzeichnung (d.h. der Übertragung der Unterschrift)	„2005-01-31T» 16:30:45.123Z“
Permission	– (komplex)	1	zusätzliche Berechtigungsinformationen zu dem Teilnehmer, der unterzeichnet hat	- (komplex)
Permission» @Authorisation» Level	ebics:Authorisation» LevelType (→token, length=1: "E", "A", "B", "T")	1	Unterschriftsberechtigung des Teilnehmers, der unterzeichnet hat	„A“
OriginatorInfo	ebics:HVUOriginator» InfoType (komplex)	1	Informationen zu dem Einreicher des Auftrags	- (komplex)
PartnerID (in OriginatorInfo)	ebics:PartnerIDType (→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})	1	Kunden-ID des Einreichers	„PARTNER2“
UserID (in OriginatorInfo)	ebics:UserIDType (→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})	1	Teilnehmer-ID des Einreichers	„USER0002“
Name (in OriginatorInfo)	ebics:NameType (→normalizedString)	0..1	Name des Einreichers	„Erich Einreicher“
Timestamp (in OriginatorInfo)	ebics:TimestampType (→dateTime)	1	Zeitstempel der Einreichung (d.h. der Übertragung der Auftragsdatei)	„2005-01-30T» 15:30:45.123Z“

8.3.1.4.4 Beispiel-XML

```
<?xml version="1.0" encoding="UTF-8"?>
<HVZResponseOrderData xmlns="urn:org:ebics:H004"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:org:ebics:H004 ebics_orders_H004.xsd">
   <OrderDetails>
      <OrderType>IZV</OrderType>
      <OrderID>OR01</OrderID>
      <DataDigest SignatureVersion="A004">9H/rQr2Axe9hYTV2n/tCp+3UIQQ=</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>2005-01-31T16:30:45.123Z</Timestamp>
         <Permission AuthorisationLevel="A" />
      </SignerInfo>
      <SignerInfo>
         <PartnerID>PARTNER2</PartnerID>
         <UserID>USER0002</UserID>
         <Name>Maxime Musterfrau</Name>
         <Timestamp>2005-01-31T17:30:45.123Z</Timestamp>
         <Permission AuthorisationLevel="B" />
      </SignerInfo>
      <OriginatorInfo>
         <PartnerID>PARTNER1</PartnerID>
         <UserID>USER0001</UserID>
         <Name>Erich Einreicher</Name>
         <Timestamp>2005-01-30T15:30:45.123Z</Timestamp>
      </OriginatorInfo>
   </OrderDetails>
</HVZResponseOrderData>
```

8.3.2 HVD (VEU-Status abrufen)

Mit HVD kann ein Teilnehmer den Status eines Auftrags abrufen, der sich aktuell in der VEU-Prozessverarbeitung befindet und für den der Teilnehmer unterschriftsberechtigt ist. Er bekommt Informationen über den Auftrag in Form eines elektronischen Begleitzettels (`DisplayFile`) und des Auftrags-Hashwerts (`DataDigest`) sowie über die bisherigen Unterzeichner (`SignerInfo`). Den Auftragshashwert hat das Banksystem aus der EU des ersten Unterzeichners des Auftrags extrahiert oder es wird in dem Fall, dass der Teilnehmer, der HVD durchführt, ein anderes Signaturverfahren verwendet, ggf. neu berechnet. Die Daten des Begleitzettels MÜSSEN inhaltlich mit den Auftragsdaten, deren Hashwert mitgeliefert wird, übereinstimmen.
Das Banksystem muss prüfen, ob der Teilnehmer eine bankfachliche Unterschriftsberechtigung (Unterschriftsklasse E, A oder B) für den konkreten Auftrag besitzt und der Auftrag noch in der Unterschriftsmappe ist. Falls die Berechtigung fehlt, muss die Transaktion mit dem Fehlercode EBICS_DISTRIBUTED_SIGNATURE_AUTHORISATION_FAILED abgebrochen werden.

- Bei einigen zugrunde liegenden Auftragsarten / Geschäftsvorfällen ist ein Abruf von Detailinformationen zu einem bestimmten Auftrag in der VEU-Prozessverarbeitung mittels der Transaktion HVT nicht möglich. Ob das für den aktuellen Auftrag möglich ist oder nicht, wird vom Banksystem in der HVD-Response signalisiert.

- Das Banksystem prüft vor der Ausführung von HVD, dass sich der Auftrag aktuell in der VEU-Prozessverarbeitung befindet und beendet im Fehlerfall die Transaktion mit dem fachlichen Fehlercode EBICS_ORDERID_UNKNOWN.

HVD ist eine Auftragsart vom Typ „Download“.

8.3.2.1 HVD-Request

Im HVD-Request übergibt der Teilnehmer die relevanten Daten zur Identifikation des Auftrags, für den er den VEU-Status abrufen will.

Ausprägung der `OrderParams` (Auftragsparameter) für HVD: `HVDOrderParams`

8.3.2.1.1 XML-Schema (grafische Darstellung)

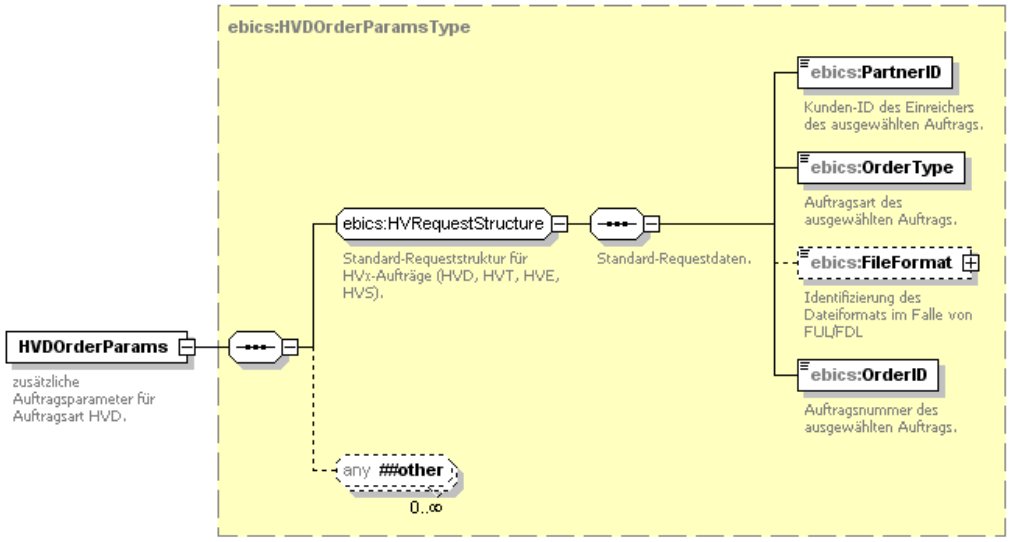


Abbildung 80: HVDOrderParams

8.3.2.1.2 XML-Schema (textuelle Darstellung)

```
 <element name="HVDOrderParams" type="ebics:HVDOrderParamsType"
substitutionGroup="ebics:OrderParams">
   <annotation>
     <documentation xml:lang="de">zusätzliche Auftragsparameter für Auftragsart
HVD.</documentation>
   </annotation>
 </element>
 <complexType name="HVDOrderParamsType">
   <annotation>
     <documentation xml:lang="de">Datentyp für zusätzliche Auftragsparameter für Auftragsart
HVD.</documentation>
   </annotation>
   <sequence>
     <group ref="ebics:HVRequestStructure"/>
     <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
 </complexType>
 <group name="HVRequestStructure">
   <annotation>
     <documentation xml:lang="de">Standard-Requeststruktur für HVx-Aufträge (HVD, HVT, HVE,
HVS).</documentation>
   </annotation>
   <sequence>
     <annotation>
       <documentation xml:lang="de">Standard-Requestdaten.</documentation>
     </annotation>
     <element name="PartnerID" type="ebics:PartnerIDType">
       <annotation>
         <documentation xml:lang="de">Kunden-ID des Einreichers des ausgewählten
Auftrags.</documentation>
       </annotation>
     </element>
     <element name="OrderType" type="ebics:OrderTBaseType">
       <annotation>
         <documentation xml:lang="de">Auftragsart des ausgewählten Auftrags gemäß Anhang
(Kapitel 13).</documentation>
       </annotation>
     </element>
     <element name="OrderID" type="ebics:OrderIDType">
       <annotation>
         <documentation xml:lang="de">Auftragsnummer des ausgewählten Auftrags gemäß Kapitel
10.1.</documentation>
       </annotation>
     </element>
   </sequence>
 </group>
```

8.3.2.1.3 Bedeutung der XML-Elemente/-Attribute

XML-Element/ -Attribut	Datentyp	#	Bedeutung	Beispiel
`HVDOrderParams`	`ebics:HVDOrderParamsType` (komplex)	1	Auftragsparameter für Auftragsart HVD	- (komplex)
`PartnerID`	`ebics:PartnerIDType`	1	Kunden-ID des Einreichers	„PARTNER1“

	(→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})			
OrderType	ebics:OrderTBaseType (→token, length=3, pattern="[A-Z0-9]{3}")	1	Auftragsart des zur VEU eingereichten Auftrags	„IZV“
FileFormat	FileFomatType (complex) (→token)	0..1	Dateiformat des Auftrags Bemerkung: Zu verwenden, wenn OrderType = “FUL” oder “FDL”	
FileFormat» @CountryCode	CountryCodeType (→token, length=2, pattern= " [A-Z]{2,2}")		Information zum Anwendungsbereich des Formats (z.B. länderspezifische Formate)	“FR”…
OrderID	ebics:OrderIDType (→token, fixLength=4)	1	Auftragsnummer des zur VEU eingereichten Auftrags	„OR01“

8.3.2.1.4 Beispiel-XML (gekürzt)

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
 xmlns="urn:org:ebics:H004"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_request_H004.xsd"
 Version="H004" Revision="1">
  <header authenticate="true">
    <static>
      <!-- […] -->
      <OrderDetails>
        <OrderType>HVD</OrderType>
        <OrderAttribute>DZHNN</OrderAttribute>
        <HVDOrderParams>
          <PartnerID>PARTNER1</PartnerID>
          <OrderType>IZV</OrderType>
          <OrderID>OR01</OrderID>
        </HVDOrderParams>
      </OrderDetails>
      <!-- […] -->
    </static>
    <!-- […] -->
  </header>
  <!-- […] -->
</ebicsRequest>
```

8.3.2.2 HVD-Response

Die HVD-Response enthält VEU-Informationen zu dem Auftrag, den der Teilnehmer im HVD-Request angefragt hat. Insbesondere wird der Hashwert der Auftragsdaten aus der EU des ersten Unterzeichners samt Begleitzettel zurückgeliefert. Zusätzlich wird angegeben, ob das

Banksystem die Transaktion HVT für den konkreten Auftrag unterstützt. Dabei wird unterschieden zwischen

- `OrderDataAvailable` : Abholung der kompletten Auftragsdatei mit HVT und completeOrderData=true möglich?

- `OrderDetailsAvailable` : Abholung der aufbereiteten Auftragsdetails im XML-Format mit HVT und `completeOrderData=false` möglich?

Die HVD-Response stellt dem Teilnehmer sämtliche Daten zur Verfügung, die dieser zur Bestätigung des Auftrags mittels HVE oder Stornierung mittels HVS benötigt.

Ausprägung der (dekodierten & entschlüsselten & dekomprimierten) `OrderData` (Auftragsdaten) für HVD: `HVDResponseOrderData`

8.3.2.2.1 XML-Schema (grafische Darstellung)

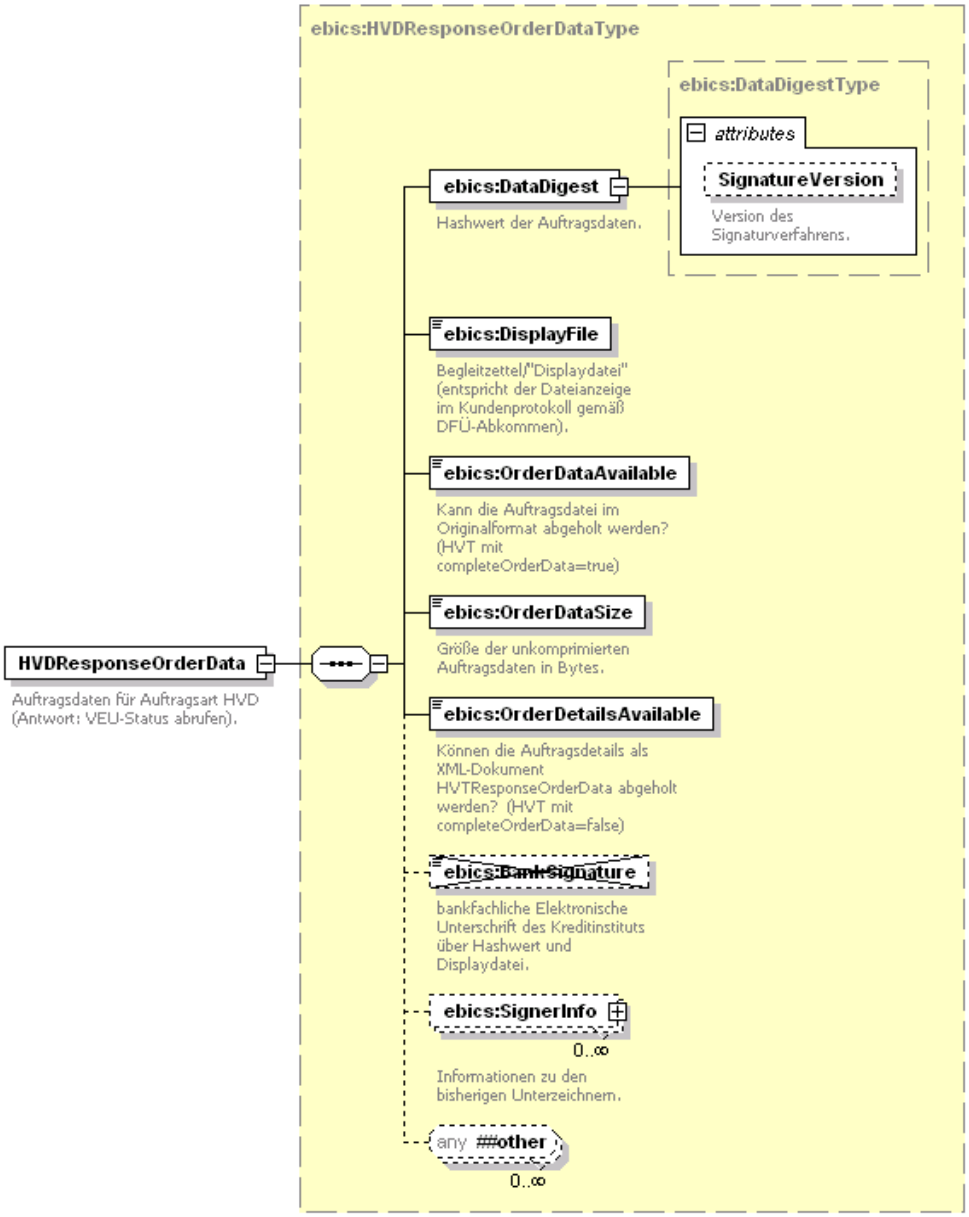


Abbildung 81: HVDResponseOrderData

8.3.2.2.2 XML-Schema (textuelle Darstellung)

```
 <element name="HVDResponseOrderData" type="ebics:HVDResponseOrderDataType"
substitutionGroup="ebics:EBICSOrderData">
   <annotation>
     <documentation xml:lang="de">Auftragsdaten für Auftragsart HVD (Antwort: VEU-Status
abrufen).</documentation>
   </annotation>
 </element>
 <complexType name="HVDResponseOrderDataType">
   <annotation>
     <documentation xml:lang="de">Datentyp für Auftragsdaten für Auftragsart HVD (Antwort:
VEU-Status abrufen).</documentation>
   </annotation>
   <sequence>
     <element name="DataDigest" type="ebics:DigestType">
       <annotation>
         <documentation xml:lang="de">Hashwert der Auftragsdaten.</documentation>
       </annotation>
     </element>
     <element name="DisplayFile" type="base64Binary">
       <annotation>
         <documentation xml:lang="de">Begleitzettel/"Displaydatei" (entspricht der
Dateianzeige im Kundenprotokoll, siehe Fehler! Verweisquelle konnte nicht gefunden
werden.).</documentation>
       </annotation>
     </element>
     <element name="OrderDataAvailable" type="boolean">
       <annotation>
         <documentation xml:lang="de">Kann die Auftragsdatei im Originalformat abgeholt
werden? (HVT mit completeOrderData=true)</documentation>
       </annotation>
     </element>
     <element name="OrderDataSize" type="positiveInteger">
       <annotation>
         <documentation xml:lang="de">Größe der unkomprimierten Auftragsdaten in
Bytes.</documentation>
       </annotation>
     </element>
     <element name="OrderDetailsAvailable" type="boolean">
       <annotation>
         <documentation xml:lang="de">Können die Auftragsdetails als XML-Dokument
HVTResponseOrderData abgeholt werden?  (HVT mit completeOrderData=false)</documentation>
       </annotation>
     </element>
     <element name="BankSignature" type="ebics:SignatureType" minOccurs="0" maxOccurs="0">
       <annotation>
         <documentation xml:lang="de">Elektronische Unterschrift des Kreditinstituts über
Hashwert und Displaydatei.</documentation>
       </annotation>
     </element>
     <element name="SignerInfo" type="ebics:SignerInfoType" minOccurs="0"
maxOccurs="unbounded">
       <annotation>
         <documentation xml:lang="de">Informationen zu den bisherigen
Unterzeichnern.</documentation>
       </annotation>
     </element>
     <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
 </complexType>
```

8.3.2.2.3 Bedeutung der XML-Elemente/-Attribute

XML-Element/-Attribut	Datentyp	#	Bedeutung	Beispiel
HVDResponseOrderData	ebics:HVDResponse»OrderDataType (komplex)	1	XML-Auftragsdaten für Auftragsart HVD	- (komplex)
DataDigest	ebics:DigestType (→dsig:DigestValue»Type →base64Binary)	1	Hashwert des Auftrags für das Signaturverfahren des Teilnehmers gemäß Request-Element UserID	- (Base64-Daten)
DataDigest»@SignatureVersion	ebics:Signature»VersionType (→token, length=4, pattern="A\d{3}"		Version des Signaturverfahrens des Teilnehmers gemäß Request-Element UserID	z.B. „A006“
DisplayFile	base64Binary	1	Begleitzettel/ „Display-Datei“ zum eingereichten Auftrag	- (Base64-Daten)
OrderDataAvailable	boolean	1	Kann die Auftragsdatei im Originalformat abgeholt werden? (HVT mit completeOrderData=true)	true
OrderDataSize	positiveInteger	1	Größe der unkomprimierten Auftragsdaten in Bytes.	1280
OrderDetailsAvailable	boolean	1	Können die Auftragsdetails als XML-Dokument HVTResponseOrderData abgeholt werden? (HVT mit completeOrderData=false)	true
BankSignature	ebics:SignatureType (→base64Binary)	0	EU des Kreditinstituts über Hashwert und Begleitzettel; vorgesehenes Feature	- (Base64-Daten)
SignerInfo	ebics:SignerInfo»Type (komplex)	0..∞	Informationen zu den bisherigen Unterzeichnern	- (komplex)

Für die restlichen XML-Elemente und -Attribute: Siehe Auftragsart HVU (Kapitel 8.3.1.2).

8.3.2.2.4 Beispiel-XML

```
<?xml version="1.0" encoding="UTF-8"?>
<HVDResponseOrderData
 xmlns="urn:org:ebics:H004"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_orders_H004.xsd">
```

```
<DataDigest SignatureVersion="A004">9H/rQr2Axe9hYTV2n/tCp+3UIQQ=</DataDigest>
<DisplayFile>…</DisplayFile>
<OrderDataAvailable>true</OrderDataAvailable>
<OrderDataSize>1280</OrderDataSize>
<OrderDetailsAvailable>true</OrderDetailsAvailable>
<SignerInfo>
  <PartnerID>PARTNER1</PartnerID>
  <UserID>USER0001</UserID>
  <Name>Max Mustermann</Name>
  <Timestamp>2005-01-31T16:30:45.123Z</Timestamp>
  <Permission AuthorisationLevel="A"/>
</SignerInfo>
<SignerInfo>
  <PartnerID>PARTNER2</PartnerID>
  <UserID>USER0002</UserID>
  <Name>Maxime Musterfrau</Name>
  <Timestamp>2005-01-31T17:30:45.123Z</Timestamp>
  <Permission AuthorisationLevel="B"/>
</SignerInfo>
</HVDResponseOrderData>
```

8.3.3 HVT (VEU-Transaktionsdetails abrufen)

HVT liefert dem Teilnehmer detaillierte Informationen über einen Auftrag aus der VEU-Prozessverarbeitung zurück, für den der Teilnehmer unterschriftsberechtigt ist. Je nach Request (`OrderFlags@completeOrderData`) bekommt er entweder die komplette Auftragsdatei oder Details zu Konten, Ausführungstermin, Beträgen und weitere Beschreibungen (`OrderInfo`).

Weitere Filterkriterien (z.B. zur Auswahl von Einzelaufträgen innerhalb eines Gesamtauftrags) kann der Teilnehmer beim Request in der generischen Schlüssel-Wert-Struktur (`Parameter`) übermitteln.

Bei einigen zugrunde liegenden Auftragsarten / Geschäftsvorfällen ist ein Abruf von Detailinformationen mittels `OrderFlags@completeOrderData="false"` nicht möglich. In diesem Fall wird vom Banksystem der fachliche Fehlercode EBICS_UNSUPPORTED_REQUEST_FOR_ORDER_INSTANCE zurückgeliefert. Das Banksystem signalisiert in der HVD-Response mit `OrderDataAvailable` und `OrderDetailsAvailable`, ob für einen bestimmten Auftrag in der VEU-Verwaltung die Durchführung der HVT Transaktion möglich ist.

Das Banksystem prüft vor der Ausführung von HVT, dass sich der Auftrag aktuell in der VEU-Prozessverarbeitung befindet und beendet im Fehlerfall die Transaktion mit dem fachlichen Fehlercode EBICS_ORDERID_UNKNOWN.

Das Banksystem muss prüfen, ob der Teilnehmer eine bankfachliche Unterschriftsberechtigung (Unterschriftsklasse E, A oder B) für den konkreten Auftrag besitzt und der Auftrag noch in der Unterschriftsmappe ist. Falls die Berechtigung fehlt, muss die

Transaktion mit dem Fehlercode
EBICS_DISTRIBUTED_SIGNATURE_AUTHORISATION_FAILED abgebrochen werden.

HVT ist eine Auftragsart vom Typ „Download“.

8.3.3.1 HVT-Request

Im HVT-Request spezifiziert der Teilnehmer den Auftrag, für den er VEU-Transaktionsdetails abrufen will. Außerdem entscheidet er durch Setzen des OrderFlags `completeOrderData`, ob er als Rückmeldung Auftragsdetails (`completeOrderData="false"`) oder die komplette Auftragsdatei (`completeOrderData="true"`) wünscht.

Das Kundensystem kann bei `completeOrderData="false"` die Anzahl der vom Banksystem zu liefernden Auftragsdetails begrenzen. Mit dem Attribut `fetchLimit` am Element `OrderFlags` kann die maximale Anzahl der zu liefernden Auftragsdetails angegeben werden (Ein Vorschlag ist `fetchLimit` = 100). Die Angabe `fetchLimit`=0 bedeutet, dass alle Auftragsdetails des Auftrags angefordert werden.
Mit dem Attribut `fetchOffset` kann das Kundensystem festlegen, ab welcher Offset-Position in der Originalauftragsdatei die Auftragsdetails zurückgeliefert werden.
`fetchOffset` = 0 bedeutet, dass die Auftragsdetails vom Anfang der Auftragsdatei angefordert werden.
Wenn `fetchOffset` größer als die Gesamtanzahl der Auftragsdetails ist, wird die fachliche Fehler EBICS_INVALID_ORDER_PARAMS geliefert.

Für weitere Filterkriterien steht die generische Schlüssel-Wert-Struktur (`Parameter`) bereit.

Ausprägung der `OrderParams` (Auftragsparameter) für HVT: `HVTOrderParams`

8.3.3.1.1 XML-Schema (grafische Darstellung)

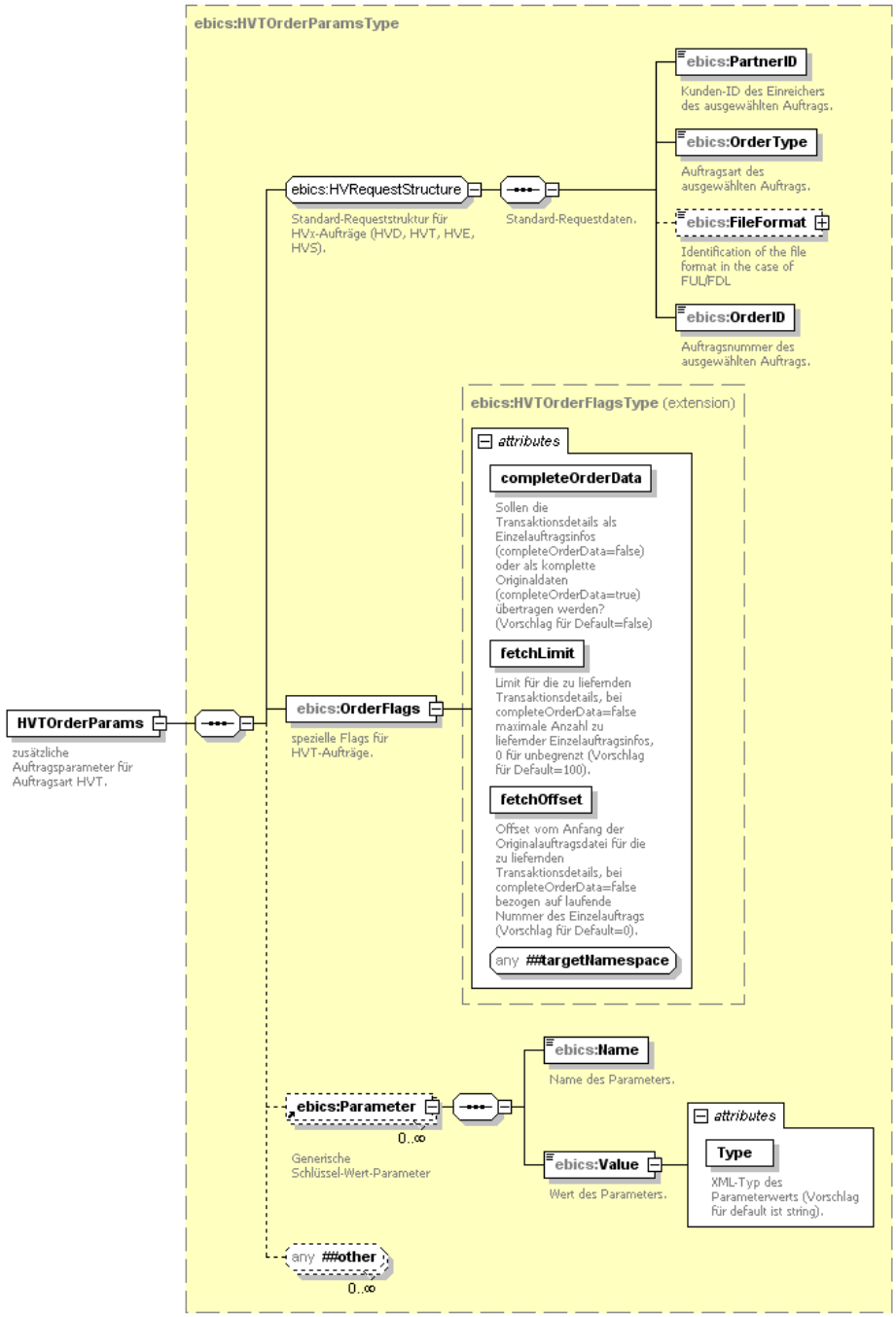


Abbildung 82: HVTOrderParams

8.3.3.1.2 XML-Schema (textuelle Darstellung)

```
<element name="HVTOrderParams" type="ebics:HVTOrderParamsType"
   substitutionGroup="ebics:OrderParams">
   <annotation>
      <documentation xml:lang="de">
         zusätzliche Auftragsparameter für Auftragsart HVT.
      </documentation>
   </annotation>
</element>
<complexType name="HVTOrderParamsType">
   <annotation>
      <documentation xml:lang="de">
         Datentyp für zusätzliche Auftragsparameter für Auftragsart HVT.
      </documentation>
   </annotation>
   <sequence>
      <group ref="ebics:HVRequestStructure" />
      <element name="OrderFlags" type="ebics:HVTOrderFlagsType">
         <annotation>
            <documentation xml:lang="de">
               spezielle Flags für HVT-Aufträge.
            </documentation>
         </annotation>
      </element>
      <element ref="ebics:Parameter" minOccurs="0"
         maxOccurs="unbounded" />
      <any namespace="##other" processContents="lax" minOccurs="0"
         maxOccurs="unbounded" />
   </sequence>
</complexType>
<group name="HVRequestStructure">
   <annotation>
      <documentation xml:lang="de">
         Standard-Requeststruktur für HVx-Aufträge (HVD, HVT, HVE, HVS).
      </documentation>
   </annotation>
   <sequence>
      <annotation>
         <documentation xml:lang="de">
            Standard-Requestdaten.
         </documentation>
      </annotation>
      <element name="PartnerID" type="ebics:PartnerIDType">
         <annotation>
            <documentation xml:lang="de">
               Kunden-ID des Einreichers des ausgewählten Auftrags.
            </documentation>
         </annotation>
      </element>
      <element name="OrderType" type="ebics:OrderTBaseType">
         <annotation>
            <documentation xml:lang="de">
               Auftragsart des ausgewählten Auftrags gemäß Anhang
               (Kapitel 14).
            </documentation>
         </annotation>
      </element>
      <element name="OrderID" type="ebics:OrderIDType">
         <annotation>
            <documentation xml:lang="de">
```

```
               Auftragsnummer des ausgewählten Auftrags gemäß Kapitel
               10.1.
            </documentation>
         </annotation>
      </element>
   </sequence>
</group>
<complexType name="HVTOrderFlagsType">
   <annotation>
      <documentation xml:lang="de">
         Datentyp für HVT-Auftragsnummern.
      </documentation>
   </annotation>
   <simpleContent>
      <extension base="ebics:OrderIDType">
         <attribute name="completeOrderData" type="boolean"
            use="required">
            <annotation>
               <documentation xml:lang="de">
                  Sollen die Transaktionsdetails als Einzelauftragsinfos
                  (completeOrderData=false) oder als komplette
                  Originaldaten (completeOrderData=true) übertragen
                  werden?
               </documentation>
            </annotation>
         </attribute>
         <attribute name="fetchLimit" use="required">
            <annotation>
               <documentation xml:lang="de">
                  Limit für die zu liefernden Transaktionsdetails, bei
                  completeOrderData=false maximale Anzahl zu liefernder
                  Einzelauftragsinfos, 0 für unbegrenzt.
               </documentation>
            </annotation>
            <simpleType>
               <restriction base="nonNegativeInteger">
                  <totalDigits value="10" />
               </restriction>
            </simpleType>
         </attribute>
         <attribute name="fetchOffset" use="required">
            <annotation>
               <documentation xml:lang="de">
                  Offset vom Anfang der Originalauftragsdatei für die zu
                  liefernden Transaktionsdetails, bei
                  completeOrderData=false bezogen auf laufende Nummer
                  des Einzelauftrags.
               </documentation>
            </annotation>
            <simpleType>
               <restriction base="nonNegativeInteger">
                  <totalDigits value="10" />
               </restriction>
            </simpleType>
         </attribute>
         <anyAttribute namespace="##targetNamespace"
            processContents="strict" />
      </extension>
   </simpleContent>
</complexType>
<element name="Parameter">
   <annotation>
```

```
      <documentation xml:lang="de">
         generische Schlüssel-Wert-Parameter.
      </documentation>
   </annotation>
   <complexType>
      <sequence>
         <element name="Name" type="token">
            <annotation>
               <documentation xml:lang="de">
                  Name des Parameters.
               </documentation>
            </annotation>
         </element>
         <element name="Value">
            <annotation>
               <documentation xml:lang="de">
                  Wert des Parameters.
               </documentation>
            </annotation>
            <complexType>
               <simpleContent>
                  <extension base="anySimpleType">
                     <attribute name="Type" type="NCName" use="required">
                        <annotation>
                           <documentation xml:lang="de">
                              XML-Typ des Parameterwerts
                              (Vorschlag für Default=string).
                           </documentation>
                        </annotation>
                     </attribute>
                  </extension>
               </simpleContent>
            </complexType>
         </element>
      </sequence>
   </complexType>
</element>
```

8.3.3.1.3 Bedeutung der XML-Elemente/-Attribute

XML-Element/ -Attribut	Datentyp	#	Bedeutung	Beispiel
`HVTOrderParams`	`ebics:HVTOrderParams»Type` (komplex)	1	Auftragsparameter für Auftragsart HVT	- (komplex)
`PartnerID`	`ebics:PartnerIDType (→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})`	1	Kunden-ID des Einreichers	„PARTNER1“
`OrderType`	`ebics:OrderTBaseType (→token, length=3, pattern="[A-Z0-9]{3}")`	1	Auftragsart des zur VEU eingereichten Auftrags	„IZV“
`FileFormat`	`FileFomatType` (complex) `(→token)`	0..1	Dateiformat des Auftrags Bemerkung: Zu verwenden, wenn `OrderType` = “FUL” oder “FDL”	
`FileFormat» @CountryCode`	`CountryCodeType (→token, length=2,`		Information zum Anwendungsbereich des	“FR”…

	pattern= " [A-Z]{2,2}")		Formats (z.B. länderspezifische Formate)	
OrderID	ebics:OrderIDType (→token, fixLength=4)	1	Auftragsnummer des zur VEU eingereichten Auftrags	„OR01“
OrderFlags	ebics:HVTOrderFlags» Type (komplex)	1	spezifische „Schalter“ für HVT-Aufträge	- (komplex)
OrderFlags» @complete» OrderData	boolean	1	Sollen die Transaktions-details als einzelne Auftrags-detailinformationen (@completeOrderData= "false") oder als komplette Auftragsdatei (@completeOrderData= "true") übertragen werden? (Vorschlag für Default="false")	„false“
OrderFlags» @fetchLimit	nonNegativeInteger	1	Maximale Anzahl zu liefernder Auftragsdetails bei @completeOrderData= "false", „0“ für unbegrent (Vorschlag für Default=“100“)	10
OrderFlags» @fetchOffset	nonNegativeInteger	1	Offset vom Anfang der Auftragsdatei für zu liefernde Auftragsdetails bezogen auf laufende Nummer der Auftragsdetails bei @completeOrderData= "false" (Vorschlag für Default=“0“)	20
Parameter	Referenz auf globales Element (komplex)	0..∞	Struktur für generische Schlüssel-Wert-Parameter mit optionaler Typangabe	- (komplex)

8.3.3.1.4 Beispiel-XML (gekürzt)

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest xmlns="urn:org:ebics:H004"
   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:org:ebics:H004 ebics_request_H004.xsd"
   Version="H004" Revision="1">
   <header authenticate="true">
      <static>
         <!-- […] -->
         <OrderDetails>
            <OrderType>HVT</OrderType>
            <OrderAttribute>DZHNN</OrderAttribute>
            <HVTOrderParams>
               <PartnerID>PARTNER1</PartnerID>
               <OrderType>IZV</OrderType>
               <OrderID>OR01</OrderID>
               <OrderFlags completeOrderData="false" fetchLimit="50"
```

```
                fetchOffset="0" />
            </HVTOrderParams>
          </OrderDetails>
          <!-- […] -->
        </static>
        <!-- […] -->
    </header>
    <!-- […] -->
</ebicsRequest>
```

8.3.3.2 HVT-Response

Die HVT-Response enthält je nach Wahl des Attributes `completeOrderData` am Element `OrderFlags` zu dem im HVT-Request spezifizierten Auftrag zwei unterschiedliche Formate.

Mit dem Flag `completeOrderData=true` fordert das Kundensystem die Abholung der Auftragsdaten im Originalformat an. Dabei handelt es sich um einen Standard-Download ohne zusätzliche Einbettung der Auftragsdaten in ein XML-Dokument, d.h. die Auftragsdaten werden komprimiert, verschlüsselt und wenn erforderlich segmentiert zum Kundensystem übertragen.

Mit dem `Flag completeOrderData=false` fordert das Kundensystem die Abholung der Auftragsdetails im aufbereiteten XML-Format an. Dabei handelt es sich um ein XML-Dokument mit dem Wurzelelement `HVTResponseOrderData`, das komprimiert, verschlüsselt und wenn erforderlich segmentiert zum Kundensystem übertragen wird. In diesem Fall enthält die Response im Element `NumOrderInfos` die Gesamtanzahl der Auftragsdetails in der Originalauftragsdatei.

Ausprägung der (dekodierten & entschlüsselten & dekomprimierten) `OrderData` (Auftragsdaten) für HVT: `HVTResponseOrderData`

Wenn ein Teilnehmer HVT durchführt, obwohl die Bank HVT für diesen konkreten Auftrag nicht unterstützt, muss die Transaktion mit einem Returncode abgebrochen werden. Dazu ist EBICS_UNSUPPORTED_REQUEST_FOR_ORDER_INSTANCE anzuwenden.

8.3.3.2.1 XML-Schema (grafische Darstellung)

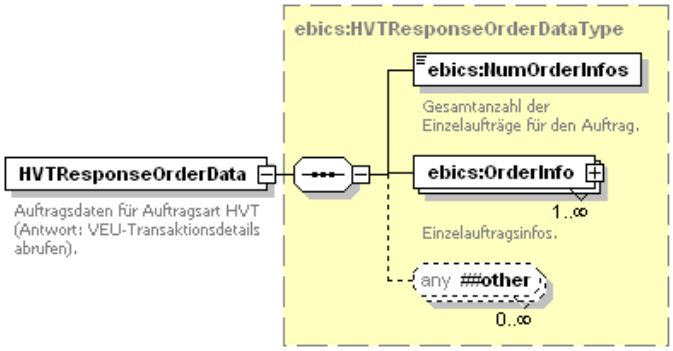


Abbildung 83: HVTResponseOrderData

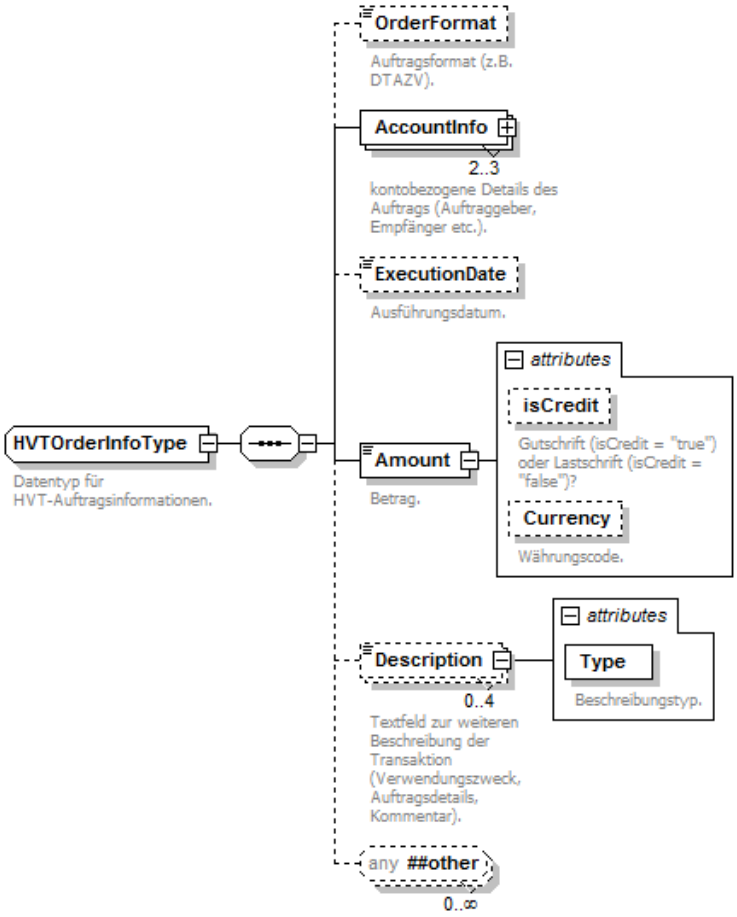


Abbildung 84: HVTOrderInfoType (zu OrderInfo)

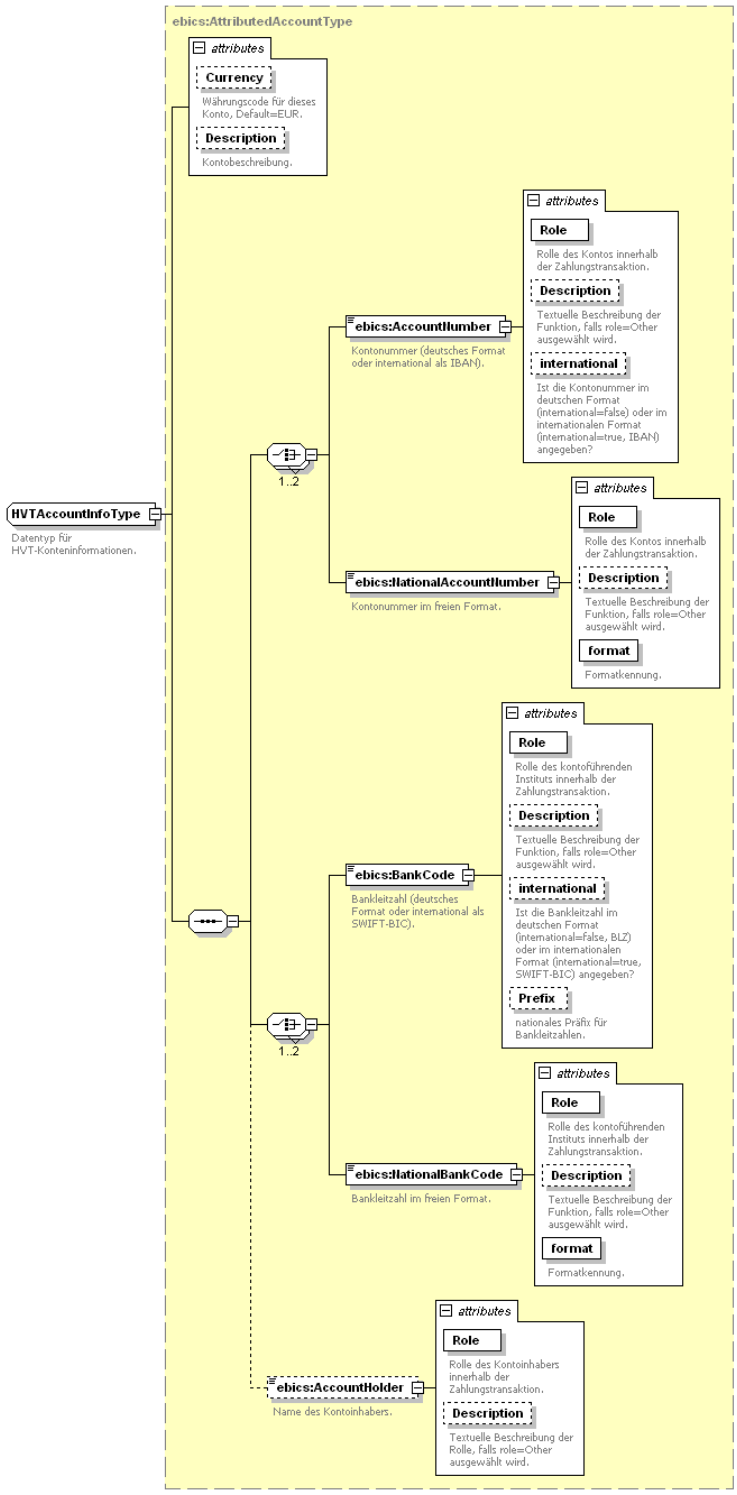


Abbildung 85: HVTAccountInfoType (zu AccountInfo)

8.3.3.2.2 XML-Schema (textuelle Darstellung)

```
<element name="HVTResponseOrderData"
   type="ebics:HVTResponseOrderDataType"
   substitutionGroup="ebics:EBICSOrderData">
   <annotation>
      <documentation xml:lang="de">
         Auftragsdaten für Auftragsart HVT (Antwort:
         VEU-Transaktionsdetails abrufen).
      </documentation>
   </annotation>
</element>
<complexType name="HVTResponseOrderDataType">
   <annotation>
      <documentation xml:lang="de">
         Datentyp für Antwort mit Einzelauftraginfos für Auftragsart HVT
         (Antwort VEU-Transaktionsdetails abrufen mit
         completeOrderData="false").
      </documentation>
   </annotation>
   <sequence>
      <element name="NumOrderInfos" type="ebics:NumOrderInfosType">
         <annotation>
            <documentation xml:lang="de">
               Gesamtanzahl der Einzelaufträge für den Auftrag.
            </documentation>
         </annotation>
      </element>
      <element name="OrderInfo" type="ebics:HVTOrderInfoType"
         maxOccurs="unbounded">
         <annotation>
            <documentation xml:lang="de">
               Einzelauftragsinfos.
            </documentation>
         </annotation>
      </element>
      <any namespace="##other" processContents="lax" minOccurs="0"
         maxOccurs="unbounded" />
   </sequence>
</complexType>
<complexType name="HVTOrderInfoType">
   <annotation>
      <documentation xml:lang="de">
         Datentyp für HVT-Auftragsinformationen.
      </documentation>
   </annotation>
   <sequence>
      <element name="OrderFormat" type="ebics:OrderFormatType"
         minOccurs="0">
         <annotation>
            <documentation xml:lang="de">
               Auftragsformat (z.B. DTAZV).
            </documentation>
         </annotation>
      </element>
      <element name="AccountInfo" type="ebics:HVTAccountInfoType"
         minOccurs="2" maxOccurs="3">
         <annotation>
            <documentation xml:lang="de">
               kontobezogene Details des Auftrags (Auftraggeber,
               Empfänger etc.).
```

```
            </documentation>
         </annotation>
      </element>
      <element name="ExecutionDate" minOccurs="0">
         <annotation>
            <documentation xml:lang="de">
               Ausführungsdatum.
            </documentation>
         </annotation>
         <complexType>
            <simpleContent>
               <extension base="date" />
            </simpleContent>
         </complexType>
      </element>
      <element name="Amount">
         <annotation>
            <documentation xml:lang="de">Betrag.</documentation>
         </annotation>
         <complexType>
            <simpleContent>
               <extension base="ebics:AmountValueType">
                  <attribute name="isCredit" type="boolean"
                     use="optional">
                     <annotation>
                        <documentation xml:lang="de">
                           Gutschrift (isCredit = "true") oder
                           Lastschrift (isCredit = "false")?
                        </documentation>
                     </annotation>
                  </attribute>
                  <attribute name="Currency"
                     type="ebics:CurrencyBaseType" use="optional">
                     <annotation>
                        <documentation xml:lang="de">
                           Währungscode.
                        </documentation>
                     </annotation>
                  </attribute>
               </extension>
            </simpleContent>
         </complexType>
      </element>
      <element name="Description" minOccurs="0" maxOccurs="4">
         <annotation>
            <documentation xml:lang="de">
               Textfeld zur weiteren Beschreibung der Transaktion
               (Verwendungszweck, Auftragsdetails, Kommentar).
            </documentation>
         </annotation>
         <complexType>
            <simpleContent>
               <extension base="string">
                  <attribute name="Type" use="required">
                     <annotation>
                        <documentation xml:lang="de">
                           Beschreibungstyp.
                        </documentation>
                     </annotation>
                     <simpleType>
                        <restriction base="token">
                           <enumeration value="Purpose">
```

```
                        <annotation>
                           <documentation xml:lang="de">
                              Verwendungszweck
                           </documentation>
                        </annotation>
                     </enumeration>
                     <enumeration value="Details">
                        <annotation>
                           <documentation xml:lang="de">
                              Auftragsdetails
                           </documentation>
                        </annotation>
                     </enumeration>
                     <enumeration value="Comment">
                        <annotation>
                           <documentation xml:lang="de">
                              Kommentar
                           </documentation>
                        </annotation>
                     </enumeration>
                  </restriction>
               </simpleType>
            </attribute>
         </extension>
      </simpleContent>
   </complexType>
</element>
<any namespace="##other" processContents="lax" minOccurs="0"
   maxOccurs="unbounded" />
</sequence>
</complexType>
<complexType name="HVTAccountInfoType">
<annotation>
   <documentation xml:lang="de">
      Datentyp für HVT-Konteninformationen.
   </documentation>
</annotation>
<complexContent>
   <extension base="ebics:AttributedAccountType" />
</complexContent>
</complexType>
<complexType name="AttributedAccountType">
<annotation>
   <documentation xml:lang="de">
      Datentyp für Kontoinformationen inkl. der Eigenschaftszuordnung
      innerhalb einer Zahlungstransaktion.
   </documentation>
</annotation>
<sequence>
   <choice maxOccurs="2">
      <element name="AccountNumber">
         <annotation>
            <documentation xml:lang="de">
               Kontonummer (deutsches Format oder international als
               IBAN).
            </documentation>
         </annotation>
         <complexType>
            <simpleContent>
               <extension base="ebics:AccountNumberType">
                  <attribute name="Role"
                     type="ebics:AccountNumberRoleType"
```

```
                  use="required">
                  <annotation>
                     <documentation xml:lang="de">
                        Rolle des Kontos innerhalb der
                        Zahlungstransaktion.
                     </documentation>
                  </annotation>
               </attribute>
               <attribute name="Description"
                  type="normalizedString">
                  <annotation>
                     <documentation xml:lang="de">
                        Textuelle Beschreibung der Funktion, falls
                        role=Other ausgewählt wird.
                     </documentation>
                  </annotation>
               </attribute>
               <attribute name="international" type="boolean"
                  use="optional" default="false">
                  <annotation>
                     <documentation xml:lang="de">
                        Ist die Kontonummer im deutschen Format
                        (international=false) oder im
                        internationalen Format
                        (international=true, IBAN) angegeben?
                     </documentation>
                  </annotation>
               </attribute>
            </extension>
         </simpleContent>
      </complexType>
   </element>
   <element name="NationalAccountNumber">
      <annotation>
         <documentation xml:lang="de">
            Kontonummer im freien Format.
         </documentation>
      </annotation>
      <complexType>
         <simpleContent>
            <extension base="ebics:NationalAccountNumberType">
               <attribute name="Role"
                  type="ebics:AccountNumberRoleType"
                  use="required">
                  <annotation>
                     <documentation xml:lang="de">
                        Rolle des Kontos innerhalb der
                        Zahlungstransaktion.
                     </documentation>
                  </annotation>
               </attribute>
               <attribute name="Description"
                  type="normalizedString">
                  <annotation>
                     <documentation xml:lang="de">
                        Textuelle Beschreibung der Funktion, falls
                        role=Other ausgewählt wird.
                     </documentation>
                  </annotation>
               </attribute>
               <attribute name="format" type="token"
                  use="required">
```

```
                    <annotation>
                       <documentation xml:lang="de">
                          Formatkennung.
                       </documentation>
                    </annotation>
                 </attribute>
              </extension>
           </simpleContent>
        </complexType>
     </element>
  </choice>
  <choice maxOccurs="2">
     <element name="BankCode">
        <annotation>
           <documentation xml:lang="de">
              Bankleitzahl (deutsches Format oder international als
              SWIFT-BIC).
           </documentation>
        </annotation>
        <complexType>
           <simpleContent>
              <extension base="ebics:BankCodeType">
                 <attribute name="Role"
                    type="ebics:BankCodeRoleType" use="required">
                    <annotation>
                       <documentation xml:lang="de">
                          Rolle des kontoführenden Instituts
                          innerhalb der Zahlungstransaktion.
                       </documentation>
                    </annotation>
                 </attribute>
                 <attribute name="Description"
                    type="normalizedString">
                    <annotation>
                       <documentation xml:lang="de">
                          Textuelle Beschreibung der Funktion, falls
                          role=Other ausgewählt wird.
                       </documentation>
                    </annotation>
                 </attribute>
                 <attribute name="international" type="boolean"
                    use="optional" default="false">
                    <annotation>
                       <documentation xml:lang="de">
                          Ist die Bankleitzahl im deutschen Format
                          (international=false, BLZ) oder im
                          internationalen Format
                          (international=true, SWIFT-BIC) angegeben?
                       </documentation>
                    </annotation>
                 </attribute>
                 <attribute name="Prefix"
                    type="ebics:BankCodePrefixType"
                    use="optional">
                    <annotation>
                       <documentation xml:lang="de">
                          nationales Präfix für Bankleitzahlen.
                       </documentation>
                    </annotation>
                 </attribute>
              </extension>
           </simpleContent>
```

```
            </complexType>
         </element>
         <element name="NationalBankCode">
            <annotation>
               <documentation xml:lang="de">
                  Bankleitzahl im freien Format.
               </documentation>
            </annotation>
            <complexType>
               <simpleContent>
                  <extension base="ebics:NationalBankCodeType">
                     <attribute name="Role"
                        type="ebics:BankCodeRoleType" use="required">
                        <annotation>
                           <documentation xml:lang="de">
                              Rolle des kontoführenden Instituts
                              innerhalb der Zahlungstransaktion.
                           </documentation>
                        </annotation>
                     </attribute>
                     <attribute name="Description"
                        type="normalizedString">
                        <annotation>
                           <documentation xml:lang="de">
                              Textuelle Beschreibung der Funktion, falls
                              role=Other ausgewählt wird.
                           </documentation>
                        </annotation>
                     </attribute>
                     <attribute name="format" type="token"
                        use="required">
                        <annotation>
                           <documentation xml:lang="de">
                              Formatkennung.
                           </documentation>
                        </annotation>
                     </attribute>
                  </extension>
               </simpleContent>
            </complexType>
         </element>
      </choice>
      <element name="AccountHolder" minOccurs="0">
         <annotation>
            <documentation xml:lang="de">
               Name des Kontoinhabers.
            </documentation>
         </annotation>
         <complexType>
            <simpleContent>
               <extension base="ebics:AccountHolderType">
                  <attribute name="role"
                     type="ebics:AccountHolderRoleType"
                     use="required">
                     <annotation>
                        <documentation xml:lang="de">
                           Rolle des Kontoinhabers innerhalb der
                           Zahlungstransaktion.
                        </documentation>
                     </annotation>
                  </attribute>
                  <attribute name="description"
```

```
                    type="normalizedString">
                    <annotation>
                       <documentation xml:lang="de">
                          Textuelle Beschreibung der Rolle, falls
                          role=Other ausgewählt wird.
                       </documentation>
                    </annotation>
                 </attribute>
              </extension>
           </simpleContent>
        </complexType>
     </element>
  </sequence>
  <attribute name="Currency" type="ebics:CurrencyBaseType"
     use="optional" default="EUR">
     <annotation>
        <documentation xml:lang="de">
           Währungscode für dieses Konto, Default=EUR.
        </documentation>
     </annotation>
  </attribute>
  <attribute name="Description" type="ebics:AccountDescriptionType"
     use="optional">
     <annotation>
        <documentation xml:lang="de">Kontobeschreibung.</documentation>
     </annotation>
  </attribute>
</complexType>
<complexType name="AmountType">
  <annotation>
     <documentation xml:lang="de">
        Datentyp für einen Betrag inkl. Währungscode-Attribut (Default
        = "EUR").
     </documentation>
  </annotation>
  <simpleContent>
     <extension base="ebics:AmountValueType">
        <attribute name="Currency" type="ebics:CurrencyBaseType"
           use="optional" default="EUR">
           <annotation>
              <documentation xml:lang="de">
                 Währungscode, Default="EUR".
              </documentation>
           </annotation>
        </attribute>
     </extension>
  </simpleContent>
</complexType>
```

8.3.3.2.3 Bedeutung der XML-Elemente/-Attribute

XML-Element/ -Attribut	Datentyp	#	Bedeutung	Beispiel
NumOrderInfos	ebics:NumOrderInfosType	1	Gesamtanzahl der Einzelaufträge in der Originaldatei	42
OrderInfo	ebics:HVTOrderInfo		Einzelauftragsinformationen	- (komplex)

	(komplex)	1..∞		
OrderFormat	ebics:OrderFormat» Type (→token, maxLength=8)	0..1	Auftragsformat	„DTAZV“
AccountInfo	ebics:HVTAccount» InfoType (komplex)	2..3	kontobezogene Detailinformationen des Einzelauftrags (Auftraggeber, Empfänger, opt. Einreicher)	- (komplex)
AccountInfo» @Currency	ebics:CurrencyBase» Type (→token, length=3, pattern="[A-Z]{3}")	0..1	Währungscode gemäß ISO 4217 des Kontos; Default= „EUR”	„EUR“
AccountInfo» @Description	ebics:Account» DescriptionType (→normalizedString)	0..1	textuelle Beschreibung des Kontos	„Savings“
ExecutionDate	date	0..1	Ausführungsdatum gemäß ISO 8601 des Einzelauftrags	2005-01-31
Amount	ebics:AmountValue» Type (→decimal, totalDigits=24, fractionDigits=4)	1	Betrag des Einzelauftrags	1234.567
Amount» @Currency	ebics:CurrencyBase» Type (→token, length=3, pattern="[A-Z]{3}")	0..1	Währungscode gemäß ISO 4217 des Einzelauftragsbetrags	„EUR“
Amount» @isCredit	boolean	0..1	Flag zur Unterscheidung zwischen Gutschrift (isCredit="true") und Lastschrift (isCredit="false")	„false“
Description	string	0..4	Textfelder zur weiteren Beschreibung der Auftragstransaktion (Verwendungszweck, Auftragsdetails, Kommentar)	„Rechnung Nr. 2345“
Description» @Type	token: "Purpose", "Details", "Comment")	1	Typ der Beschreibung: „Purpose“=Verwendungszweck, „Details“=Auftragsdetails, „Comment“=Kommentar	„Purpose“
-	-	1..2	Angaben zur Kontonummer: AccountNumber und/oder NationalAccountNumber	-
AccountNumber	ebics:AccountNumber» Type (→token, maxLength=40, pattern="\d{3,10}|([A-Z]{2}\d{2}[A-Za-z0-9]{3,30}")	1	Kontonummer, wahlweise im nationalen (=deutschen) oder internationalen Format (IBAN)	„12345678“

AccountNumber» @Role	ebics:AccountNumber» RoleType (→token: "Originator", "Recipient", "Charges", "Other")	1	Rolle des Kontos innerhalb der Zahlungstransaktion: “Originator”=Auftraggeberkonto, “Recipient”=Empfängerkonto, “Charges”=Gebührenkonto, “Other”= andere Rolle (siehe AccountNumber» @Description)	„Originator“
AccountNumber» @Description	normalizedString	0..1	textuelle Beschreibung der Rolle des Kontos innerhalb der Zahlungstransaktion, falls AccountNumber@Role= "Other" ausgewählt wird	„Nostro“
AccountNumber» @international	boolean	0..1	Ist die Kontonummer im nationalen=deutschen (AccountNumber» @international="false") oder im internationalen= IBAN-Format (AccountNumber» @international="true") angegeben? Default="false"	„false“
NationalAccount Number		1	Kontonummer im freien Format (für nationationale Kontronummer, die weder den deutschen noch den internationalen Vorgaben entsprechen)	„12345678 90123456“
NationalAccount Number» @Role	ebics:AccountNumber» RoleType (→token: "Originator", "Recipient", "Charges", "Other")	1	Rolle des Kontos innerhalb der Zahlungstransaktion: “Originator”=Auftraggeberkonto, “Recipient”=Empfängerkonto, “Charges”=Gebührenkonto, “Other”= andere Rolle (siehe AccountNumber» @Description)	„Originator“
NationalAccount Number» @Description	normalizedString	0..1	textuelle Beschreibung der Rolle des Kontos innerhalb der Zahlungstransaktion, falls AccountNumber@Role= "Other" ausgewählt wird	„Nostro“
National» AccountNumber» @format	token	1	Beschreibung für Format der Kontonummer	„other“
-	-	1..2	Angaben zur Bankleitzahl: BankCode und/oder NationalBankCode	-
BankCode	ebics:BankCodeType (→token, maxLength=11,	1	Bankleitzahl, wahlweise im nationalen (=deutschen) oder	„50010060“

	pattern="\d{8}|([A-Z]{6}[A-Z0-9]{2}([A-Z0-9]{3})?)")		internationalen Format (SWIFT-BIC)	
BankCode@Role	ebics:BankCodeRole» Type (→token: "Originator", "Recipient", "Correspondent", "Other")	1	Rolle des Kreditinstituts innerhalb der Zahlungstransaktion: „Originator"=Auftraggeberbank, „Recipient"=Empfängerbank, „Correspondent"=Korrespondenzbank, „Other"=andere Rolle (siehe BankCode@Description)	„Originator"
BankCode» @Description	normalizedString	0..1	textuelle Beschreibung der Rolle des Kreditinstituts innerhalb der Zahlungstransaktion, falls BankCode@Role="Other" ausgewählt wird	„Clearing"
BankCode» @international	boolean	0..1	Ist die Bankleitzahl im nationalen=deutschen (BankCode» @international= "false") oder internationalen=SWIFT-BIC-Format (BankCode» @international="true") angegeben? Default="false"	„false"
BankCode@Prefix	token, maxLength=2	0..1	nationales Präfix für Bankleitzahlen	„DE"
NationalBank» Code	ebics:National» BankCodeType (→token, maxLength=30)	1	Bankleitzahl im freien Format. (weder deutsches Format noch SWIFT-BIC)	„123456789012"
NationalBank» Code@Role	ebics:BankCodeRole» Type (→token: "Originator", "Recipient", "Correspondent", "Other")	1	Rolle des Kreditinstituts innerhalb der Zahlungstransaktion: „Originator"=Auftraggeberbank, „Recipient"=Empfängerbank, „Correspondent"=Korrespondenzbank, „Other"=andere Rolle (siehe BankCode@Description)	„Originator"
BankCode» @Description	normalizedString	0..1	textuelle Beschreibung der Rolle des Kreditinstituts innerhalb der Zahlungstransaktion, falls BankCode@Role="Other" ausgewählt wird	„Clearing"
NationalBank» Code@format	token	1	Formatkennung	"other"
AccountHolder	ebics:AccountHolder» Type (→normalizedString)	0..1	Name des Kontoinhabers	„Hans Mustermann"

AccountHolder» @Role	ebics:AccountHolder» RoleType (→token: "Originator", "Recipient", "Presenter", "Other")	0..1	Rolle des Kontoinhabers innerhalb der Zahlungstransaktion: „Originator“=Auftraggeber, „Recipient“=Empfänger, „Presenter“=Einreicher, „Other“=andere Rolle (siehe AccountHolder» @Description)	„Originator“
AccountHolder» @Description	normalizedString	0..1	textuelle Beschreibung der Rolle des Kontoinhabers innerhalb der Zahlungstransaktion, falls AccountHolder@Role= "Other" ausgewählt wird	„Trustee“

8.3.3.2.4 Beispiel-XML

```
<?xml version="1.0" encoding="UTF-8"?>
<HVTResponseOrderData
 xmlns="urn:org:ebics:H004"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_orders_H004.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">Arnold Auftraggeber</AccountHolder>
   </AccountInfo>
   <AccountInfo Currency="EUR">
     <AccountNumber Role="Recipient" international="false">1122334455</AccountNumber>
     <BankCode Role="Recipient" international="false">50070010</BankCode>
     <AccountHolder Role="Recipient">Erich Empfänger</AccountHolder>
   </AccountInfo>
   <ExecutionDate>2005-01-31</ExecutionDate>
   <Amount isCredit="true" Currency="EUR">500.00</Amount>
   <Description Type="Purpose">Testüberweisung</Description>
 </OrderInfo>
</HVTResponseOrderData>
```

8.3.4 HVE (Elektronische Unterschrift hinzufügen)

Mit HVE fügt der Teilnehmer einem Auftrag aus der VEU-Prozessverarbeitung eine weitere bankfachliche Unterschrift zur Autorisation hinzu.

Das Banksystem muss prüfen, ob der Teilnehmer eine bankfachliche Unterschriftsberechtigung (nicht Unterschriftsklasse T) für den referenzierten Auftrag besitzt. Falls die Berechtigung fehlt, wird die Transaktion mit dem existierenden Returncode EBICS_AUTHORISATION_ORDER_TYPE_FAILED abgebrochen.

Das Banksystem prüft vor der Ausführung von HVE, dass sich der Auftrag aktuell in der VEU-Prozessverarbeitung befindet und beendet im Fehlerfall die Transaktion mit dem fachlichen Fehlercode EBICS_ORDERID_UNKNOWN.

HVE ist eine Auftragsart vom Typ „Upload“. Das Auftragsattribut „OrderAttribute“ ist auf „UZHNN“ zu setzen. Es wird nur die EU über den Hashwert des Auftrags aus der VEU-Prozessverarbeitung übermittelt (keine Auftragsdaten, keine EU für die Auftragsart HVE selbst), wobei lediglich der Hashwert aus der VEU-Prozessverarbeitung signiert wird.

8.3.4.1 HVE-Request

Mit dem HVE-Request spezifiziert der Teilnehmer den Auftrag, zu dem er eine bankfachliche Unterschrift hinzufügen will, und liefert diese Unterschrift im gleichen Request im XML-Rumpfelement `ebicsRequest/body/DataTransfer/SignatureData` in komprimierter, verschlüsselter und base64-kodierter Form mit. Ein HVE-Request enthält keine Auftragsdaten, d.h. das XML-Rumpfelement `ebicsRequest/body/DataTransfer/OrderData` bleibt unbefüllt.
Zur Leistung der bankfachlichen Unterschrift benötigt der Teilnehmer entweder den Hashwert der ursprünglichen Auftragsdaten (z.B. mittels HVD oder HVZ abrufbar) oder die Auftragsdaten selbst (z.B. mittels HVT mit `completeOrderData="true"`).

Ausprägung der `OrderParams` (Auftragsparameter) für HVE: `HVEOrderParams`

8.3.4.1.1 XML-Schema (grafische Darstellung)

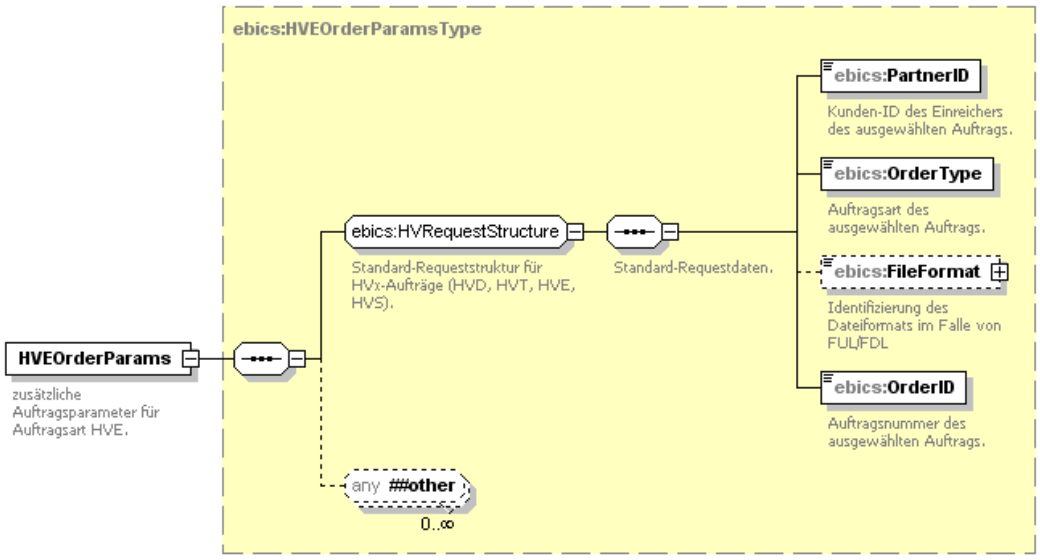


Abbildung 86: HVEOrderParams

8.3.4.1.2 XML-Schema (textuelle Darstellung)

```
 <element name="HVEOrderParams" type="ebics:HVEOrderParamsType"
substitutionGroup="ebics:OrderParams">
   <annotation>
     <documentation xml:lang="de">zusätzliche Auftragsparameter für Auftragsart
HVE.</documentation>
   </annotation>
 </element>
 <complexType name="HVEOrderParamsType">
   <annotation>
     <documentation xml:lang="de">Datentyp für zusätzliche Auftragsparameter für Auftragsart
HVE.</documentation>
   </annotation>
   <sequence>
     <group ref="ebics:HVRequestStructure"/>
     <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
 </complexType>
 <group name="HVRequestStructure">
   <annotation>
     <documentation xml:lang="de">Standard-Requeststruktur für HVx-Aufträge (HVD, HVT, HVE,
HVS).</documentation>
   </annotation>
   <sequence>
     <annotation>
       <documentation xml:lang="de">Standard-Requestdaten.</documentation>
     </annotation>
     <element name="PartnerID" type="ebics:PartnerIDType">
       <annotation>
```

```
        <documentation xml:lang="de">Kunden-ID des Einreichers des ausgewählten
Auftrags.</documentation>
      </annotation>
    </element>
    <element name="OrderType" type="ebics:OrderTBaseType">
      <annotation>
        <documentation xml:lang="de">Auftragsart des ausgewählten Auftrags gemäß Anhang
(Kapitel 13).</documentation>
      </annotation>
    </element>
    <element name="OrderID" type="ebics:OrderIDType">
      <annotation>
        <documentation xml:lang="de">Auftragsnummer des ausgewählten Auftrags gemäß Kapitel
10.1.</documentation>
      </annotation>
    </element>
  </sequence>
 </group>
```

8.3.4.1.3 Bedeutung der XML-Elemente/-Attribute

XML-Element/ -Attribut	Datentyp	#	Bedeutung	Beispiel
`HVEOrderParams`	`ebics:HVEOrderParamsType` (komplex)	1	Auftragsparameter für Auftragsart HVE	- (komplex)
`PartnerID`	`ebics:PartnerIDType` (→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})	1	Kunden-ID des Einreichers	„PARTNER1“
`OrderType`	`ebics:OrderTBaseType` (→token, length=3, pattern="[A-Z0-9]{3}")	1	Auftragsart des Auftrags in der VEU-Prozessverarbeitung	„IZV“
`FileFormat`	`FileFomatType` (complex) (→token)	0..1	Dateiformat des Auftrags Bemerkung: Zu verwenden, wenn `OrderType` = “FUL” oder “FDL”	
`FileFormat»@CountryCode`	`CountryCodeType` (→token, length=2, pattern=" [A-Z]{2,2}")		Information zum Anwendungsbereich des Formats (z.B. länderspezifische Formate)	“FR”…
`OrderID`	`ebics:OrderIDType` (→token, fixLength=4)	1	Auftragsnummer des Auftrags in der VEU-Prozessverarbeitung	„OR01“

8.3.4.1.4 Beispiel-XML (gekürzt)

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
 xmlns="urn:org:ebics:H004"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
```

```
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_request_H004.xsd"
 Version="H004" Revision="1">
  <header authenticate="true">
    <static>
      <!-- […] -->
      <OrderDetails>
        <OrderType>HVE</OrderType>
        <OrderID>HO04</OrderID>
        <OrderAttribute>UZHNN</OrderAttribute>
        <HVEOrderParams>
          <PartnerID>PARTNER1</PartnerID>
          <OrderType>IZV</OrderType>
          <OrderID>OR01</OrderID>
        </HVEOrderParams>
      </OrderDetails>
      <!-- […] -->
    </static>
    <!-- […] -->
  </header>
  <!-- […] -->
</ebicsRequest>
```

8.3.4.2 HVE-Response

Die HVE-Response enthält keine VEU-spezifischen Daten.

8.3.5 HVS (VEU-Storno)

Mit HVS storniert der Teilnehmer einen bestehenden Auftrag aus der VEU-Prozessverarbeitung unwiderruflich.

Das Banksystem prüft vor der Ausführung von HVS, dass sich der Auftrag aktuell in der VEU-Prozessverarbeitung befindet und beendet im Fehlerfall die Transaktion mit dem fachlichen Fehlercode EBICS_ORDERID_UNKNOWN.

HVS ist eine Auftragsart vom Typ „Upload“. Das Auftragsattribut „OrderAttribute“ ist auf „UZHNN“ zu setzen. Als Stornoautorisation wird die EU über den Hashwert des zu stornierenden Auftrags übermittelt (keine Auftragsdaten, keine EU für die Auftragsart HVS selbst).

8.3.5.1 HVS-Request

Mit dem HVS-Request spezifiziert der Teilnehmer den zu stornierenden Auftrag und liefert die zum Storno notwendige bankfachliche Signatur über den Hashwert der Auftragsdaten mit.

Das Banksystem muss prüfen, ob der Teilnehmer eine bankfachliche Unterschriftsberechtigung (nicht Unterschriftsklasse T) für den referenzierten Auftrag besitzt. Falls die Berechtigung fehlt, wird die Transaktion mit dem existierenden Returncode EBICS_AUTHORISATION_ORDER_TYPE_FAILED abgebrochen.

Die Signatur wird im XML-Rumpfelement `ebicsRequest/body/DataTransfer/SignatureData` in komprimierter, verschlüsselter und base64-kodierter Form transportiert. Die Stornierung ist endgültig und bedarf immer nur einer berechtigten bankfachlichen Unterschrift der Unterschriftsklassen „E“, „A“ oder „B“.

Zur Leistung der bankfachlichen Unterschrift benötigt der Teilnehmer entweder den Hashwert der ursprünglichen Auftragsdaten (z.B. mittels HVD oder HVZ abrufbar) oder die Auftragsdaten selbst (z.B. mittels HVT mit `completeOrderData="true"`).

Ausprägung der `OrderParams` für HVS: `HVSOrderParams`

8.3.5.1.1 XML-Schema (grafische Darstellung)

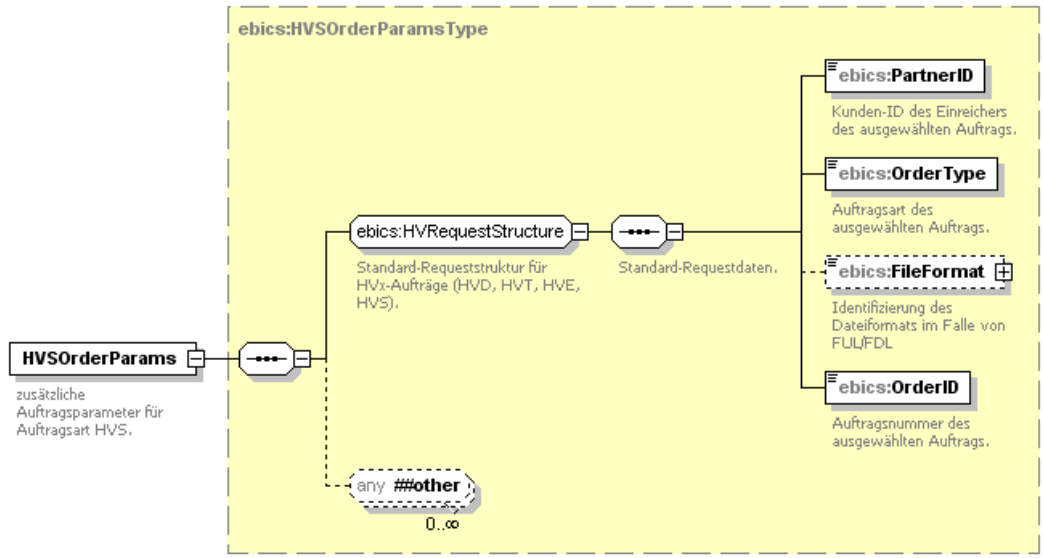


Abbildung 87: HVSOrderParams

8.3.5.1.2 XML-Schema (textuelle Darstellung)

```
 <element name="HVSOrderParams" type="ebics:HVSOrderParamsType"
substitutionGroup="ebics:OrderParams">
   <annotation>
     <documentation xml:lang="de">zusätzliche Auftragsparameter für Auftragsart
HVS.</documentation>
   </annotation>
 </element>
 <complexType name="HVSOrderParamsType">
   <annotation>
     <documentation xml:lang="de">Datentyp für zusätzliche Auftragsparameter für Auftragsart
HVS.</documentation>
   </annotation>
   <sequence>
     <group ref="ebics:HVRequestStructure"/>
     <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
 </complexType>
 <group name="HVRequestStructure">
   <annotation>
     <documentation xml:lang="de">Standard-Requeststruktur für HVx-Aufträge (HVD, HVT, HVE,
HVS).</documentation>
   </annotation>
   <sequence>
     <annotation>
       <documentation xml:lang="de">Standard-Requestdaten.</documentation>
     </annotation>
     <element name="PartnerID" type="ebics:PartnerIDType">
       <annotation>
         <documentation xml:lang="de">Kunden-ID des Einreichers des ausgewählten
Auftrags.</documentation>
       </annotation>
     </element>
```

```
      <element name="OrderType" type="ebics:OrderTBaseType">
        <annotation>
          <documentation xml:lang="de">Auftragsart des ausgewählten Auftrags gemäß Anhang
(Kapitel 13).</documentation>
        </annotation>
      </element>
      <element name="OrderID" type="ebics:OrderIDType">
        <annotation>
          <documentation xml:lang="de">Auftragsnummer des ausgewählten Auftrags gemäß Kapitel
10.1.</documentation>
        </annotation>
      </element>
    </sequence>
  </group>
```

8.3.5.1.3 Bedeutung der XML-Elemente/-Attribute

XML-Element/ -Attribut	Datentyp	#	Bedeutung	Beispiel
`HVSOrderParams`	`ebics:HVSOrderParamsType` (komplex)	1	Auftragsparameter für Auftragsart HVS	- (komplex)
`PartnerID`	`ebics:PartnerIDType` (→`token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})`	1	Kunden-ID des Einreichers.	„PARTNER1“
`OrderType`	`ebics:OrderTBaseType` (→`token, length=3, pattern="[A-Z0-9]{3}")`	1	Auftragsart des zu stornierenden Auftrags in der VEU-Prozessverarbeitung	„IZV“
`FileFormat`	`FileFomatType` (complex) (→`token`)	0..1	Dateiformat des Auftrags Bemerkung: Zu verwenden, wenn `OrderType` = “FUL” oder “FDL”	
`FileFormat»` `@CountryCode`	`CountryCodeType` (→`token, length=2, pattern=" [A-Z]{2,2}")`		Information zum Anwendungsbereich des Formats (z.B. länderspezifische Formate)	“FR”…
`OrderID`	`ebics:OrderIDType` (→`token, fixLength=4)`	1	Auftragsnummer des zu stornierenden Auftrags in der VEU-Prozessverarbeitung	„OR01“

8.3.5.1.4 Beispiel-XML (gekürzt)

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
 xmlns="urn:org:ebics:H004"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_request_H004.xsd"
 Version="H004" Revision="1">
```

```
  <header authenticate="true">
    <static>
      <!-- […] -->
      <OrderDetails>
        <OrderType>HVS</OrderType>
        <OrderID>HO05</OrderID>
        <OrderAttribute>UZHNN</OrderAttribute>
        <HVSOrderParams>
          <PartnerID>PARTNER1</PartnerID>
          <OrderType>IZV</OrderType>
          <OrderID>OR01</OrderID>
        </HVSOrderParams>
      </OrderDetails>
      <!-- […] -->
    </static>
    <!-- […] -->
  </header>
  <!-- […] -->
</ebicsRequest>
```

8.3.5.2 HVS-Response

Die HVS-Response enthält keine VEU-spezifischen Daten.

9 „Sonstige“ EBICS-Auftragsarten

In den folgenden Abschnitten werden die folgenden Auftragsarten beschrieben:

- HAA (abrufbare Auftragsarten abholen)
- HPD (Bankparameter abholen)
- HKD (Kunden- und Teilnehmerinformationen des Kunden abrufen)
- HTD (Kunden- und Teilnehmerinformationen des Teilnehmers abrufen)
- HEV (Unterstützte EBICS-Versionen abrufen).
- FUL (Datei mit beliebigem Format senden)
- FDL (Datei mit beliebigem Format abholen)

Informationen zur bankseitigen Unterstützung (verpflichtend, optional, bedingt) siehe Kapitel 13.

9.1 HAA (abrufbare Auftragsarten abholen)

Mit HAA kann sich der Teilnehmer eine Liste von Auftragsarten liefern lassen, für die im Banksystem aktuelle Kundendaten zur Abholung bereit stehen.

HAA ist eine Auftragsart vom Typ „Download“.

9.1.1 HAA-Request

Der HAA-Request enthält keine spezifischen Daten, die über die in der allgemeinen Transaktionsbeschreibung genannten hinausgehen (siehe Kapitel 5.6.1.1).

9.1.2 HAA-Response

Ausprägung der (dekodierten & entschlüsselten & dekomprimierten) `OrderData` (Auftragsdaten) für HAA: `HAAResponseOrderData`

9.1.2.1.1 XML-Schema (grafische Darstellung)

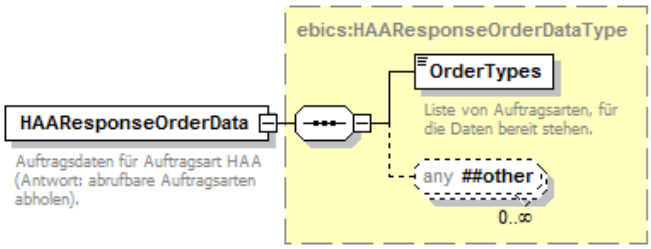


Abbildung 88: HAAResponseOrderData

9.1.2.1.2 XML-Schema (textuelle Darstellung)

```
 <element name="HAAResponseOrderData" type="ebics:HAAResponseOrderDataType"
substitutionGroup="ebics:EBICSOrderData">
   <annotation>
     <documentation xml:lang="de">Auftragsdaten für Auftragsart HAA (Antwort: abrufbare
Auftragsarten abholen).</documentation>
   </annotation>
 </element>
 <complexType name="HAAResponseOrderDataType">
   <annotation>
     <documentation xml:lang="de">Datentyp für Auftragsdaten für Auftragsart HAA (Antwort:
abrufbare Auftragsarten abholen).</documentation>
   </annotation>
   <sequence>
     <element name="OrderTypes" type="ebics:OrderTListType">
       <annotation>
         <documentation xml:lang="de">Liste von Auftragsarten, für die Daten bereit
stehen.</documentation>
       </annotation>
     </element>
     <any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
 </complexType>
```

9.1.2.1.3 Bedeutung der XML-Elemente/-Attribute

XML-Element/ -Attribut	Datentyp	#	Bedeutung	Beispiel
HAAResponse» OrderData	ebics:HAAResponse» OrderDataType (komplex)	1	Auftragsdaten für Auftragsart HAA	- (komplex)
OrderTypes	ebics:OrderTListType (→list<OrderTBaseType>	1	Liste von Auftragsarten, für die Daten bereit stehen	„STA PTK“

	→list<token, length=3, pattern="[A-Z0-9]{3}">)			

9.1.2.1.4 Beispiel-XML

```
<?xml version="1.0" encoding="UTF-8"?>
<HAAResponseOrderData
 xmlns="urn:org:ebics:H004"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_orders_H004.xsd">
 <OrderTypes>STA PTK</OrderTypes>
</HAAResponseOrderData>
```

9.2 HPD (Bankparameter abholen)

Mit HPD kann sich der Teilnehmer über die spezifischen Zugangs- (`AccessParams`) und Protokollparameter (`ProtocolParams`) des Kreditinstituts informieren.

Zu den Zugangsparametern gehören:

- `URL`: URL oder IP-Adresse für den elektronischen Zugang zum Kreditinstitut. Das optionale Attribut `valid_from` gibt den Gültigkeitsbeginn (Zeitstempel) der Angabe vor
- `Institute`: Bezeichnung des Kreditinstituts
- `HostID` (optional): EBICS Host-ID des Banksystems.

Bei den Protokollparametern werden folgende Informationen übermittelt:

- `Version`: zulässige Versionen (jeweils als Auflistung) für EBICS-Protokoll (`Protocol`), Authentifikation (`Authentication`), Verschlüsselung (`Encryption`) und Signatur (`Signature`)
- `Recovery` (optional): Unterstützung für das Wiederaufsetzen von Transaktionen (`@supported`)
- `PreValidation` (optional): Unterstützung der Vorabprüfung (`@supported`). Falls dieser Parameter gesetzt ist, sichert das Kreditinstitut damit lediglich zu, dass der Teilnehmer im Rahmen der Vorabprüfung Daten an das Kreditinstitut übermitteln kann. Das Kreditinstitut ist aber nicht verpflichtet, diese Daten in vollem Umfang zu prüfen.
- `X509Data` (optional): Unterstützung für X.509-Daten wie z.B. Zertifikate (`@supported`) aus dem XML-Feld `ebicsRequest/body/X509Data`. Darüber hinaus kann das Kreditinstitut angeben, ob es die X.509-Daten der Teilnehmer im Zustand “Bereit” persistent vorhält (`@persistent`). In diesem Fall müsste der Teilnehmer sie dann nicht bei jeder Transaktionsinitialisierung erneut übermitteln. Falls nicht angegeben, unterstützt das Kreditinstitut die persistente X.509-Datenhaltung nicht.

Für die persistente Speicherung von X.509-Daten muss das Kreditinstitut im Zuge der Teilnehmerinitialisierung die übermittelten Zertifikatsdaten speichern und der eigenen Transaktionsverwaltung zugänglich machen

- `ClientDataDownload` (optional): Unterstützung der Auftragsarten HKD (Download von Kundendaten) und HTD (Download von Teilnehmerdaten) (`@supported`). Siehe hierzu Kapitel 9.3 (HKD) bzw. 9.4 (HTD)
- `DownloadableOrderData` (optional): Unterstützung der Auftragsart HAA (abrufbare Auftragsarten abholen) (`@supported`). Siehe hierzu Kapitel 9.1

Für alle optionalen Elemente der Protokollparameter ist – sofern nicht explizit anders angegeben – folgendes Standardverhalten definiert:

- Falls der Parameter fehlt, MUSS der Teilnehmer dies als fehlende Unterstützung der entsprechenden Funktionalität werten, d.h. das Resultat entspricht *`Parameter`*`@supported="false"`
- Falls der Parameter angegeben ist, aber das Attribut `@supported` fehlt, so MUSS der Teilnehmer dies als Unterstützung der entsprechenden Funktionalität werten, d.h. das Resultat entspricht *`Parameter`*`@supported="true"`.

Diese Festlegung erleichtert die Interoperabilität von Kundenprodukt und Banksystem: Einerseits ist gewährleistet, dass ein Kreditinstitut, welches eine Funktionalität nicht unterstützt, diese in den Bankparametern auch nicht explizit als „nicht unterstützt“ aufführen muss. Andererseits wird davon ausgegangen, dass bei Nennung einer Funktionalität diese auch unterstützt wird, so dass in diesem Fall das `@supported`-Flag entfallen kann.

HPD ist eine Auftragsart vom Typ „Download“.

9.2.1 HPD-Request

Der HPD-Request enthält keine spezifischen Daten, die über die in der allgemeinen Transaktionsbeschreibung genannten hinausgehen.

9.2.2 HPD-Response

Die HPD-Response enthält die Bankparameter, aufgeteilt auf Zugangsparameter (`AccessParams`) und Protokollparameter (`ProtocolParams`).

Ausprägung der (dekodierten & entschlüsselten & dekomprimierten) `OrderData` (Auftragsdaten) für HPD: `HPDResponseOrderData`

9.2.2.1.1 XML-Schema (grafische Darstellung)

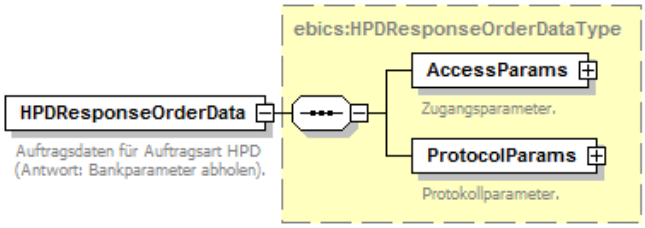


Abbildung 89: HPDResponseOrderData

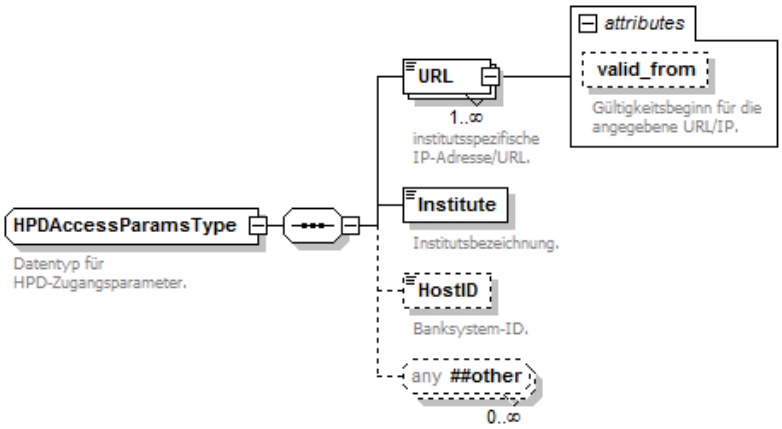


Abbildung 90: HPDAccessParamsType (zu AccessParams)

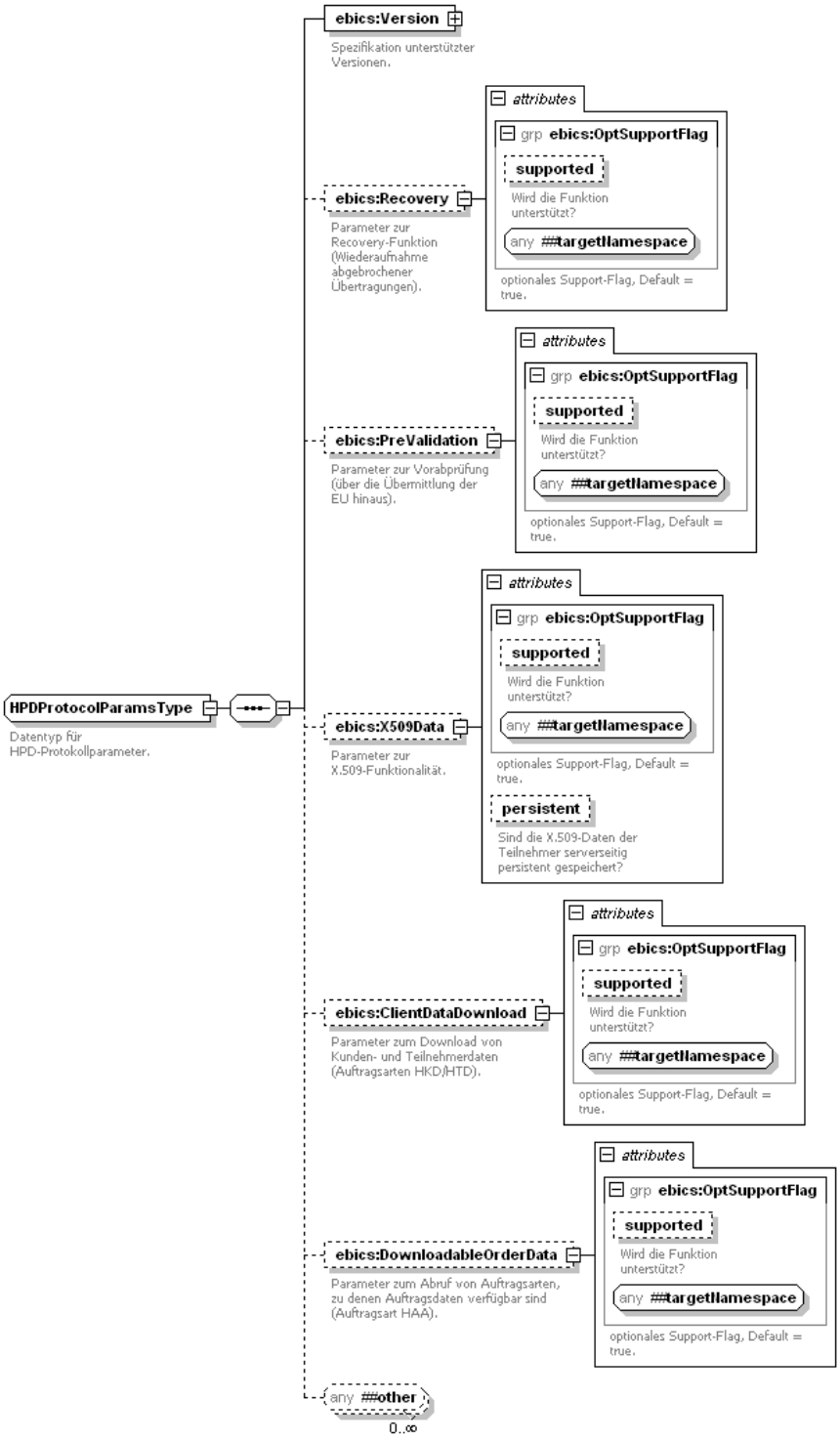


Abbildung 91: HPDProtocolParamsType (zu ProtocolParams)

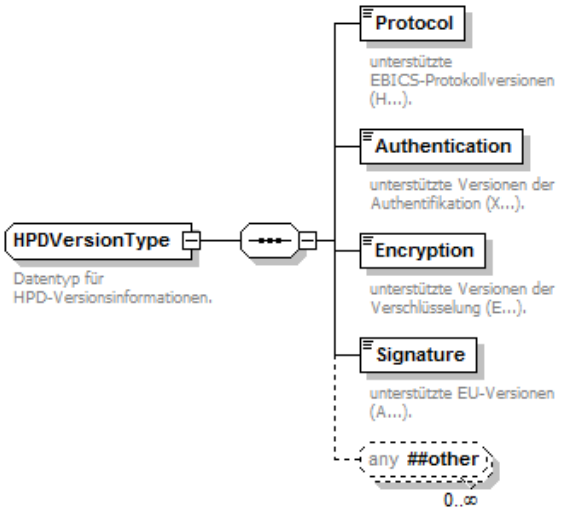


Abbildung 92: HPDVersionType (zu Version)

9.2.2.1.2 XML-Schema (textuelle Darstellung)

```
 <element name="HPDResponseOrderData" type="ebics:HPDResponseOrderDataType"
substitutionGroup="ebics:EBICSOrderData">
   <annotation>
     <documentation xml:lang="de">Auftragsdaten für Auftragsart HPD (Antwort: Bankparameter
abholen).</documentation>
   </annotation>
 </element>
 <complexType name="HPDResponseOrderDataType">
   <annotation>
     <documentation xml:lang="de">Datentyp für Auftragsdaten für Auftragsart HPD (Antwort:
Bankparameter abholen).</documentation>
   </annotation>
   <sequence>
     <element name="AccessParams" type="ebics:HPDAccessParamsType">
       <annotation>
         <documentation xml:lang="de">Zugangsparameter.</documentation>
       </annotation>
     </element>
     <element name="ProtocolParams" type="ebics:HPDProtocolParamsType">
       <annotation>
         <documentation xml:lang="de">Protokollparameter.</documentation>
       </annotation>
     </element>
   </sequence>
 </complexType>
 <complexType name="HPDAccessParamsType">
   <annotation>
     <documentation xml:lang="de">Datentyp für HPD-Zugangsparameter.</documentation>
   </annotation>
   <sequence>
     <element name="URL" maxOccurs="unbounded">
       <annotation>
```

```
          <documentation xml:lang="de">institutsspezifische IP-Adresse/URL.</documentation>
        </annotation>
        <complexType>
          <simpleContent>
            <extension base="anyURI">
              <attribute name="valid_from" type="ebics:TimestampType">
                <annotation>
                  <documentation xml:lang="de">Gültigkeitsbeginn für die angegebene
URL/IP.</documentation>
                </annotation>
              </attribute>
            </extension>
          </simpleContent>
        </complexType>
      </element>
      <element name="Institute">
        <annotation>
          <documentation xml:lang="de">Institutsbezeichnung.</documentation>
        </annotation>
        <simpleType>
          <restriction base="normalizedString">
            <maxLength value="80"/>
          </restriction>
        </simpleType>
      </element>
      <element name="HostID" type="ebics:HostIDType" minOccurs="0">
        <annotation>
          <documentation xml:lang="de">Banksystem-ID.</documentation>
        </annotation>
      </element>
      <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
  <complexType name="HPDProtocolParamsType">
    <annotation>
      <documentation xml:lang="de">Datentyp für HPD-Protokollparameter.</documentation>
    </annotation>
    <sequence>
      <element name="Version" type="ebics:HPDVersionType">
        <annotation>
          <documentation xml:lang="de">Spezifikation unterstützter Versionen.</documentation>
        </annotation>
      </element>
      <element name="Recovery" minOccurs="0">
        <annotation>
          <documentation xml:lang="de">Parameter zur Recovery-Funktion (Wiederaufnahme
abgebrochener Übertragungen).</documentation>
        </annotation>
        <complexType>
          <attributeGroup ref="ebics:OptSupportFlag"/>
        </complexType>
      </element>
      <element name="PreValidation" minOccurs="0">
        <annotation>
          <documentation xml:lang="de">Parameter zur Vorabprüfung (über die Übermittlung der EU
hinaus).</documentation>
        </annotation>
        <complexType>
          <attributeGroup ref="ebics:OptSupportFlag"/>
        </complexType>
      </element>
```

```
      <element name="X509Data" minOccurs="0">
        <annotation>
          <documentation xml:lang="de">Parameter zur X.509-Funktionalität.</documentation>
        </annotation>
        <complexType>
          <attributeGroup ref="ebics:OptSupportFlag"/>
          <attribute name="persistent" type="boolean" use="optional" default="false">
            <annotation>
              <documentation xml:lang="de">Sind die X.509-Daten der Teilnehmer serverseitig
persistent gespeichert?</documentation>
            </annotation>
          </attribute>
        </complexType>
      </element>
      <element name="ClientDataDownload" minOccurs="0">
        <annotation>
          <documentation xml:lang="de">Parameter zum Download von Kunden- und Teilnehmerdaten
(Auftragsarten HKD/HTD).</documentation>
        </annotation>
        <complexType>
          <attributeGroup ref="ebics:OptSupportFlag"/>
        </complexType>
      </element>
      <element name="DownloadableOrderData" minOccurs="0">
        <annotation>
          <documentation xml:lang="de">Parameter zum Abruf von Auftragsarten, zu denen
Auftragsdaten verfügbar sind (Auftragsart HAA).</documentation>
        </annotation>
        <complexType>
          <attributeGroup ref="ebics:OptSupportFlag"/>
        </complexType>
      </element>
      <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
  <complexType name="HPDVersionType">
    <annotation>
      <documentation xml:lang="de">Datentyp für HPD-Versionsinformationen.</documentation>
    </annotation>
    <sequence>
      <element name="Protocol">
        <annotation>
          <documentation xml:lang="de">unterstützte EBICS-Protokollversionen
(H...).</documentation>
        </annotation>
        <simpleType>
          <list itemType="ebics:ProtocolVersionType"/>
        </simpleType>
      </element>
      <element name="Authentication">
        <annotation>
          <documentation xml:lang="de">unterstützte Versionen der Authentifikation
(X...).</documentation>
        </annotation>
        <simpleType>
          <list itemType="ebics:AuthenticationVersionType"/>
        </simpleType>
      </element>
      <element name="Encryption">
        <annotation>
          <documentation xml:lang="de">unterstützte Versionen der Verschlüsselung
(E...).</documentation>
```

```
      </annotation>
      <simpleType>
        <list itemType="ebics:EncryptionVersionType"/>
      </simpleType>
    </element>
    <element name="Signature">
      <annotation>
        <documentation xml:lang="de">unterstützte EU-Versionen (A...).</documentation>
      </annotation>
      <simpleType>
        <list itemType="ebics:SignatureVersionType"/>
      </simpleType>
    </element>
    <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>
<attributeGroup name="OptSupportFlag">
  <annotation>
    <documentation xml:lang="de">optionales Support-Flag, Default = true.</documentation>
  </annotation>
  <attribute name="supported" type="boolean" use="optional" default="true">
    <annotation>
      <documentation xml:lang="de">Wird die Funktion unterstützt?</documentation>
    </annotation>
  </attribute>
  <anyAttribute namespace="##targetNamespace" processContents="strict"/>
</attributeGroup>
```

9.2.2.1.3 Bedeutung der XML-Elemente/-Attribute

XML-Element/ -Attribut	Datentyp	#	Bedeutung	Beispiel
HPDResponse» OrderData	ebics:HPDResponse» OrderDataType (komplex)	1	Auftragsdaten für Auftragsart HPD	- (komplex)
AccessParams	ebics:HPDAccessParams» Type (komplex)	1	Zugangsparameter	- (komplex)
ProtocolParams	ebics:HPDProtocol» ParamsType (komplex)	1	Protokollparameter	- (komplex)
URL	anyURI	1..∞	institutsspezifische IP-Adresse/ URL	„https://www.die-bank.de“
URL@valid_from	ebics:TimestampType (→dateTime)	0..1	Gültigkeitsbeginn für die angegebene URL/IP; falls nicht angegeben, ist die URL/IP mit sofortiger Wirkung gültig	„2005-02-28T» 15:30:45.123Z“
Institute	normalizedString, maxLength=80	1	Kreditinstitutsbezeichnung	„Die Bank“
HostID	ebics:HostIDType (→token, maxLength=35)	0..1	EBICS Banksystem-ID	„EBIXHOST“
Version	ebics:HPDVersionType (komplex)	1	Spezifikation unterstützter Versionen	- (komplex)

Protocol	list<ebics:Protocol» VersionType> (→list<token, length=4, pattern="H\d{3}">)	1	Liste unterstützter EBICS-Protokollversionen	„H004“
Authentication	list<ebics:Authentica» tionVersionType> (→list<token, length=4, pattern= "X\d{3}">)	1	Liste unterstützter Versionen der Authentifikation	„X002“
Encryption	list<ebics:Encryption» VersionType> (→list<token, length=4, pattern="E\d{3}">)	1	Liste unterstützter Versionen der Verschlüsselung	„E002“
Signature	list<ebics:Signature» VersionType> (→list<token, length=4, pattern="A\d{3}">)	1	Liste unterstützter EU-Versionen	„A004 A005 A006”
Recovery	– (komplex)	0..1	Parameter zur Recovery-Funktion (Wiederaufnahme abgebrochener Verbindungen); falls nicht angegeben, wird die Funktion nicht unterstützt	- (komplex)
Recovery» @supported	boolean	0..1	Wird Recovery unterstützt? (Default=true)	„true“
PreValidation	– (komplex)	0..1	Parameter zur Vorabprüfung; falls nicht angegeben, wird die Funktion nicht unterstützt	- (komplex)
PreValidation» @supported	boolean	0..1	Wird die Vorabprüfung unterstützt? (Default=true)	„true“
X509Data	– (komplex)	0..1	Parameter zu X.509-Daten; falls nicht angegeben, wird die Funktion nicht unterstützt	- (komplex)
X509Data» @supported	boolean	0..1	Werden X.509-Daten unterstützt? (Default=true)	„false“
X509Data» @persistent	boolean	0..1	Werden die X.509-Daten des Teilnehmers serverseitig persistent gespeichert? (Default=false)	„false“
ClientData» Download	– (komplex)	0..1	Parameter zum Download von Kunden- und Teilnehmerdaten (HKD/HTD); falls nicht angegeben, wird die	„true“

			Funktion nicht unterstützt	
ClientData» Download» @supported	boolean	0..1	Werden die Auftragsarten HKD/HTD unterstützt? (Default=true)	„true“
Downloadable» OrderData	– (komplex)	0..1	Parameter zum Abruf von Auftragsarten, zu denen Auftragsdaten verfügbar sind (HAA); falls nicht angegeben, wird die Funktion nicht unterstützt	- (komplex)
Downloadable» OrderData» @supported	boolean	0..1	Wird die Auftragsart HAA unterstützt? (Default=true)	„true“

9.2.2.1.4 Beispiel-XML

```
<?xml version="1.0" encoding="UTF-8"?>
<HPDResponseOrderData
 xmlns="urn:org:ebics:H004"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_orders_H004.xsd">
 <AccessParams>
   <URL>https://www.die-bank.de</URL>
   <URL valid_from="2005-02-15T15:30:45.123Z">192.168.0.1</URL>
   <Institute>Die Bank</Institute>
   <HostID>EBIXHOST</HostID>
 </AccessParams>
 <ProtocolParams>
   <Version>
     <Protocol>H004</Protocol>
     <Authentication>X002</Authentication>
     <Encryption>E002</Encryption>
     <Signature>A004 A005 A006</Signature>
   </Version>
   <Recovery supported="true"/>
   <PreValidation supported="true"/>
   <X509Data supported="false"/>
   <ClientDataDownload supported="true"/>
   <DownloadableOrderData supported="true"/>
 </ProtocolParams>
</HPDResponseOrderData>
```

9.3 HKD (Kunden- und Teilnehmerinformationen des Kunden abrufen)

Mit HKD kann der Teilnehmer bankseitig gespeicherte Informationen zu seinem Unternehmen und zu allen zugehörigen Teilnehmern (auch sich selbst) abrufen.

Die Antwort der Bank enthält eine Liste mit Konten des Kunden.
Ein Konto wird in der HKD Response nur dann aufgeführt, wenn mindestens eine dieser Bedingungen erfüllt ist:

1. Der Kunde hat für das Konto eine Bereitstellungsvereinbarung für Kontoauszüge.

2. Mindestens ein Teilnehmer des Kunden hat eine Unterschriftsberechtigung für das Konto.

Es ist nicht relevant, ob der Inhaber eines Kontos derselbe Kunde ist, für den HKD abgerufen wird.

HKD ist eine Auftragsart vom Typ „Download“.

9.3.1 HKD-Request

Der HKD-Request enthält keine spezifischen Daten, die über die in der allgemeinen Transaktionsbeschreibung genannten hinausgehen.

9.3.2 HKD-Response

Ausprägung der (dekodierten & entschlüsselten & dekomprimierten) `OrderData` (Auftragsdaten) für HKD: `HKDResponseOrderData`

9.3.2.1.1 XML-Schema (grafische Darstellung)

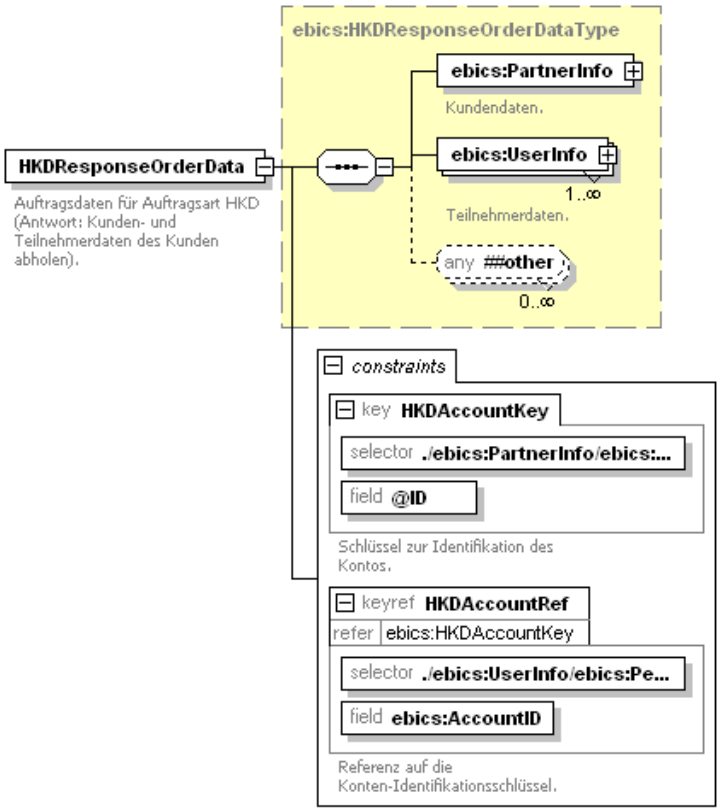


Abbildung 93: HKDResponseOrderData

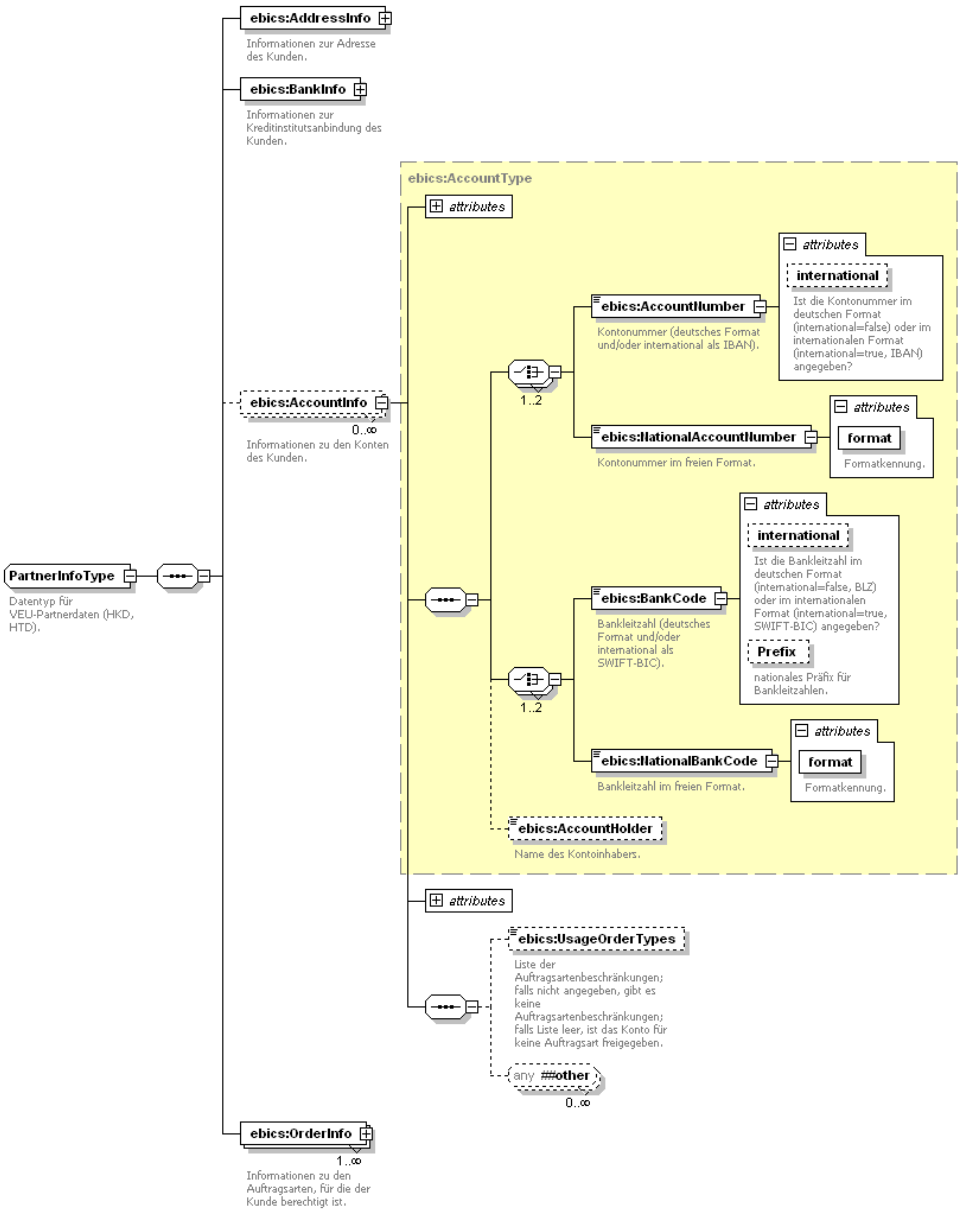


Abbildung 94: PartnerInfoType (zu PartnerInfo)

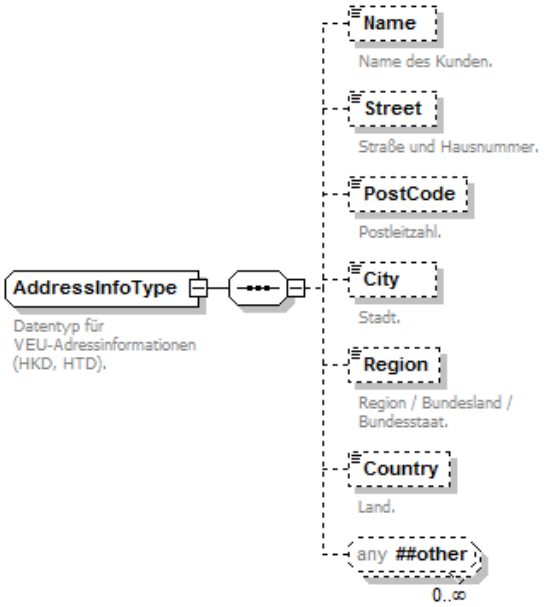


Abbildung 95: AddressInfoType (zu AddressInfo)

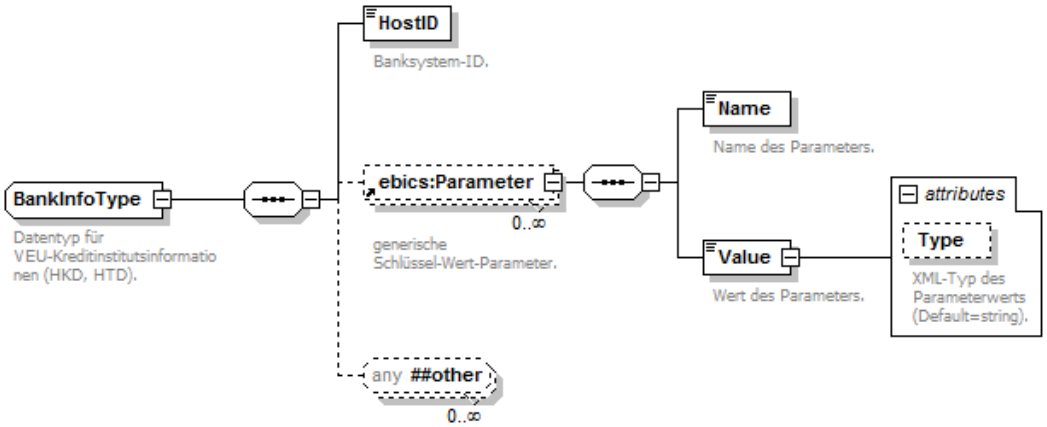


Abbildung 96: BankInfoType (zu BankInfo)

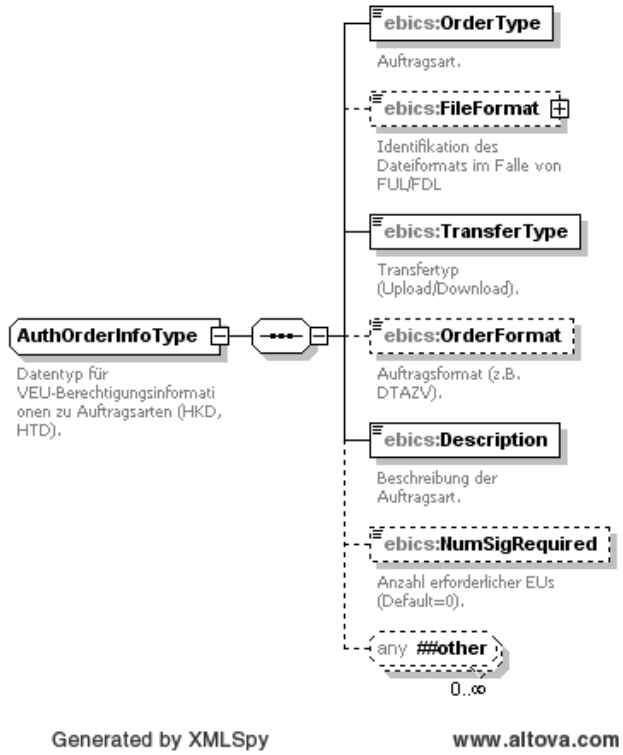


Abbildung 97: AuthOrderInfoType (zu OrderInfo)

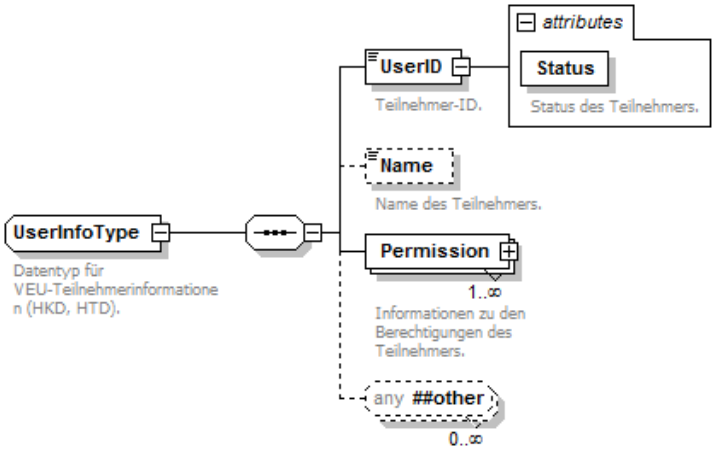


Abbildung 98: UserInfoType (zu UserInfo)

Abbildung 99: UserPermissionType (zu Permission)

9.3.2.1.2 XML-Schema (textuelle Darstellung)

```
<element name="HKDResponseOrderData"
   type="ebics:HKDResponseOrderDataType"
   substitutionGroup="ebics:EBICSOrderData">
   <annotation>
      <documentation xml:lang="de">
         Auftragsdaten für Auftragsart HKD (Antwort: Kunden- und
         Teilnehmerdaten des Kunden abholen).
      </documentation>
   </annotation>
   <key name="HKDAccountKey">
      <annotation>
         <documentation xml:lang="de">
            Schlüssel zur Identifikation des Kontos.
         </documentation>
      </annotation>
      <selector xpath="./ebics:PartnerInfo/ebics:AccountInfo" />
      <field xpath="@ID" />
   </key>
   <keyref name="HKDAccountRef" refer="ebics:HKDAccountKey">
      <annotation>
         <documentation xml:lang="de">
            Referenz auf die Konten-Identifikationsschlüssel.
         </documentation>
      </annotation>
      <selector xpath="./ebics:UserInfo/ebics:Permission" />
      <field xpath="AccountID" />
   </keyref>
</element>
```

```
<complexType name="HKDResponseOrderDataType">
   <annotation>
      <documentation xml:lang="de">
         Datentyp für Auftragsdaten für Auftragsart HKD (Antwort:
         Kunden- und Teilnehmerdaten des Kunden abholen).
      </documentation>
   </annotation>
   <sequence>
      <element name="PartnerInfo" type="ebics:PartnerInfoType">
         <annotation>
            <documentation xml:lang="de-DE">Kundendaten.</documentation>
         </annotation>
      </element>
      <element name="UserInfo" type="ebics:UserInfoType"
         maxOccurs="unbounded">
         <annotation>
            <documentation xml:lang="de">
               Teilnehmerdaten.
            </documentation>
         </annotation>
      </element>
      <any namespace="##other" minOccurs="0" maxOccurs="unbounded" />
   </sequence>
</complexType>
<complexType name="PartnerInfoType">
   <annotation>
      <documentation xml:lang="de">
         Datentyp für VEU-Partnerdaten (HKD, HTD).
      </documentation>
   </annotation>
   <sequence>
      <element name="AddressInfo" type="ebics:AddressInfoType">
         <annotation>
            <documentation xml:lang="de">
               Informationen zur Adresse des Kunden.
            </documentation>
         </annotation>
      </element>
      <element name="BankInfo" type="ebics:BankInfoType">
         <annotation>
            <documentation xml:lang="de">
               Informationen zur Kreditinstitutsanbindung des Kunden.
            </documentation>
         </annotation>
      </element>
      <element name="AccountInfo" minOccurs="0" maxOccurs="unbounded">
         <annotation>
            <documentation xml:lang="de">
               Informationen zu den Konten des Kunden.
            </documentation>
         </annotation>
         <complexType>
            <complexContent>
               <extension base="ebics:AccountType">
                  <sequence>
                     <element name="UsageOrderTypes"
                        type="ebics:OrderTListType" minOccurs="0">
                        <annotation>
                           <documentation xml:lang="de">
                              Liste der Auftragsartenbeschränkungen;
                              falls nicht angegeben, gibt es keine
                              Auftragsartenbeschränkungen; falls Liste
```

```
                        leer, ist das Konto für keine Auftragsart
                        freigegeben.
                     </documentation>
                  </annotation>
               </element>
               <any namespace="##other" processContents="lax"
                  minOccurs="0" maxOccurs="unbounded" />
            </sequence>
            <attribute name="ID" type="ebics:AccountIDType"
               use="required">
               <annotation>
                  <documentation xml:lang="de">
                     Identifikationscode des Kontos.
                  </documentation>
               </annotation>
            </attribute>
         </extension>
      </complexContent>
   </complexType>
</element>
<element name="OrderInfo" type="ebics:OrderInfoType"
   maxOccurs="unbounded">
   <annotation>
      <documentation xml:lang="de">
         Informationen zu den Auftragsarten, für die der Kunde
         berechtigt ist.
      </documentation>
   </annotation>
</element>
</sequence>
</complexType>
<complexType name="AddressInfoType">
   <annotation>
      <documentation xml:lang="de">
         Datentyp für VEU-Adressinformationen (HKD, HTD).
      </documentation>
   </annotation>
   <sequence>
      <element name="Name" type="ebics:NameType" minOccurs="0">
         <annotation>
            <documentation xml:lang="de">
               Name des Kunden.
            </documentation>
         </annotation>
      </element>
      <element name="Street" type="ebics:NameType" minOccurs="0">
         <annotation>
            <documentation xml:lang="de">
               Straße und Hausnummer.
            </documentation>
         </annotation>
      </element>
      <element name="PostCode" type="token" minOccurs="0">
         <annotation>
            <documentation xml:lang="de">Postleitzahl.</documentation>
         </annotation>
      </element>
      <element name="City" type="ebics:NameType" minOccurs="0">
         <annotation>
            <documentation xml:lang="de">Stadt.</documentation>
         </annotation>
      </element>
```

```
      <element name="Region" type="ebics:NameType" minOccurs="0">
         <annotation>
            <documentation xml:lang="de-DE">
               Region / Bundesland / Bundesstaat.
            </documentation>
         </annotation>
      </element>
      <element name="Country" type="ebics:NameType" minOccurs="0">
         <annotation>
            <documentation xml:lang="de-DE">Land.</documentation>
         </annotation>
      </element>
      <any namespace="##other" processContents="lax" minOccurs="0"
         maxOccurs="unbounded" />
   </sequence>
</complexType>
<complexType name="BankInfoType">
   <annotation>
      <documentation xml:lang="de">
         Datentyp für VEU-Kreditinstitutsinformationen (HKD, HTD).
      </documentation>
   </annotation>
   <sequence>
      <element name="HostID" type="ebics:HostIDType">
         <annotation>
            <documentation xml:lang="de">Banksystem-ID.</documentation>
         </annotation>
      </element>
      <element ref="ebics:Parameter" minOccurs="0"
         maxOccurs="unbounded" />
      <any namespace="##other" processContents="lax" minOccurs="0"
         maxOccurs="unbounded" />
   </sequence>
</complexType>
<complexType name="AuthOrderInfoType">
   <annotation>
      <documentation xml:lang="de">
         Datentyp für VEU-Berechtigungsinformationen zu Auftragsarten
         (HKD, HTD).
      </documentation>
   </annotation>
   <sequence>
      <element name="OrderType" type="ebics:OrderTBaseType">
         <annotation>
            <documentation xml:lang="de">Auftragsart.</documentation>
         </annotation>
      </element>
      <element name="FileFormat" type="ebics:FileFormatType" minOccurs="0">
       <annotation>
          <documentation xml:lang="en">File format parameter.</documentation>
       </annotation>
      </element>
      <element name="TransferType" type="ebics:TransferType">
         <annotation>
            <documentation xml:lang="de">
               Transfertyp (Upload/Download).
            </documentation>
         </annotation>
      </element>
      <element name="OrderFormat" type="ebics:OrderFormatType"
         minOccurs="0">
         <annotation>
```

```
            <documentation xml:lang="de">
               Auftragsformat (z.B. DTAUS).
            </documentation>
         </annotation>
      </element>
      <element name="Description" type="ebics:OrderDescriptionType">
         <annotation>
            <documentation xml:lang="de">
               Beschreibung der Auftragsart.
            </documentation>
         </annotation>
      </element>
      <element name="NumSigRequired" type="nonNegativeInteger"
         minOccurs="0">
         <annotation>
            <documentation xml:lang="de">
               Anzahl erforderlicher EUs (Default=0).
            </documentation>
         </annotation>
      </element>
      <any namespace="##other" processContents="lax" minOccurs="0"
         maxOccurs="unbounded" />
   </sequence>
</complexType>
<complexType name="UserInfoType">
   <annotation>
      <documentation xml:lang="de">
         Datentyp für VEU-Teilnehmerinformationen (HKD, HTD).
      </documentation>
   </annotation>
   <sequence>
      <element name="UserID">
         <annotation>
            <documentation xml:lang="de">Teilnehmer-ID.</documentation>
         </annotation>
         <complexType>
            <simpleContent>
               <extension base="ebics:UserIDType">
                  <attribute name="Status" type="ebics:UserStatusType"
                     use="required">
                     <annotation>
                        <documentation xml:lang="de">
                           Status des Teilnehmers.
                        </documentation>
                     </annotation>
                  </attribute>
               </extension>
            </simpleContent>
         </complexType>
      </element>
      <element name="Name" type="ebics:NameType" minOccurs="0">
         <annotation>
            <documentation xml:lang="de">
               Name des Teilnehmers.
            </documentation>
         </annotation>
      </element>
      <element name="Permission" type="ebics:UserPermissionType"
         maxOccurs="unbounded">
         <annotation>
            <documentation xml:lang="de">
               Informationen zu den Berechtigungen des Teilnehmers.
```

```
            </documentation>
         </annotation>
      </element>
      <any namespace="##other" processContents="lax" minOccurs="0"
         maxOccurs="unbounded" />
   </sequence>
</complexType>
<complexType name="UserPermissionType">
   <annotation>
      <documentation xml:lang="de">
         Datentyp für VEU-Berechtigungsinformationen des Teilnehmers
         (HKD, HTD).
      </documentation>
   </annotation>
   <sequence>
      <element name="OrderTypes" type="ebics:OrderTListType">
         <annotation>
            <documentation xml:lang="de">
               Liste von Auftragsarten, für die die Berechtigung des
               Teilnehmers gültig ist.
            </documentation>
         </annotation>
      </element>
      <element name="FileFormat" type="ebics:FileFormatType" minOccurs="0">
       <annotation>
         <documentation xml:lang="en">File format parameters which the user's permission
belongs to.</documentation>
       </annotation>
      </element>
      <element name="AccountID" type="ebics:AccountIDType"
         minOccurs="0">
         <annotation>
            <documentation xml:lang="de">
               Verweis auf die Identifikationscodes der berechtigten
               Konten.
            </documentation>
         </annotation>
      </element>
      <element name="MaxAmount" type="ebics:AmountType" minOccurs="0">
         <annotation>
            <documentation xml:lang="de">
               Betragshöchstgrenze, bis zu der die Berechtigung des
               Teilnehmers gültig ist.
            </documentation>
         </annotation>
      </element>
      <any namespace="##other" processContents="lax" minOccurs="0"
         maxOccurs="unbounded" />
   </sequence>
   <attribute name="AuthorisationLevel"
      type="ebics:AuthorisationLevelType" use="optional">
      <annotation>
         <documentation xml:lang="de">
            Unterschriftsklasse, für die der Teilnehmer berechtigt ist;
            nicht anzugeben bei Download-Auftragsarten.
         </documentation>
      </annotation>
   </attribute>
   <anyAttribute namespace="##targetNamespace" processContents="strict" />
</complexType>
<complexType name="AccountType">
   <annotation>
```

```
      <documentation xml:lang="de">
         Datentyp für Kontoinformationen.
      </documentation>
   </annotation>
   <sequence>
      <choice maxOccurs="2">
         <element name="AccountNumber">
            <annotation>
               <documentation xml:lang="de">
                  Kontonummer (deutsches Format und/oder international
                  als IBAN).
               </documentation>
            </annotation>
            <complexType>
               <simpleContent>
                  <extension base="ebics:AccountNumberType">
                     <attribute name="international" type="boolean"
                        use="optional" default="false">
                        <annotation>
                           <documentation xml:lang="de">
                              Ist die Kontonummer im deutschen Format
                              (international=false) oder im
                              internationalen Format
                              (international=true, IBAN) angegeben?
                           </documentation>
                        </annotation>
                     </attribute>
                  </extension>
               </simpleContent>
            </complexType>
         </element>
         <element name="NationalAccountNumber">
            <annotation>
               <documentation xml:lang="de">
                  Kontonummer im freien Format.
               </documentation>
            </annotation>
            <complexType>
               <simpleContent>
                  <extension base="ebics:NationalAccountNumberType">
                     <attribute name="format" type="token"
                        use="required">
                        <annotation>
                           <documentation xml:lang="de">
                              Formatkennung.
                           </documentation>
                        </annotation>
                     </attribute>
                  </extension>
               </simpleContent>
            </complexType>
         </element>
      </choice>
      <choice maxOccurs="2">
         <element name="BankCode">
            <annotation>
               <documentation xml:lang="de">
                  Bankleitzahl (deutsches Format und/oder international
                  als SWIFT-BIC).
               </documentation>
            </annotation>
            <complexType>
```

```
            <simpleContent>
               <extension base="ebics:BankCodeType">
                  <attribute name="international" type="boolean"
                     use="optional" default="false">
                     <annotation>
                        <documentation xml:lang="de">
                           Ist die Bankleitzahl im deutschen Format
                           (international=false, BLZ) oder im
                           internationalen Format
                           (international=true, SWIFT-BIC) angegeben?
                        </documentation>
                     </annotation>
                  </attribute>
                  <attribute name="Prefix"
                     type="ebics:BankCodePrefixType"
                     use="optional">
                     <annotation>
                        <documentation xml:lang="de">
                           nationales Präfix für Bankleitzahlen.
                        </documentation>
                     </annotation>
                  </attribute>
               </extension>
            </simpleContent>
         </complexType>
      </element>
      <element name="NationalBankCode">
         <annotation>
            <documentation xml:lang="de">
               Bankleitzahl im freien Format.
            </documentation>
         </annotation>
         <complexType>
            <simpleContent>
               <extension base="ebics:NationalBankCodeType">
                  <attribute name="format" type="token"
                     use="required">
                     <annotation>
                        <documentation xml:lang="de">
                           Formatkennung.
                        </documentation>
                     </annotation>
                  </attribute>
               </extension>
            </simpleContent>
         </complexType>
      </element>
   </choice>
   <element name="AccountHolder" type="ebics:AccountHolderType"
      minOccurs="0">
      <annotation>
         <documentation xml:lang="de">
            Name des Kontoinhabers.
         </documentation>
      </annotation>
   </element>
</sequence>
<attribute name="Currency" type="ebics:CurrencyBaseType"
   use="optional" default="EUR">
   <annotation>
      <documentation xml:lang="de">
         Währungscode für dieses Konto, Default=EUR.
```

```
         </documentation>
      </annotation>
   </attribute>
   <attribute name="Description" type="ebics:AccountDescriptionType"
      use="optional">
      <annotation>
         <documentation xml:lang="de">Kontobeschreibung.</documentation>
      </annotation>
   </attribute>
</complexType>
<complexType name="AmountType">
   <annotation>
      <documentation xml:lang="de">
         Datentyp für einen Betrag inkl. Währungscode-Attribut (Default
         = "EUR").
      </documentation>
   </annotation>
   <simpleContent>
      <extension base="ebics:AmountValueType">
         <attribute name="Currency" type="ebics:CurrencyBaseType"
            use="optional" default="EUR">
            <annotation>
               <documentation xml:lang="de">
                  Währungscode, Default="EUR".
               </documentation>
            </annotation>
         </attribute>
      </extension>
   </simpleContent>
</complexType>
<element name="Parameter">
   <annotation>
      <documentation xml:lang="de">
         generische Schlüssel-Wert-Parameter.
      </documentation>
   </annotation>
   <complexType>
      <sequence>
         <element name="Name" type="token">
            <annotation>
               <documentation xml:lang="de">
                  Name des Parameters.
               </documentation>
            </annotation>
         </element>
         <element name="Value">
            <annotation>
               <documentation xml:lang="de">
                  Wert des Parameters.
               </documentation>
            </annotation>
            <complexType>
               <simpleContent>
                  <extension base="anySimpleType">
                     <attribute name="Type" type="NCName" use="optional"
                        default="string">
                        <annotation>
                           <documentation xml:lang="de">
                              XML-Typ des Parameterwerts
                              (Default=string).
                           </documentation>
                        </annotation>
```

```
                    </attribute>
                  </extension>
                </simpleContent>
              </complexType>
            </element>
          </sequence>
        </complexType>
</element>
```

9.3.2.1.3 Bedeutung der XML-Elemente/-Attribute

XML-Element/ -Attribut	Datentyp	#	Bedeutung	Beispiel
HKDResponse» OrderData	ebics:HKDResponse» OrderDataType (komplex)	1	Auftragsdaten für Auftragsart HKD	- (komplex)
PartnerInfo	ebics:PartnerInfoType (komplex)	1	Kundendaten	- (komplex)
AddressInfo	ebics:AddressInfoType	1	Adressinformationen des Kunden	- (komplex)
Name (in AddressInfo)	ebics:NameType (→normalizedString)	0..1	Name des Kunden	„Hans Mustermann“
Street	ebics:NameType (→normalizedString)	0..1	Straße und Hausnummer des Kunden	„Musterstr. 1“
PostCode	token	0..1	Postleitzahl des Kunden	„D-12345“
City	ebics:NameType (→normalizedString)	0..1	Stadt des Kunden	„Musterstadt“
Region	ebics:NameType (→normalizedString)	0..1	Region/Bundesland/ Bundesstaat des Kunden	„Musterregion“
Country	ebics:NameType (→normalizedString)	0..1	Land des Kunden	„Musterland“
BankInfo	ebics:BankInfoType (komplex)	1	Informationen zur Kreditinstitutsanbindung des Kunden	- (komplex)
HostID	ebics:HostIDType (→token, maxLength=35	1	EBICS Banksystem-ID	„EBIXHOST“
Parameter	Referenz auf globales Element (komplex)	0..∞	Struktur für generische Schlüssel-Wert-Parameter mit optionaler Typangabe	- (komplex)
AccountInfo	ebics:AccountType (komplex)	0..∞	Informationen zu den Konten des Kunden. Ein Konto wird in der HKD Response nur dann aufgeführt, wenn der Kunde dafür eine Bereitstellungs-vereinbarung hat ODER wenn mindestens ein Teilnehmer des Kunden dafür eine Unterschrifts-	- (komplex)

			berechtigung hat. Der Inhaber des Kontos kann dabei ein anderer Kunde sein als der Kunde, für den HKD abgerufen wird.	
AccountInfo» @Currency	ebics:CurrencyBaseType (→token, length=3)	0..1	Währungscode gemäß ISO 4217 für das angegebene Konto; falls nicht angegeben, wird „EUR“ angenommen	„EUR“
Description	ebics:Account» DescriptionType (→normalizedString)	0..1	textuelle Beschreibung des Kontos	„Girokonto“
AccountInfo@ID	ebics:AccountIDType (→token, maxLength=64)	1	eindeutiger Identifikations-code des Kontos	„ABCDEFG» abcdefg» 1234567890“
-	-	1..2	Angaben zur Kontonummer: AccountNumber und/oder NationalAccountNumber	-
AccountNumber	ebics:AccountNumber» Type (→token, maxLength=40, pattern="\d{3,10}| ([A-Z]{2}\d{2} [A-Za-z0-9]{3,30})")	1	Kontonummer (deutsches Format oder international als IBAN)	„123456789“
AccountNumber» @international	boolean	0..1	Ist die Kontonummer im nationalen=deutschen (false, Default) oder im internationalen=IBAN-Format (true) angegeben?	„false“
National» AccountNumber	ebics:National» AccountNumberType (→token, maxLength=40)	1	Kontonummer im freien Format (für nationale Kontonummer, die weder den deutschen noch den internationalen Vorgaben entsprechen)	„12345678901 23456“
National» Account» Number@format	token	1	Beschreibung für Format der Kontonummer	„other“
-	-	1..2	Angaben zur Bankleitzahl: BankCode und/oder NationalBankCode	-
BankCode	ebics:BankCodeType (→token, maxLength=11, pattern="\d{8}| ([A-Z]{6}[A-Z0-9]{2} ([A-Z0-9]{3})?)")	1	Bankleitzahl (deutsches Format oder international als SWIFT-BIC)	„50010070“
BankCode» @internationa	boolean	0..1	Ist die Bankleitzahl im	„false“

l			nationalen=deutschen (false, Default) oder im internationalen=SWIFT-BIC-Format (true) angegeben?	
BankCode» @Prefix	ebics:BankCodePrefix» Type (→token, length=2)	0..1	nationales Bankleitzahl-Präfix	„DE“
NationalBank» Code	ebics:National» BankCodeType (→token, maxLength=30)	1	Bankleitzahl im freien Format. (weder deutsches Format noch SWIFT-BIC)	„123456789012“
NationalBank» Code@format	token	1	Beschreibung für Format der Bankleitzahl	“other”
AccountHolder	ebics:AccountHolder» Type (→normalizedString)	0..1	Name des Kontoinhabers	„Hans Mustermann“
UsageOrder» Types	ebics:OrderTListType (→list<ebics:» OrderTBaseType> →list<token, length=3, pattern= "[A-Z0-9]{3}">)	0..1	Liste der Auftragsarten-beschränkungen für das angegebene Konto; falls nicht angegeben, gibt es keine Auftragsartenbe-schränkungen für das an-gegebene Konto; falls die Liste leer ist, ist das ange-gebene Konto für keine Auftragsart freigegeben	„STA IZV“
OrderInfo	ebics:OrderInfoType (komplex)	1..∞	Informationen zu den dem Kunden zugewiesenen Auftragsarten	- (komplex)
OrderType	ebics:OrderTBaseType (→token, length=3, pattern="[A-Z0-9]{3}")	1	dem Kunden zugewiesene Auftragsart	„IZV“, “FDL” …
FileFormat	FileFomatType (complex) (→token)	0..1	Dem Kunden zugewiesenes Dateiformat	“camt.xxx.cfonb120.stm”
FileFormat» @CountryCode	CountryCodeType (→token, length=2, pattern= " [A-Z]{2,2}")		Information zum Anwendungsbereich des Formats (z.B. länderspezifische Formate)	“FR”…
TransferType	ebics:TransferType (→token: "Upload", "Download")	1	Transfertyp („Upload“=Auftragsdaten vom Client zum Server, „Download“=Auftragsdaten vom Server zum Client)	„Upload“
OrderFormat	ebics:OrderFormatType (→token, maxLength=8)	0..1	Datenformat des Auftrags	„DTAUS“
Description	ebics:Order» DescriptionType (→normalizedString, maxLength=128)	1	textuelle Beschreibung der Auftragsart	„Inlands-überweisung“

NumSig» Required	nonNegativeInteger	0..1	Anzahl erforderlicher EUs für die Auftragsart; Default=0, falls nicht angegeben	2
UserInfo	ebics:UserInfoType (komplex)	1..∞	Teilnehmerinformationen	- (komplex)
UserID	ebics:UserIDType (→token, maxLength=35, pattern="[a-zA-Z0-9,=]{1,35})	1	Teilnehmer-ID	„USER01“
UserID@Status	ebics:UserStatusType (→nonNegativeInteger, maxInclusive=99)	1	Status des Teilnehmers: 1: Bereit: Teilnehmer ist freigeschaltet 2: Neu: Initialzustand nach Einrichtung des Teilnehmers für EBICS („eingerichtet“) 3: Teilweise Initialisiert(INI): Teilnehmer hat INI-Datei gesendet, aber noch kein HIA 4: Teilweise Initialisiert(HIA): Teilnehmer hat HIA-Auftrag gesendet, aber noch kein INI 5: Initialisiert: Teilnehmer hat HIA-Auftrag und INI-Datei gesendet 6: Gesperrt (mehrmalige Fehlversuche), neue Initialisierung über INI und HIA ist möglich 7: Neu_FTAM: Teilnehmer ist für EBICS eingerichtet und für FTAM im Zustand bereit mit einem EBICS-konformen Signierschlüssel (A004) 8: Gesperrt (durch SPR-Auftrag des Kunden) , neue Initialisierung über INI und HIA ist möglich 9: Gesperrt (durch Bank) , neue Initialisierung über INI und HIA ist nicht möglich, Entsperrung nur durch die Bank	1
Name (in UserInfo)	ebics:NameType (→normalizedString)	0..1	Name des Teilnehmers	„Hans Mustermann“
Permission	ebics:PermissionType (komplex)	1..∞	Informationen zu den Berechtigungen des Teilnehmers	- (komplex)
Permission»	ebics:Authorisation»	0..1	Unterschriftsklasse, für die	„A“

@AuthorisationLevel	LevelType (→token, length=1: "E", "A", "B", "T")		der Teilnehmer berechtigt ist: „E“=Einzelunterschrift, „A“=Erstunterschrift, „B“=Zweitunterschrift, „T“=Transportunterschrift. Nicht anzugeben bei Download-Auftragsarten	
OrderTypes	ebics:OrderTListType (→list<OrderTBaseType> →list<token, length=3, pattern="[A-Z0-9]{3}">)	1	Liste von Auftragsarten, für die die Unterschriftsberechtigung des Teilnehmers gültig ist, getrennt durch ein Leerzeichen	„IZV AZV“, “FDL”
FileFormat	FileFomatType (complex) (→token)	0..1	Dateiformat, für das die Auftragsartenberechtigung gilt. Anmerkung: Zu verwenden, wenn OrderTypes = “FUL” or “FDL”	“camt.xxx.cfonb120.stm”
FileFormat» @CountryCode	CountryCodeType (→token, length=2, pattern=" [A-Z]{2,2}")		Information über den Anwendungsbereich des Dateiformats (z.B. länderspezifische Formate)	“FR”…
AccountID	ebics:AccountIDType (→token, maxLength=64)	0.. 1	Verweis auf den Identifikationscode eines berechtigten Kontos (Gültigkeit des Verweises wird vom EBICS XML-Schema erzwungen)	„ABCDEFG» abcdefg» 1234567890“
MaxAmount	ebics:AmountType (→ebics:AmountValue» Type →decimal, totalDigits=24, fractionDigits=4)	0..1	Betragshöchstgrenze, bis zu der die Unterschriftsberechtigung des Teilnehmers gültig ist	5000.00
MaxAmount» @Currency	ebics:CurrencyBaseType (→token, length=3)	0..1	Währung gemäß ISO 4217 des Maximalbetrags; falls nicht angegeben, wird „EUR“ angenommen	„EUR“

Bemerkung zur Klarstellung:

Die Vergabe der Kontoberechtigungen für den einzelnen Teilnehmer erfolgt über das Element `UserInfo/Permission` in folgender Weise:

Falls innerhalb von `UserInfo/Permission` kein Element `AccountID` übergeben wird, gelten die in `UserInfo/Permission` übergebenen Auftragsarten automatisch für *alle* Konten des zugehörigen Kunden.

Wird hingegen in `UserInfo/Permission` ein Element `AccountID` übergeben, so gelten die im zugehörigen Element `UserInfo/Permission/OrderTypes` übergebenen Auftragsartenberechtigungen ausschließlich für die über `AccountID` referenzierten Account-IDs.

Die Berechtigungen für FUL und FDL werden nicht als Zeichenkette definiert, in der alle Auftragsarten, jeweils durch ein Leerzeichen getrennt, hinterlegt sind: “FUL” und “FDL” werden als einzelne Berechtigungen, jeweils kombiniert mit dem zulässigen Dateiformat (in `FileFormat`) definiert:

```
<UserInfo>
   <UserID Status="1">USR200</UserID>
   <Permission AuthorisationLevel="A">
     <OrderType>FUL</OrderType>
     <FileFormat CountryCode="FR">pain.xxx.cfonb160.dct</ FileFormat >
     <MaxAmount Currency="EUR">6000.00</MaxAmount>
   </Permission>
   <Permission AuthorisationLevel="A">
     <OrderType>FUL</OrderType>
      <FileFormat CountryCode="FR">pain.001.001.02.sct</FileFormat >
      <MaxAmount Currency="EUR">12000.00</MaxAmount>
    </Permission>
    <Permission>
     <OrderType>FDL</OrderType>
     <FileFormat CountryCode="FR">camt.xxx.cfonb120.stm</FileFormat >
   </Permission>
 </UserInfo>
```

9.3.2.1.4 Beispiel-XML

```
<?xml version="1.0" encoding="UTF-8"?>
<HKDResponseOrderData
 xmlns="urn:org:ebics:H004"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_orders_H004.xsd">
 <PartnerInfo>
   <AddressInfo>
     <Name>Hans Mustermann</Name>
     <Street>Musterstr. 1</Street>
     <PostCode>D-12345</PostCode>
     <City>Musterstadt</City>
     <Region>Musterregion</Region>
     <Country>Musterland</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>Hans Mustermann</AccountHolder>
   </AccountInfo>
   <OrderInfo>
     <OrderType>STA</OrderType>
     <TransferType>Download</TransferType>
     <Description>Abholen SWIFT-Tagesauszüge</Description>
   </OrderInfo>
   <OrderInfo>
     <OrderType>IZV</OrderType>
     <TransferType>Upload</TransferType>
     <Description>Inlandszahlungsverkehrsauftrag senden</Description>
     <NumSigRequired>2</NumSigRequired>
   </OrderInfo>
```

```
  </PartnerInfo>
  <UserInfo>
    <UserID Status="1">USER01</UserID>
    <Permission>
      <OrderTypes>STA</OrderTypes>
    </Permission>
  </UserInfo>
  <UserInfo>
    <UserID Status="1">USER02</UserID>
    <Permission AuthorisationLevel="A">
      <OrderTypes>IZV</OrderTypes>
      <AccountID>accid01</AccountID>
      <MaxAmount Currency="EUR">6000.00</MaxAmount>
    </Permission>
    <Permission>
      <OrderTypes>STA</OrderTypes>
    </Permission>
  </UserInfo>
</HKDResponseOrderData>
```

9.4 HTD (Kunden- und Teilnehmerinformationen des Teilnehmers abrufen)

Mit HTD kann der Teilnehmer bankseitig gespeicherte Informationen zu seinem Unternehmen und zu sich selbst als Teilnehmer abrufen; im Unterschied zu HKD erfährt er aber nichts über die restlichen Teilnehmer des Unternehmens.

HTD ist eine Auftragsart vom Typ „Download“.

9.4.1 HTD-Request

Der HTD-Request enthält keine spezifischen Daten, die über die in der allgemeinen Transaktionsbeschreibung genannten hinausgehen.

9.4.2 HTD-Response

Ausprägung der (dekodierten & entschlüsselten & dekomprimierten) `OrderData` (Auftragsdaten) für HTD: `HTDResponseOrderData`

9.4.2.1.1 XML-Schema (grafische Darstellung)

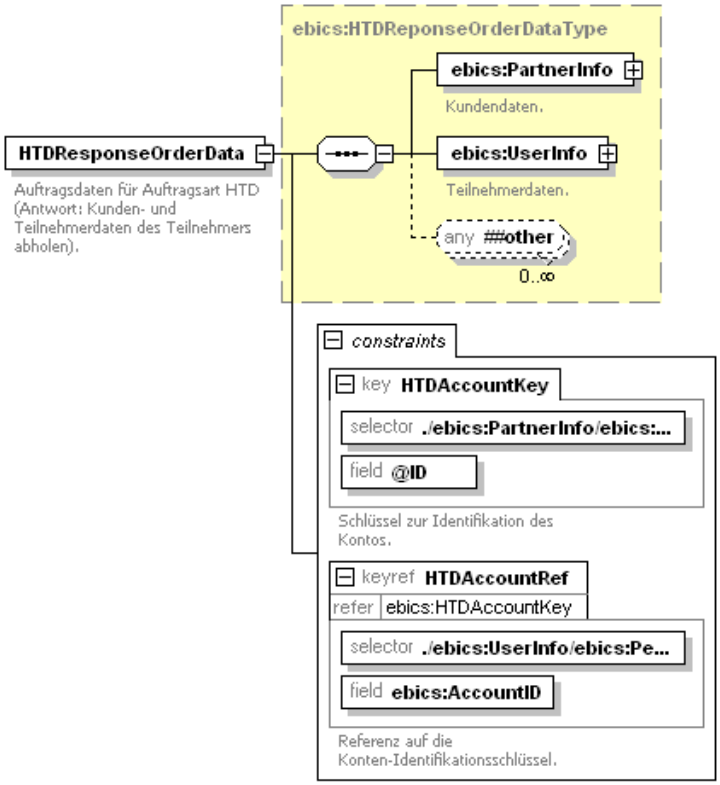


Abbildung 100: HTDResponseOrderData

9.4.2.1.2 XML-Schema (textuelle Darstellung)

```
 <element name="HTDResponseOrderData" type="ebics:HTDReponseOrderDataType"
substitutionGroup="ebics:EBICSOrderData">
   <annotation>
     <documentation>Auftragsdaten für Auftragsart HTD (Antwort: Kunden- und Teilnehmerdaten
des Teilnehmers abholen).</documentation>
   </annotation>
   <key name="HTDAccountKey">
     <annotation>
       <documentation xml:lang="de">Schlüssel zur Identifikation des Kontos.</documentation>
     </annotation>
     <selector xpath="./ebics:PartnerInfo/ebics:AccountInfo"/>
     <field xpath="@ID"/>
   </key>
   <keyref name="HTDAccountRef" refer="ebics:HTDAccountKey">
     <annotation>
       <documentation xml:lang="de">Referenz auf die Konten-
Identifikationsschlüssel.</documentation>
     </annotation>
     <selector xpath="./ebics:UserInfo/ebics:Permission"/>
```

```
      <field xpath="AccountID"/>
    </keyref>
  </element>
  <complexType name="HTDReponseOrderDataType">
    <annotation>
      <documentation>Datentyp für Auftragsdaten für Auftragsart HTD (Antwort: Kunden- und
Teilnehmerdaten des Teilnehmers abholen).</documentation>
    </annotation>
    <sequence>
      <element name="PartnerInfo" type="ebics:PartnerInfoType">
        <annotation>
          <documentation xml:lang="de">Kundendaten.</documentation>
        </annotation>
      </element>
      <element name="UserInfo" type="ebics:UserInfoType">
        <annotation>
          <documentation xml:lang="de">Teilnehmerdaten.</documentation>
        </annotation>
      </element>
      <any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
```

9.4.2.1.3 Bedeutung der XML-Elemente/-Attribute

XML-Element/ -Attribut	Datentyp	#	Bedeutung	Beispiel
`HTDResponse» OrderData`	`ebics:HTDResponse» OrderDataType` (komplex)	1	Auftragsdaten für Auftragsart HTD	- (komplex)
`PartnerInfo`	`ebics:PartnerInfoType` (komplex)	1	Kundendaten	- (komplex)
`UserInfo`	`ebics:UserInfo` (komplex)	1	Teilnehmer-informationen	- (komplex)

Für die restlichen XML-Elemente & -Attribute: siehe Auftragsart HKD (Kapitel 9.3.2.1.3).
Ebenso gilt die dort formulierte Klarstellung zur Vergabe der Kontoberechtigungen.

9.4.2.1.4 Beispiel-XML

```
<?xml version="1.0" encoding="UTF-8"?>
<HTDResponseOrderData
 xmlns="urn:org:ebics:H004"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_orders_H004.xsd">
 <PartnerInfo>
   <AddressInfo>
     <Name>Hans Mustermann</Name>
     <Street>Musterstr. 1</Street>
     <PostCode>D-12345</PostCode>
     <City>Musterstadt</City>
     <Region>Musterregion</Region>
     <Country>Musterland</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>Hans Mustermann</AccountHolder>
    </AccountInfo>
    <OrderInfo>
      <OrderType>STA</OrderType>
      <TransferType>Download</TransferType>
      <Description>Abholen SWIFT-Tagesauszüge</Description>
    </OrderInfo>
    <OrderInfo>
      <OrderType>IZV</OrderType>
      <TransferType>Upload</TransferType>
      <Description>Inlandszahlungsverkehrsauftrag senden</Description>
      <NumSigRequired>2</NumSigRequired>
    </OrderInfo>
  </PartnerInfo>
  <UserInfo>
    <UserID Status="1">USER01</UserID>
    <Permission>
      <OrderTypes>STA</OrderTypes>
    </Permission>
    <Permission AuthorisationLevel="A">
      <OrderTypes>IZV</OrderTypes>
      <AccountID>accid01</AccountID>
      <MaxAmount Currency="EUR">6000.00</MaxAmount>
    </Permission>
  </UserInfo>
</HTDResponseOrderData>
```

9.5 HEV (Unterstützte EBICS-Versionen abrufen)

Mit HEV kann sich der Teilnehmer über die bankseitig unterstützen EBICS-Versionen informieren. Die Antwort der Bank enthält eine Liste der unterstützten EBICS-Versionen und der Version des dazu gültigen Schemas.

HEV ist eine Auftragsart vom Typ „Download“.

9.5.1 HEV-Request

Der HEV-Request ruft lediglich die EBICS-Versionen, die von der Bank unterstützt werden, ab. Dieser Request kann auch von noch nicht initialisierten Teilnehmern durchgeführt werden und ist daher ohne Authentifikationssignatur möglich.
Es wird nur folgende Information verpflichtend mit dem HEV-Request übertragen:

- Host-ID des EBICS Bankrechners

Der Request wird abgewiesen und der Return Code EBICS_INVALID_HOST_ID zurückgegeben, wenn die übertragene HostID bankseitig unbekannt ist.
Hinweis: Dieser Return Code ist nur für den HEV-Request zulässig.

9.5.2 HEV-Response

Die Response liefert:

- technischer Returncode
- technischer Reporttext
- Der Wertebereich dieser beiden Felder ist dem Dokument „EBICS Anhang 1 Return Codes“ zu entnehmen. Da dem Banksystem zum Zeitpunkt des HEV-Requests die EBICS-Version des Kundensystems noch nicht bekannt ist, befüllt es die Felder mit den Werten, die in der höchsten bankseitig unterstützten EBICS-Version definiert sind. Da in diesem Request kein Sprach-Attribut verfügbar ist, wird der Reporttext stets in Englisch zu übermittelt.
- Liste der vom Banksystem unterstützten EBICS-Versionen und der Name der dazu gültigen Schema-Version

9.5.3 Schema für HEV-Request / HEV-Response

Für HEV-Request und HEV-Response wird ein neutrales -von den aktuell von der Bank unterstützten EBICS-Versionen unabhängiges - Schema ebics_hev.xsd, abrufbar unter http://www.ebics.org (Kategorie „Spezifikation“ - verwendet. Das Schema enthält Request

und Response. Beim Request ist ebicsHEVRequest und bei der Response ebicsHEVResponse zu füllen.

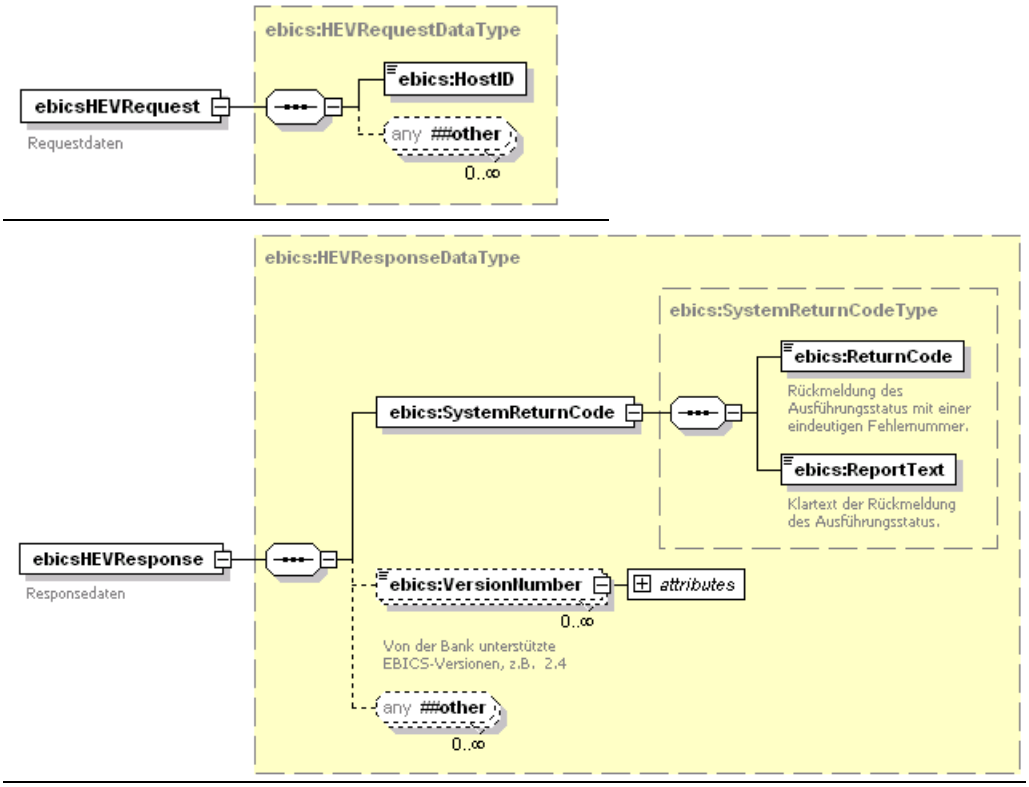


Abbildung 101: HEVRequest / HEVResponse

Die textuelle Darstellung des Schemas ebics_hev.xsd findet sich unter www.ebics.org.

9.5.3.1 Bedeutung der XML-Elemente /-Attribute der HEV-Response

XML-Element/ -Attribut	Datentyp	#	Bedeutung	Beispiel
`SystemReturnCode`	`ebics:SystemReturnCodeType` (komplex)	1	Technischer Return-Code und Fehlertext (in Englisch)	Wertebereich für Code gemäß Dokument „EBICS Anhang 1

				Return Codes“
VersionNumber	ebics:VersionNumberType (komplex) (→token, length=5, pattern="[0-9]{2}[.][0-9]{2}"	0..∞	Vom Banksystem unterstützte EBICS-Version	02.40 (entspricht also 2.4)
ProtocolVersion	ebics:ProtocolVersionType (→token, length=4)	1	Die zur unterstützenden EBICS-Version gültige Schemaversion	H004

9.5.3.2 Beispiel-XML für eine 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="H003">02.40</ebics:VersionNumber>
       <ebics:VersionNumber ProtocolVersion="H004">02.50</ebics:VersionNumber>
</ebics:ebicsHEVResponse>
```

9.6 FUL und FDL (Datei mit beliebigem Format senden und abholen)

FUL ist eine Auftragsart vom Typ „Upload“. Der Standardablauf ist in Kapitel 5.5.1 beschrieben. Die Ausprägung der Auftragsparameter (dazu auch 3.11) `ebicsRequest/header/static/OrderDetails` ist jedoch vom Typ `FULOrderParams`:

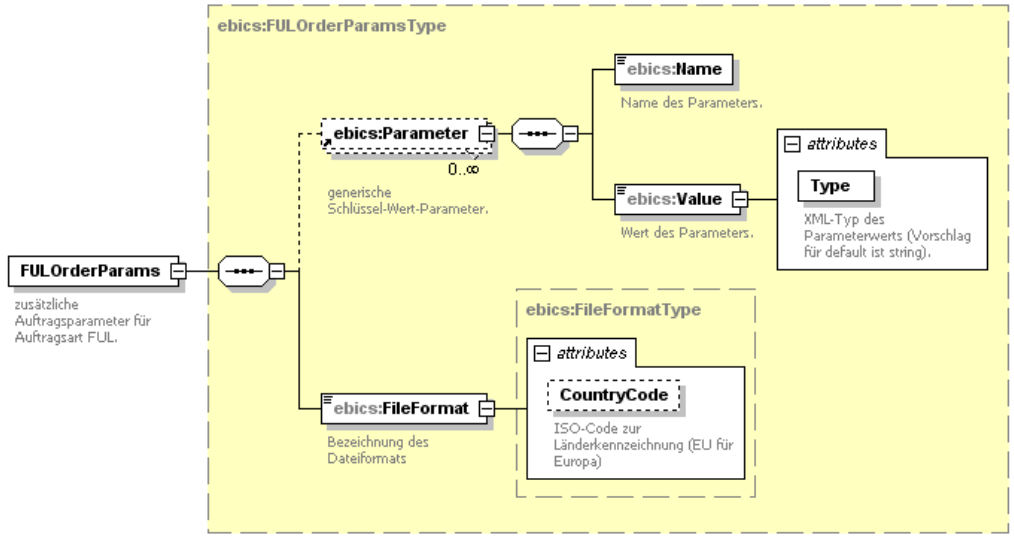


Abbildung 102: FULOrderParams

XML-Element/ -Attribut	Datentyp	#	Bedeutung	Beispiele
`FileFormat`	`FileFomatType` (complex) (→`token`)	1	Beschreibt das Format der gesendeten Datei	„DTAUS“ , „pain.001.001.02“
`FileFormat»` `@CountryCode`	`CountryCodeType` (→`token, length=2, pattern=` `" [A-Z]{2,2}")`		Information zum Verwendungsbereich des Formates (z.B. länderspezifische Formate)	EU, FR, DE …

FDL ist eine Auftragsart vom Typ „Download“. Der Standardablauf ist in Kapitel 5.6.1 beschrieben. Die Ausprägung der Auftragsparameter (dazu auch 3.11) `ebicsRequest/header/static/OrderDetails` ist jedoch vom Typ `FDLOrderParams`:

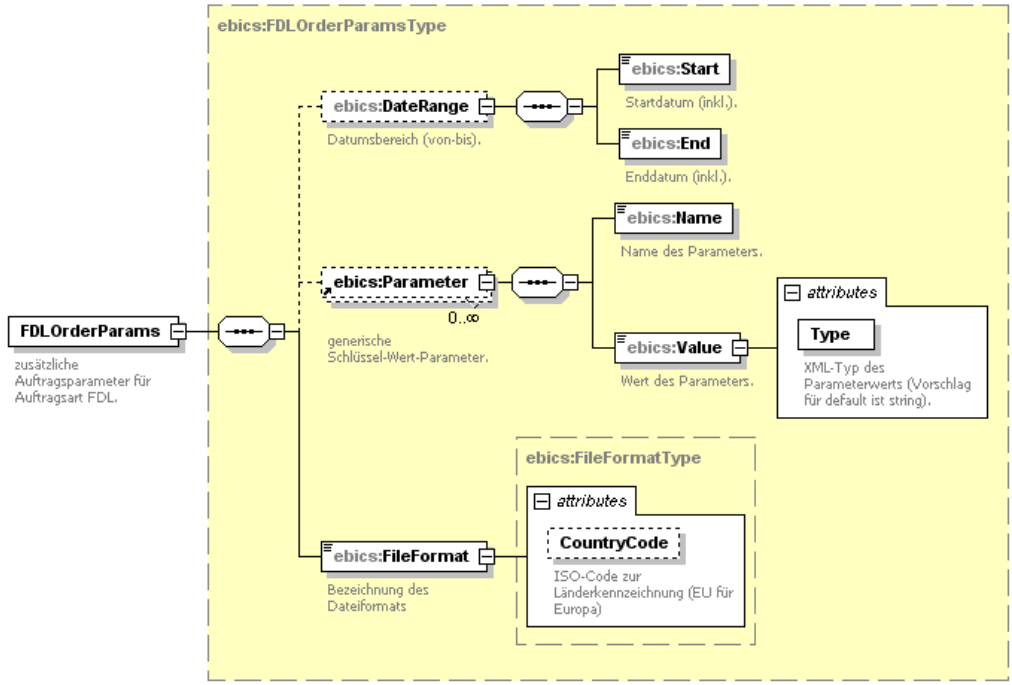


Abbildung 103: FDLOrderParams

Wenn FUL bzw. FDL vom Kreditinstitut nicht unterstützt wird, ist die Fehlermeldung EBICS_UNSUPPORTED_ORDER_TYPE zurückzugeben.

Wenn das durch `FileFomat` und `CountryCode` spezifizierte Auftragsformat nicht vom Kreditinstitut nicht unterstützt wird, ist die Fehlermeldung EBICS_INVALID_ORDER_PARAMS zurückzugeben.

10 EBICS-Kundenprotokoll (HAC)

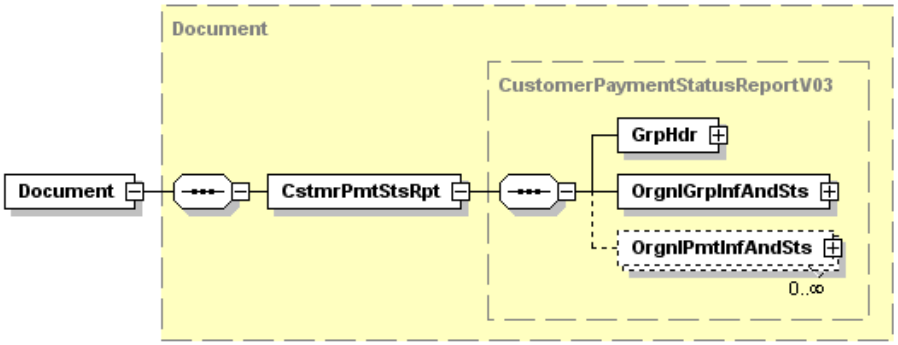
10.1 Vorbemerkungen

1. Die folgenden Festlegungen für die Belegung der pain.002-Nachricht (ISO Edition 2009) beziehen sich nur auf das EBICS-Kundenprotokoll (HAC: H Kennzeichen für eine technische EBICS-Auftragsart; A = Acknowledgement; C = Customer).
   HAC beschreibt alle Aktionen und Ergebnisse, die beim Senden, Abholen und Unterschreiben von Dateien vorkommen und kann zusätzlich Informationen über den Inhalt des Auftrags / der Datei geben (Dateianzeige).

2. Die Zielvorgabe war, dafür einen internationalen Standard (Schema) zu nutzen; man entschied sich für die ISO20022-Nachricht pain.002.
   - Denn zurzeit ist die pain.002-Nachricht die beste Alternative, auch wenn sie nicht ideal ist. Zudem wird die pain.002-Nachricht bereits als EBICS-Kundenprotokoll in Frankreich verwendet (allerdings mit weniger Belegungsrichtlinien; in Deutschland gibt es mehr Anforderungen, die Protokollierung von EBICS-Aktionen betreffend). Das Ziel ist ein gemeinsame Teilmenge von Belegungsrichtilien für die pain.002-Nachricht in Deutschland und Frankreich (EBICS-Kundenprotokoll; Auftragsart HAC)
   - Als langfristige Lösung sollte eine eigens für diesen Zweck spezifizierte ISO-Nachricht
3. Beim Abholen von HAC erhält der Kunde alle Statusinformationen seit letzter Abholung von HAC. Das Kundenprotokoll enthält alle Aktionen und Statusinformationen der PartnerID (Kunden-ID). Dafür kommt die Elementgruppe <OrgnlPmtInfAndSts> 1 bis n Mal vor. Jede Wiederholung ist ein Protokollschritt.
   Hinweis: Eigentlich ist diese Elementgruppe optional. Eine wesentliche Regel für das EBICS-Kundenprotokoll ist, dass die Elementgruppe <OrgnlPmtInfAndSts> mindestens einmal vorkommt.
4. Im Gegensatz zum “menschenlesbaren” alten deutschen Kundenprotokoll PTK liegt der Schwerpunkt beim XML-basierten HAC auf der automatischen Auswertung (geeignete Aufbereitung durch das Kundensystem ist notwendig)

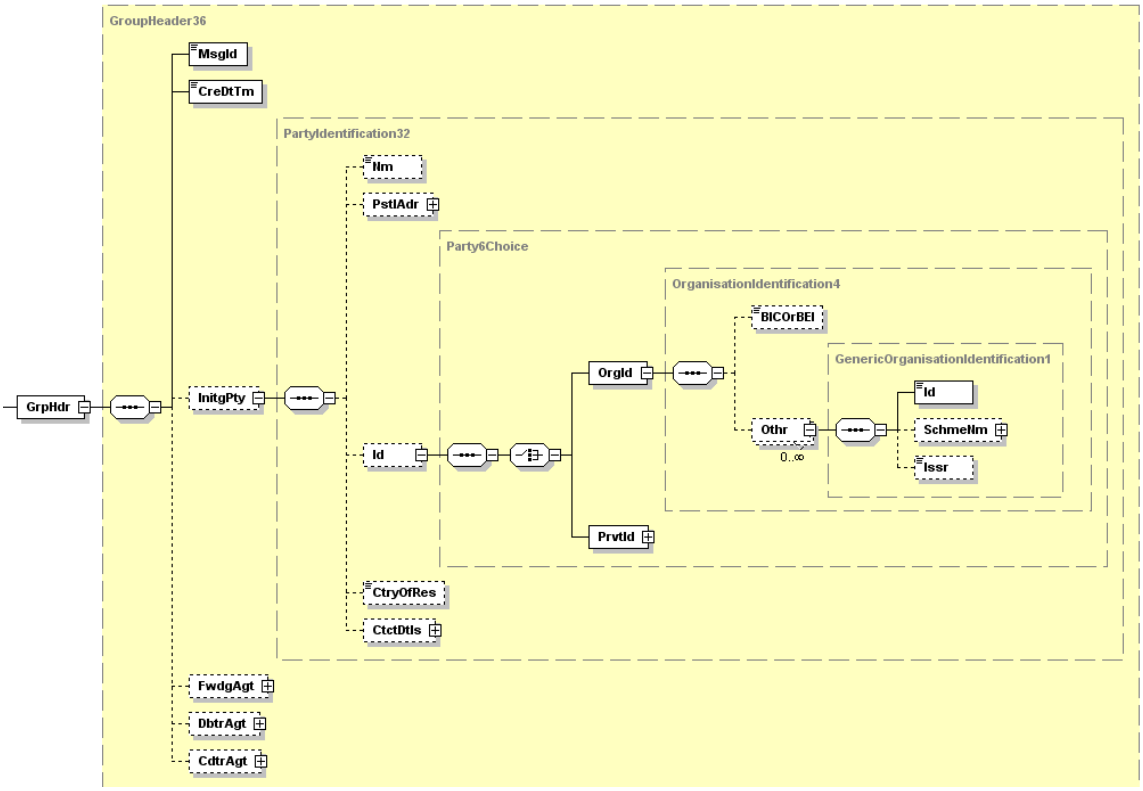
10.2 Belegung von pain.002 für HAC

Gesamte Nachricht – Allgemeiner Überblick:



10.2.1 Belegung der Elementgruppe Group Header

Elementgruppe <GrpHdr> - Überblick:



Festlegungen für die Belegung:
Diese Elementgruppe kommt genau einmal vor.
Alle Elemente in <GrpHdr>, die in dieser Liste nicht erwähnt werden, werden nie in HAC verwendet!

Name	XML Tag	Regeln für HAC
MessageIdentification	<MsgId>	Verpflichtend gemäß ISO-Schema (und somit auch in HAC)
CreationDateTime	<CreDtTm>	Verpflichtend gemäß ISO-Schema (und somit auch in HAC): Erstellungsdatum/-uhrzeit der pain.002-Nachricht
InitiatingParty	<InitPty><Id><OrgId>	Elementgruppe für die Übertragung der HostID (optional im ISO-Schema, jedoch verpflichtend in HAC) HostId (technische ID des EBICS-Bankrechners) ist in <Othr> zu hinterlegen.

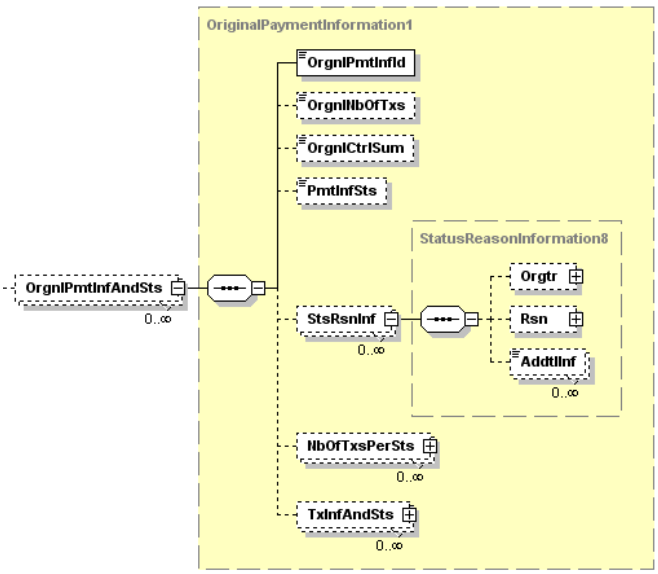
10.2.2 Belegung der Elementgruppe Original Group Information and Status

Wie in ISO kommt diese Elementgruppe genau einmal vor.
Es gibt zwei Pflichtfelder: <OrgnlMsgId> und <OrgnlMsgNmId> (beide vom Typ Max35Text). Sie legen Gruppierungsinformationen der Originalnachricht dar. Da HAC Informationen über eine Sammlung von verschiedenen Aufträgen und Aktionen eines Kunden (einer PartnerID), darstellt, werden diese Elemente mit der Konstante „EBICS“ belegt.

10.2.3 Belegung der Elementgruppe Original Payment Information and Status

Diese Elementgruppe ist in ISO optional, in HAC muss sie jedoch mindestens einmal vorkommen!

Elementgruppe <OrgnlPmtInfAndSts> - Überblick:



Festlegungen für die Belegung:
Alle Elemente in <OrgnlPmtInfAndSts>, die in dieser Liste nicht erwähnt werden, werden nie in HAC verwendet!

Name	XML Tag	Regeln für HAC
OriginalPaymentInforamtionIdentification	<OrgnlPmtInfId>	Verpflichtend im ISO-Schema (und somit auch in HAC). Informationen über die Art der Aktion; siehe Kapitel 10.2.3.1
StatusReasonInformation	<StsRsnInf>	[0..unbounded] im ISO-Schema, in HAC kommt diese Elementgruppe genau einmal vor Informationen über den Auftrag (dazu gehören alle beteiligten Teilnehmer und der Timestamp), Ergebnis der Aktion und Daten für die Dateianzeige; siehe Kapitel 10.2.3.2

10.2.3.1 Art der Aktion

Die Art der Aktion wird im Element <OrgnlPmtInfId> hinterlegt.
Der folgende Wertebereich ist dafür definiert:

HAC – Art der Aktion	Bedeutung	Wertebereich (Max35Text); für EBICS definierter Wert
„Art der Aktion“ *Übertragung*		
Datei wurde zur Bank übertragen	Jede Art von Upload außer dem Upload einer EU-Datei	FILE_UPLOAD
Datei wurde von der Bank abgeholt	Each kind of file download except the download of an ES file	FILE_DOWNLOAD

HAC – Art der Aktion	Bedeutung	Wertebereich (Max35Text); für EBICS definierter Wert
Elektronische Unterschrift wurde zur Bank übertragen	Upload einer EU-Datei unter Verwendung der Auftragsattribute "UZHNN" oder innerhalb des VEU-Verfahrens (Auftragsart HVE)	ES_UPLOAD
Elektronische Unterschrift wurde von der Bank abgeholt	Download einer EU-Datei (für eine spätere Version reserviert)	ES_DOWNLOAD
„Art der Aktion" *Nachverarbeitung (VEU etc.)*		
Unterschriftsprüfung	Der Bankrechner überprüft die übertragene EU	ES_VERIFICATION Code, wenn **keine** VEU eingesetzt wird.
Weiterleitung zur VEU	Die Datei wird im VEU-Verfahren gespeichert und wartet auf die erforderlichen EU's	VEU_FORWARDING
VEU Unterschriftsprüfung	Der Bankrechner überprüft die übertragenen EUs innerhalb des VEU-Verfahrens	VEU_VERIFICATION Code, wenn die VEU eingesetzt wird.
Abschluss der VEU Unterschriftsprüfung	Der Prüfprozess im VEU-Verfahren wurde abgeschlossen, da die letzte für die Autorisierung einer Zahlung erforderliche Unterschrift nun erfolgreich geprüft wurde **oder** die übertragene Unterschrift nicht korrekt war.	VEU_VERIFICATION_END
Stornierung eines VEU-Auftrags	Der Auftrag wurde durch einen autorisierten Teilnehmer innerhalb des VEU-Verfahrens storniert	VEU_CANCEL_ORDER
„Art der Aktion" *Zusatzinformationen*	Bereitstellung von zusätzlichen Informationen der Bank an den Kunden mittels HAC (in <AddtlInf>)	ADDITIONAL

Eines der beiden Ende-Kennzeichen ist anzugeben um kenntlich zu machen, dass kein weiterer EBICS-Protokollschritt für den angegebenem Auftrag mehr. Für diese Arten der Aktion ist kein Reason-Code (im „Ende der Aktion") zulässig. Es ist lediglich ein Kennzeichen.

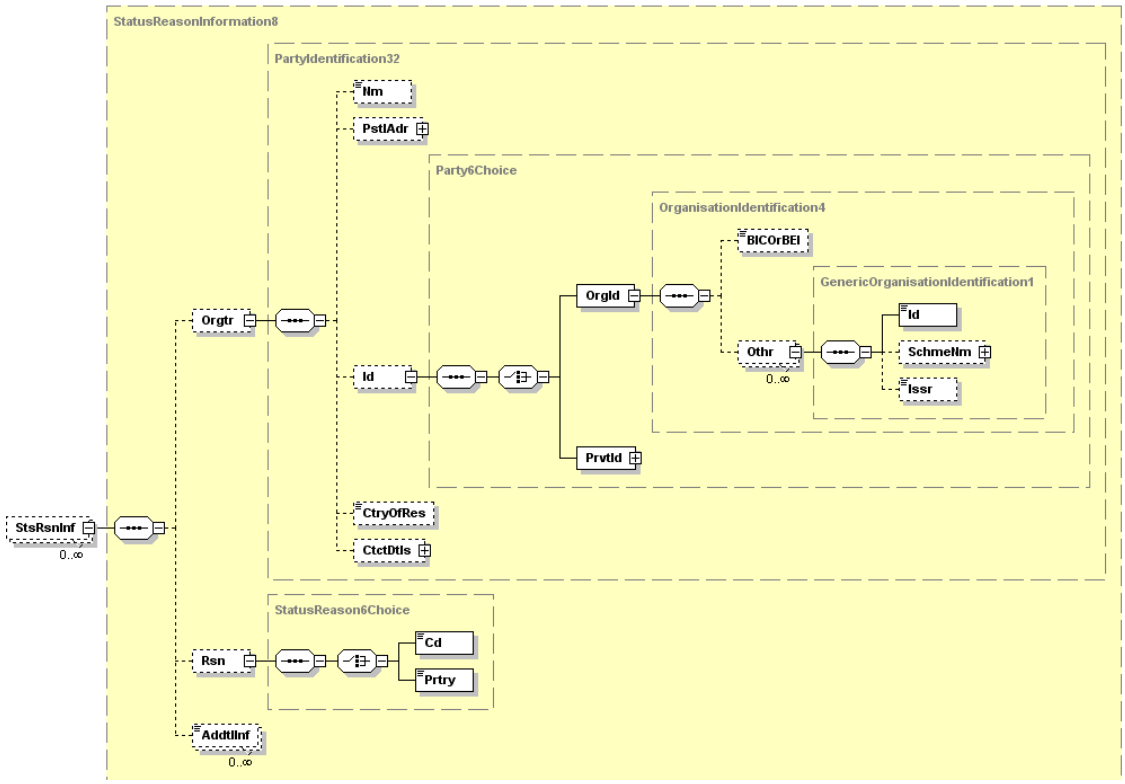
„Art der Aktion" *Ende-Kennzeichen*		
HAC (positives) Ende des Auftrags	Der Auftrag wurde auf EBICS-Ebene komplett abgearbeitet. Es werden keine weiteren (HAC-) Kundenprotokollschritte für diesen Auftrag erstellt	ORDER_HAC_FINAL_POS

HAC (negatives) Ende des Auftrags	Kein weiterer (HAC-) Kundenprotokollschritt wird für diesen Auftrag erstellt. Der Auftrag konnte auf EBICS-Ebene jedoch aus verschiedenen Gründen nicht komplett abgearbeitet werden (siehe Codes für die Ergebnisse der Aktion der vorhergehenden HAC-Protokollschritte dieses Auftrags).	ORDER_HAC_FINAL_NEG

10.2.3.2 Ergebnis der Aktion

Das Ergebnis der Aktion wird in der Elementgruppe Status Reason Information hinterlegt.

Elementgruppe <StsRsnInf> - Überblick:



Festlegungen für die Belegung:
Alle Elemente in <StsRsnInf>, die in dieser Liste nicht erwähnt werden, werden nie in HAC verwendet!

Name	XML Tag	Regeln für HAC
Originator / Name	<Orgtr><Nm>	Name des Kunden
Originator / Identification / OrganisationIdentification /Other	<Orgtr><Id><OrgId><Othr>	Elementgruppe (kommt 0 bis N-mal vor) für verschiedene Identifikationscodes mit folgender Bedeutung:
	1) <Id>	Identifikations-ID
	2) <SchmeNm><Prtry>	Über das Kennzeichen in <SchmeNm><Prtry> ist die Art von ID im Element <Id> zu identifizieren (die meisten der Kennzeichen sind bereits für EBICS definiert; in diesem Fall werden sie auch in dieser Notation verwendet): ➢ UserID (Teilnehmer-ID) ➢ PartnerID (Kunden-ID) ➢ SystemID (ID des technischen Teilnehmers) ➢ OrderID (Auftragsnummer) ➢ OrderType (Auftragsart) ➢ FileFormat (wird nur verwendet, wenn die Auftragsart FUL oder FDL ist) ➢ OrderIDRef (Wenn die Aktion sich auf einen anderen Auftrag bezieht, wird die Auftragsnummer des referenzierten Auftrags hiermit geliefert) ➢ OrderTypeRef (Wenn die Aktion sich auf einen anderen Auftrag bezieht, wird die Auftragsart des referenzierten Auftrags hiermit geliefert) ➢ PartnerIDRef (Wenn die Aktion sich auf einen anderen Auftrag bezieht, der ZUDEM einem anderen Kunden zugeordnet ist, wird dessen Kunden-ID hiermit geliefert) ➢ TimeStamp (Timestamp der Aktion im ISO 8601 Format, analog Kapitel 2.3 der EBICS-Spezifikation) ➢ DataDigest (Hashwert) Hinweis: Alle IDs, die hier beschrieben sind, sind in HAC zu liefern, wenn die entsprechenden Daten auf dem Bankrechner verfügbar sind.

Name	XML Tag	Regeln für HAC
Reason	<Rsn>	Ergebnis der Aktion, Tabelle siehe Kapitel 10.3. Alle möglichen Ergebnisse der Aktion sind als ISO Reasoncodes definiert (zu belegen in <Cd>). Dieses Element ist für HAC verpflichtend, es sei denn die Art der Aktion <OrgnlPmtInfAndSts><OrgnlPmtInfId> enthält 1) eines der beiden Ende-Kennzeichen (siehe Kapitel 10.2.3.1). In diesem Fall ist die Belegung von <Cd> nicht zulässig. 2) „ADDITIONAL“. In diesem Fall ist die Belegung optional
AdditionalInformation	<AddtlInf>	Wenn eine Dateianzeige erfolgt (Anzeige eines Auszugs/Zusammenfassung des Dateiinhalts) kommt dieses Element 1 bis n-mal vor (siehe Kapitel 10.2.3.3) Weiterer freier Text (Max105Text) ist immer zulässig. Die Nutzung ist optional und beliebig of wiederholbar (z.B. Nutzung für Freitextinformationen für den Kunden)

10.2.3.3 Dateianzeige (Verwendung in Deutschland)

Die Daten für die Dateianzeige (Informationen über den Dateiinhalt) wird in <StsRsnInf><AddtlInf> bereitgestellt.
Die Dateianzeige wird mit einem der beiden möglichen Ende-Kennzeichen geliefert (entweder ORDER_HAC_FINAL_POS oder ORDER_HAC_FINAL_NEG; siehe Kapitel 10.2.3.1).
Hinweis: Dies findet auch bei Dateien mit Dateiattributen ”DZHNN” Anwendung, was ja bedeutet, dass die Datei nicht per EU sondern per Datenträgerbegleitzettel autorisiert wird.
Die Eigenart der bestehenden Dateianzeige kann weiterverwendet werden: Pro Zeile der bekannten Dateianzeige gemäß PTK kommt das Element <AddtlInf> einmal vor.

Name	XML Tag	Rules for HAC
StatusReasonInformation / AdditionalInformation	<StsRsnInf><AddtlInf>	Wir nur für bestimmte Daten in Freitextform verwendet: 1. Beispiele für übliche Formate siehe Kapitel 10.2.3.3.1, 10.2.3.3.2, 10.2.3.3.3 und 10.2.3.3.4 2. Hashwert (nur benötigt für die SEPA Containerdatei) 3. Dateianzeige für Dateien ohne besonderes Format

10.2.3.3.1 Beispiel für DATAUS (deutsches Inlandsformat)

….
<StsRsnInf>
<AddtlInf>G U T S C H R I F T E N</AddtlInf>
<AddtlInf>Bank-Code : 30040000</AddtlInf>
<AddtlInf>Kontonummer : 0822511260</AddtlInf>

<AddtlInf>Auftraggeber : Bank-Verlag</AddtlInf>
<AddtlInf>Erstellungsdatum : 10.05.00</AddtlInf>
<AddtlInf>Anzahl der Zahlungssaetze : 1</AddtlInf>
<AddtlInf>Summe der Betraege (EUR) : 68.672,00</AddtlInf>
<AddtlInf>Summe der Kontonummern : 00000000001234567</AddtlInf>
<AddtlInf>Summe der Bank-Codes : 00000000007654321</AddtlInf>
<AddtlInf>Ausfuehrungstermin : 10.05.2000</AddtlInf>
</StsRsnInf>
….

10.2.3.3.2 Beispiel für SEPA

Dateianzeige in HAC für eine pain.001-Nachricht mit drei Payment Information Blöcken:

<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>
….

10.2.3.3.3 Beispiel für den SEPA-Container

Dateianzeige in HAC für einen Container mit zwei pain.001-Nachrichten:

<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>
….

10.2.3.3.4 Beispiel für DTAZV (deutsches Format, genutzt für internationale Zahlungen)

….
<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>Betrag : 20.000,000
<AddtlInf>Anzahl der Datensaetze T : 000000000000001</AddtlInf>
<AddtlInf>Summe der Betraege : 000000000020000</AddtlInf>
</StsRsnInf>
….

10.3 Anhang für HAC: External Reason Codes (Ergebnis der Aktion)

Die folgenden Ergebnisse der Aktion sind im Element <Rsn><Cd> zu protokollieren.
Sie sind ein Teil der externen ISO Codeliste “ExternalStatusReason1Code”:

ISO Code	ISO Name	Definition
AM21	LimitExceeded	Zwischen Kunde und Bank vereinbarter Höchstbetrag überschritten
DS01	ElectronicSignaturesCorrect	Elektronische Unterschrift(en) korrekt
DS02	OrderCancelled	Ein berechtigter Teilnehmer hat den Auftrag storniert
DS03	OrderNotCancelled	Der Versuch des Teilnehmers, den Auftrag zu stornieren, war nicht erfolgreich
DS04	OrderRejected	Der Auftrag wurde bankseitig abgewiesen (aus fachlichen Gründen)
DS05	OrderForwardedForPostprocessing	Der Auftrag war korrekt und konnte zur Weiterverarbeitung weitergegeben werden
DS06	TransferOrder	Der Auftrag wurde an die VEU weitergeleitet
DS07	ProcessingOK	Alle den Auftrag betreffenden Aktionen konnten durch den Bankrechner durchgeführt werden
DS08	DecompressionError	Die Dekomprimierung war nicht erfolgreich
DS09	DecryptionError	Die Entschlüsselung war nicht erfolgreich
DS10	Signer1CertificateRevoked	Das Zertifikat des ersten Unterzeichners wurde widerrufen
DS11	Signer1CertificateNotValid	Das Zertifikat des ersten Unterzeichners ist nicht gültig (widerrufen oder nicht aktiviert)
DS12	IncorrectSigner1Certificate	Für den ersten Unterzeichner ist kein Zertifikat vorhanden
DS13	SignerCertificationAuthority Signer1NotValid	Die CA des Zertifikates des ersten Unterzeichners ist unbekannt/ungültig
DS14	UserDoesNotExist	Der Teilnehmer ist auf dem Bankrechner unbekannt
DS15	IdenticalSignatureFound	Eine identische Unterschrift wurde bereits zur Bank geschickt
DS16	PublicKeyVersionIncorrect	Falsche Public Key Version. Dieser Code wird zurückgegeben, wenn ein Kunde nach dem Umstieg von einer älteren Programmversion (altes EU-Format) auf eine neue Programmversion (neues EU-Format) Unterschriftsdateien an das Kreditinstitut schickt, ohne sich vorher neu initialisiert beziehungsweise eine Public Key-Änderung durchgeführt zu haben.
DS17	DifferentOrderDataInSignatures	Auftragsdaten passen nicht zu der/den Unterschrift(en)
DS18	RepeatOrder	Datei nicht prüfbar, der gesamte Auftrag ist zu wiederholen. Dieser Code wird ausgegeben, wenn eine Betriebsstörung bei der Unterschriftsprüfung auftritt, z.B. zu wenig Speicherplatz
DS19	ElectronicSignatureRightsInsufficient	Die Berechtigungen des Teilnehmers (seine Unterschrift betreffend) sind für eine Ausführung des Auftrags unzureichend

ISO Code	ISO Name	Definition
DS20	Signer2CertificateRevoked	Das Zertifikat des zweiten Unterzeichners wurde widerrufen
DS21	Signer2CertificateNotValid	Das Zertifikat des zweiten Unterzeichners ist nicht gültig (widerrufen oder nicht aktiviert)
DS22	IncorrectSigner2Certificate	Für den zweiten Unterzeichner ist kein Zertifikat vorhanden
DS23	SignerCertificationAuthority Signer2NotValid	Die CA des Zertifikates des zweiten Unterzeichners ist unbekannt/ungültig
DS24	WaitingTimeExpired	Wartezeit bei unvollständigem Auftrag abgelaufen
DS25	OrderFileDeleted	Die Auftragsdatei wurde durch die Bank gelöscht (verschiedene Gründe möglich)
DS26	UserSignedMultipleTimes	Derselbe Teilnehmer hat mehrfach unterschrieben
DS27	UserNotYetActivated	Der Teilnehmer ist noch nicht aktiviert (technisch)
DS0A	DataSignRequested	Unterschriftsdaten werden benötigt *In EBICS bedeutet dies, dass die EU(s) noch nicht zum Bankrechner geschickt wurden oder dass die Anzahl der Unterschriften nicht ausreicht*
DS0B	UnknownDataSignFormat	Die Unterschrift ist für das Format nicht verfügbar oder ungültig *In EBICS bedeutet dies, dass die EU(s) nicht korrekt sind*
DS0C	SignerCertificateRevoked	Das Zertifikat des Unterzeichners wurde widerrufen *In EBICS bedeutet dies auch, dass der Teilnehmer gesperrt ist*
DS0D	SignerCertificateNotValid	Das Zertifikat des Unterzeichners ist ungültig (widerrufen oder nicht aktiviert). *In EBICS bedeutet dies, dass der öffentliche Schlüssel noch nicht aktiviert wurde oder das Zertifikat ungültig ist*
DS0E	IncorrectSignerCertificate	Das Zertifikat des Unterzeichners liegt nicht vor. *In EBICS bedeutet dies, dass der öffentliche Schlüssel nicht existiert oder kein Zertifikat vorhanden ist*
DS0F	SignerCertificationAuthority SignerNotValid	The authority of the signer certification sending the certificate is unknown
DS0G	NotAllowedPayment	Unterzeichner ist nicht berechtigt, diesen Vorgang zu unterschreiben *In EBICS bedeutet dies, dass der Teilnehmer (für dies) keine Autorisierungsrechte hat*
DS0H	NotAllowedAccount	Der Unterzeichner hat keine Berechtigung für dieses Konto
ID01	CorrespondingOriginalFileStillNotSent	Unterschriftsdatei wurde zur Bank übertragen, die dazugehörige Originaldatei jedoch noch nicht
TA01	TransmissonAborted	Die Übertragung der Datei war nicht erfolgreich – sie musste abgebrochen werden (technische Gründe)
TD01	NoDataAvailable	Keine Daten zur Abholung vorhanden
TD02	FileNonReadable	Die Datei kann nicht gelesen werden (z.B. unbekanntes Format)
TD03	IncorrectFileStructure	Das Dateiformat ist unvollständig oder falsch
TS01	TransmissionSuccessful	Die Übertragung der Datei war erfolgreich
TS04	TransferToSignByHand	Der Auftrag wurde zur Freigabe mittels Begleitzettel weitergereicht

10.4 Anhang für HAC: Art und Ergebnis der Aktion (zulässige Paare)

Falls mehr als ein Ergebniscode passend ist, sollte der präziseste gewählt werden.

<OrgnlPmtInfId> (Art der Aktion)	**Mögliche / Zulässige Werte für <Rsn><Cd> (Ergebnis der Aktion)**	**Beschreibung des Codes (gekürzt, mehr Details siehe Kapitel 10.3)**
FILE_UPLOAD	TS01 TA01 DS0C DS08 DS09	Upload erfolgreich Upload abgebrochen Teilnehmer/Zertifikat gesperrt Fehler bei Dekompremierung Fehler bei Entschlüsselung
FILE_DOWNLOAD	TS01 TA01 DS0C DS08 DS09 TD01	Download erfolgreich Download abgebrochen Teilnehmer/Zertifikat gesperrt Fehler bei Dekompremierung Fehler bei Entschlüsselung Keine Daten für Download vorhanden
ES_UPLOAD	TS01 TA01 DS0C DS08 DS09 ID01	Upload (der EU) erfolgreich Upload (der EU) abgebrochen User gesperrt/Zertifikat widerrufen Fehler bei Dekompremierung Fehler bei Entschlüsselung Originaldatei wurde noch nicht übertragen
ES_DOWNLOAD		*Wird in EBICS 2.5 noch nichtverwendet*

<OrgnlPmtInfId> (Art der Aktion)	Mögliche / Zulässige Werte für <Rsn><Cd> (Ergebnis der Aktion)	Beschreibung des Codes (gekürzt, mehr Details siehe Kapitel 10.3)
ES_VERIFICATION	AM21	Betragsgrenze überschritten
	TD02	Datei nicht lesbar
	TD03	Dateiformat ungültig
	TS04	Datei mit Attributen “DZHNN” (keine EU)
	DS01	EU(s) sind korrekt
	DS0A	Anzahl der EU(s) reicht nicht aus
	DS0B	EU(s) sind nicht korrekt
	DS0C	Zertifikat widerrufen/Teilnehmer gesperrt
	DS0D	Zertifikat ungültig/Public key nicht aktiviert
	DS0E	Zertifikat/Public key existiert nicht
	DS0F	CA für Zertifikat unbekannt
	DS0G	Unterzeichner hierzu nicht berechtigt
	DS0H	Unterzeichner nicht berechtigt für Konto
	DS10 (DS11; DS12)	Zertifikat widerrufen (ungültig; nicht vorhanden) für ersten Unterzeichner
	DS20 (DS21; DS22)	Zertifikat widerrufen (ungültig; nicht vorhanden) für zweiten Unterzeichner
	DS13/ DS23	CA unbekannt für Zertifikat des ersten/zweiten Unterzeichners
	DS14	Teilnehmer unbekannt auf Server
	DS15	The same ES already has been sent to bank
	DS16	Public key version ist nicht korrekt
	DS17	Auftragsdaten und EU passen nicht
	DS18	Auftrag wiederholen (Datei nicht testbar)
	DS19	EU-Berechtigung unzureichend
	DS24	Wartezeit überschritten, Datei gelöscht
	DS25	Datei von Bank gelöscht (versch. Gründe)
	DS26	Gleicher Teilnehmer mehrfach unterschrieben
	DS27	Teilnehmer noch nicht aktiviert
	DS08	Fehler bei Dekompremierung
	DS09	Fehler bei Entschlüsselung
VEU_FORWARDING	DS06	Auftrag zur VEU weitergeleitet

<OrgnlPmtInfId> (Art der Aktion)	Mögliche / Zulässige Werte für <Rsn><Cd> (Ergebnis der Aktion)	Beschreibung des Codes (gekürzt, mehr Details siehe Kapitel 10.3)
VEU_VERIFICATION	AM21 TD02 TD03 DS01 DS0B DS0C DS0D DS0E DS0F DS0G DS0H DS10 (DS11; DS12) DS20 (DS21; DS22) DS13/ DS23 DS14 DS15 DS16 DS17 DS18 DS19 DS24 DS25 DS26 DS27	Betragsgrenze überschritten Datei nicht lesbar Dateiformat ungültig EU(s) sind korrekt EU(s) sind nicht korrekt Zertifikat widerrufen/Teilnehmer gesperrt Zertifikat ungültig/Public key nicht aktiviert Zertifikat/Public key existiert nicht CA für Zertifikat unbekannt Unterzeichner hierzu nicht berechtigt Unterzeichner nicht berechtigt für Konto Zertifikat widerrufen (ungültig; nicht vorhanden) für ersten Unterzeichner Zertifikat widerrufen (ungültig; nicht vorhanden) für zweiten Unterzeichner CA unbekannt für Zertifikat des ersten/zweiten Unterzeichners Teilnehmer unbekannt auf Server The same ES already has been sent to bank Public key version ist nicht korrekt Auftragsdaten und EU passen nicht Auftrag wiederholen (Datei nicht testbar) EU-Berechtigung unzureichend Wartezeit überschritten, Datei gelöscht Datei von Bank gelöscht (versch. Gründe) Gleicher Teilnehmer mehrfach unterschrieben Teilnehmer noch nicht aktiviert
VEU_VERIFICATION_END	DS05	Auftrags war korrekt; wurde zur Weiterverarbeitung weiterlegeleitet
VEU_CANCEL_ORDER	DS02 DS03	Auftrag storniert Auftrag nicht storniert
ADDITIONAL	*Optional*	Hinweis: eine mögliche Belegung ist nicht um Bereich der EBICS-Spezifikation
ORDER_HAC_FINAL_POS	---	
ORDER_HAC_FINAL_NEG	---	

11 Anhang: Kryptographische Verfahren

In diesem Anhang werden die kryptographischen Verfahren beschrieben, die bei EBICS zum Einsatz kommen.

11.1 Authentifikationssignatur

11.1.1 Verfahren

Authentifikationssignaturen basieren auf dem RSA-Signaturverfahren. Die folgenden Parameter bestimmen das Verfahren zur Authentifikationssignatur: Länge des (geheimen) RSA-Schlüssels, Hashalgorithmus, Paddingverfahren, Kanonisierungsverfahren.

Für die Authentifikationssignatur definiert EBICS das **Verfahren „X002“** mit den folgenden Parametern:

Parameter	**Wert**
Schlüssellänge in KBit	>=1KBit (1024 Bit) und <=16KBit
Hashalgorithmus	SHA-256
Paddingverfahren	PKCS#1
Kanonisierungsverfahren	http://www.w3.org/TR/2001/REC-xml-c14n-20010315

Da die Mindestschlüssellänge bei X002 gegenüber X001 nicht geändert wurde, müssen die Authentifikationsschlüssel beim Umstieg von der Version X001 auf X002 nicht geändert werden.

Die optionalen XML-Signature-Felder „KeyInfo“ und „Object“ bleiben unbefüllt.
Ab EBICS 2.4 muss das Kundensystem in einen Request den Hashwert des öffentlichen Bankschlüssels X002 verwenden. Die Transaktion wird mit dem Returncode EBICS_INVALID_REQUEST_CONTENT abgebrochen, wenn noch X001 in einem Request verwendet wird.

11.1.2 Format

Authentifikationssignaturen werden in EBICS-Nachrichten gemäß der W3C-Recommendation „Signature Syntax and Processing“ (http://www.w3.org/TR/xmldsig-core/) dargestellt. Dadurch sind Identifikatoren der Algorithmen zur Bildung der Hashwerte, der Signatur, sowie der Identifikator des Kanonisierungsverfahrens Bestandteil der Authentifikationssignatur. Es ist somit keine Änderung der XML-Schnittstelle notwendig, wenn eine neue Version „X00n“ mit veränderten Parametern definiert wird. Das gilt insbesondere für Versionen, die als Hashfunktion SHA-224, SHA-256, SHA-384 oder SHA-512 einsetzen.

Bei der Einstellung der Authentifikationssignatur im Element `SignatureValue` erfolgt grundsätzlich kein Auffüllen auf die volle Länge des Modulo des RSA-Schlüssels zur Bildung dieser Signatur.

11.2 Elektronische Unterschriften

11.2.1 Verfahren

Elektronische Unterschriften basieren auf dem RSA-Signaturverfahren. Die Verfahren zur Erzeugung/ Überprüfung Elektronischer Unterschriften sind im Anhang (Kapitel 14) definiert. EBICS muss mindestens die Version „A004“ der bankfachlichen elektronischen Signatur unterstützen.

11.2.2 Format

Die XML-Schemadatei „ebics_orders.xsd“ enthält die Definition des globalen Elementes `BankSignatureData` für die Einbettung der elektronischen Unterschrift des Kreditinstituts (Da es sich hier um ein vorgesehenes Feature handelt, wurde diese Struktur noch nicht für Unterschriften in strukturierter Form fortgeschrieben, d.h. `BankSignatureData` enthält weiterhin ein Element `OrderSignature` zur Aufnahme einer Bank-EU in base64-Kodierung). (siehe auch Abbildung 4).

Die Schemadatei „ebics_signature.xsd“ enthält `UserSignatureData` für die Unterschrift der Teilnehmer in EBICS-Nachrichten. Dazu wird ein Instanzdokument zu „ebics_signature.xsd“ erstellt, das für Teilnehmer-EUs `UserSignatureData` als Toplevel-Element enthält. `UserSignatureData` enthält eine Liste von Elementen `OrderSignature` für eine oder mehrere Teilnehmer-EUs.
`UserSignatureData` enthält eine Liste von Elementen `OrderSignature` bzw. `OrderSignatureData` für eine oder mehrere Teilnehmer-EUs (siehe auch Abbildung 4).

Signaturverfahren A004:
Das Binärformat der EU eines Teilnehmers entspricht dem Format der Unterschriftsdatei gemäß Anhang (Kapitel 14.2.5.3) Das Attribut `PartnerID` von `OrderSignature` MUSS mit der Kunden-ID des jeweiligen Unterzeichners befüllt werden.
Das Binärformat der EU eines Kreditinstituts wird in Anlehnung an das Format der Unterschriftsdatei aus Kapitel 14.2.5.3 definiert.
Gegenüber der kundenseitigen EU-Datei wird lediglich das Feld „UserID“ durch das semantisch korrespondierende Feld „Host-ID“ ersetzt.

Signaturverfahren A005/A006:
Die EU des Teilnehmers in strukturierter Form ist im Anhang (Kapitel 14.1.6) dargestellt.

Das Instanzdokument wird in ZIP-komprimierter, hybrid verschlüsselter und base64-kodierter Form in die EBICS-XML-Struktur `ebicsRequest/body/DataTransfer/Signature` eingebettet.

11.2.3 EBICS-Autorisationsschemata für Unterschriftsklassen

EBICS legt die Autorisierungsschemata für Aufträge fest, die eine oder zwei bankfachliche EUs erfordern. Autorisierungsschemata für Aufträge, die mehr als zwei bankfachliche EUs erfordern, sind nicht im vorliegenden Standard beschrieben, obgleich es nicht verboten ist, mehr als zwei bankfachliche EU's zu übertragen..

E=Einzelunterschrift, A=Erstunterschrift, B=Zweitunterschrift, T=Transportunterschrift (nicht bankfachlich).

Autorisierungsschema für Aufträge mit Mindestanzahl EU = 0:

Die Mindestanzahl EU = 0 gilt für Aufträge, die über Begleitzettel autorisiert werden **("DZHNN")** oder für Aufträge des Schlüsselmanagement, für deren Autorisierung eine Transportunterschrift ausreicht (aber auch E-, A-, B-EU möglich sind). Aufträge die mit EU autorisiert werden ("OZHNN") müssen ausreichend unterschrieben werden. Da hierfür mindestens eine bankfachliche EU notwendig ist gelten hier auch die Regeln für Mindestanzahl EU = 1.

E	A	B	T
☑	☑	☑	☑

Autorisationsschema für Aufträge mit Mindestanzahl EU = 1:

- Autorisierung mit einer einzigen bankfachlichen EU:
  Die Autorisierung des Auftrags mit einer einzigen EU kann mit einer Einzelunterschrift erfolgen.

E	A	B	T
☑	☐	☐	☐

- Autorisierung mit zwei bankfachlichen EUs:
  Die Autorisierung des Auftrags kann auch mit 2 EUs der Klasse E, A oder B erfolgen, wenn mindestens eine der beiden eine Einzel- oder Erstunterschrift ist.

erste EU → / zweite EU ↓	E	A	B	T
E	☑	☑	☑	☐
A	☑	☑	☑	☐
B	☑	☑	☐	☐
T	☐	☐	☐	☐

Autorisationsschema für Aufträge mit Mindestanzahl EU = 2:

- Mit Ausnahme der Kombination bestehend aus zwei Zweitunterschriften ist die Autorisierung des Auftrags mit einer beliebigen Kombination aus zwei bankfachlichen EUs möglich.

erste EU → / zweite EU ↓	E	A	B	T
E	☑	☑	☑	☐
A	☑	☑	☑	☐
B	☑	☑	☐	☐
T	☐	☐	☐	☐

Allgemein gilt:

- Es ist keine Maximalanzahl von EUs definiert, jedoch müssen die Unterschriften im Falle einer Übertragung von mehr als zwei EUs den Regeln der oben genannten Autorisationsschemata genügen.
- Einzelunterschriften sind zur Autorisation grundsätzlich zulässig, jedoch nur bei Aufträgen mit Mindestanzahl EU = 0 oder EU = 1 ausreichend
- Eine Transportunterschrift autorisiert niemals die Ausführung eines Auftrags, sondern ermöglicht lediglich die Auftragseinreichung
- Die Reihenfolge, in der die Unterschriften geleistet werden, ist irrelevant
- Die bankfachlichen EUs eines Auftrags MÜSSEN von unterschiedlichen Teilnehmern (eventuell auch unterschiedlicher Kunden) geleistet werden.

11.3 Verschlüsselung

11.3.1 Verschlüsselung auf TLS-Ebene

11.3.1.1 Verfahren

Kundensystem und Banksystem MÜSSEN sich auf eines der folgenden Verfahren (sog. „Cyphersuites“, siehe RfCs 2246 und 3268) im Rahmen des TLS-Handshake einigen (priorisiert gemäß dieser Reihenfolge):

- TLS_RSA_WITH_AES_256_CBC_SHA: TLS mit Zertifikatstyp/Schlüsselaustauschverfahren RSA, Verschlüsselungsverfahren AES (Schlüssellänge 256bit, CBC-Modus) und Hashverfahren SHA-1
- TLS_RSA_WITH_AES_128_CBC_SHA: TLS mit Zertifikatstyp/Schlüsselaustauschverfahren RSA, Verschlüsselungsverfahren AES (Schlüssellänge 128bit, CBC-Modus) und Hashverfahren SHA-1
- TLS_RSA_WITH_3DES_EDE_CBC_SHA: TLS mit Zertifikatstyp/Schlüsselaustauschverfahren RSA, Verschlüsselungsverfahren 3-Key-Triple-DES (168bit effektive Schlüssellänge, aufgeteilt zu je 56bit Encryption-Decryption-Encryption, CBC-Modus) und Hashverfahren SHA-1.

Für die Übermittlung des Transaktionsschlüssels an den Kommunikationspartner definiert die „Spezifikation für die FTAM-Anbindung“ (Anlage 2 des DFÜ-Abkommens) eine eigene Datenstruktur, den sogenannten Vorsatz. Aus Kompatibilitätsgründen wird die Information aus dem Vorsatz in EBICS abgebildet. Der Vorsatz enthält neben dem asymmetrisch verschlüsselten symmetrischen Schlüssel auch den Hashwert des öffentlichen RSA-Schlüssels, der für die Verschlüsselung des symmetrischen Schlüssels verwendet wurde.

Das Verfahren TLS_RSA_WITH_3DES_EDE_CBC_SHA MUSS von jedem Kreditinstitut und von jedem Kundensystem unterstützt werden.

11.3.2 Verschlüsselung auf Anwendungsebene

11.3.2.1 Verfahren

Das Verfahren zur Verschlüsselung der Nutzdaten und EUs eines Auftrags ist die Hybridverschlüsselung basierend auf dem symmetrischen Verschlüsselungsverfahren 2-Key-Triple-DES und dem asymmetrischen Verschlüsselungsverfahren RSA.

Die Nutzdaten und EUs einer EBICS-Transaktion werden symmetrisch verschlüsselt. Je EBICS-Transaktion wird ein zufälliger symmetrischer Schlüssel (Transaktionsschlüssel) vom Sender der Nutzdaten und/oder EUs generiert, der dann sowohl für die Verschlüsselung der Nutzdaten als auch für die Verschlüsselung der EUs verwendet wird. Der symmetrische Schlüssel wird asymmetrisch verschlüsselt an den Empfänger übertragen.

Ausgehend vom Verschlüsselungsverfahren „V001“ (siehe Anhang, Kapitel 14.2.5.5), welches für das FTAM-Verfahren spezifiziert wurde, definiert EBICS das **Verschlüsselungsverfahren „E002“** mit den folgenden Merkmalen:

- Symmetrischer Verschlüsselungsalgorithmus (siehe Anhang, Kapitel 14.2.5.5)
  - Generierung des Transaktionsschlüssels (siehe Anhang, Kapitel 14.2.5.5)
  - AES-128 (Schlüssellänge 128 Bit) im CBC Modus
  - ICV (Initial Chaining Value) = 0
  - Paddingverfahren gemäß ANSI X9.23 / ISO 10126-2.
- RSA-Verschlüsselung des Transaktionsschlüssels, Schlüssellänge >=1KBit (1024 Bit) und <=16KBit
  - Unterschied zu V001: 768
- Paddingverfahren für die RSA-Verschlüsselung: PKCS#1
  - Unterschied zu V001: 0-Padding.

Da die Mindestschlüssellänge bei E002 gegenüber E001 nicht geändert wurde, müssen die Authentifikationsschlüssel beim Umstieg von der Version E001 auf E002 nicht geändert werden.

Das **Verfahren zur asymmetrischen Verschlüsselung** des Transaktionsschlüssels muss für EBICS wie folgt angepasst werden:

- Die Mindestlänge des (geheimen) RSA-Schlüssels beträgt 1024
- Das Paddingverfahren ist PKCS#1-konform.

Diese Anpassungen bedeuten konkret:

- Die Länge von PDEK ist gleich der Länge des verwendeten RSA-Schlüssels (>= 1024)

- PDEK wird aus DEK mittels PKCS#1 Padding erzeugt
- EDEK ist das Ergebnis der RSA-Verschlüsselung von PDEK.

Analog muss für EBICS auch das **Verfahren zur Entschlüsselung** des Transaktionsschlüssels angepasst werden:

- PDEK ist das Ergebnis der RSA-Entschlüsselung von EDEK
- Die 128 niedrigstwertigen Bits von PDEK bilden den geheimen Schlüssel DEK.

Für die **Bildung dieses Hashwertes des öffentlichen RSA-Schlüssels** wird im Kontext von „E002“ das Verfahren SHA-256 verwendet.

11.3.2.2 Formate

Die komprimierten und verschlüsselten EUs und Nutzdatensegmente werden als base64-kodierte Binärdaten in die EBICS-Nachrichten eingebettet.

Innerhalb der EBICS-Nachrichten erfolgt die Übermittlung des asymmetrisch verschlüsselten Transaktionsschlüssels innerhalb eines XML-Elements vom Typ `DataEncryptionInfoType`. Dieser Typ ist in der XML-Schema-Definitionsdatei ebics_types_H004.xsd definiert und seine graphische Darstellung ist in Abbildung 104 enthalten.

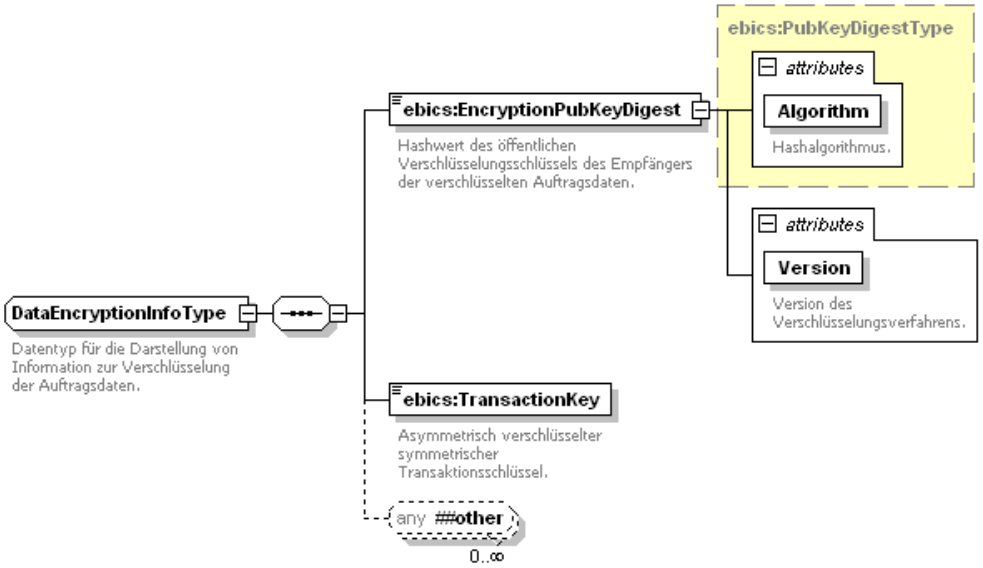


Abbildung 104: Definition des XML-Schema Typs DataEncryptionInfoType

Das Element `ebicsRequest/body/DataTransfer/DataEncryptionInfo` bzw. `ebicsReponse/body/DataTransfer/DataEncryptionInfo` vom Typ `DataEncryptionInfoType` ist Bestandteil des ersten EBICS-Requests einer Upload-Transaktion (vgl. ebics_request_H004.xsd) oder der ersten EBICS-Response einer Download-Transaktion (vgl. ebics_response_H004.xsd).

Im Unterschied zum Vorsatz enthält `DataEncryptionInfoType` keine Teilnehmerangaben. Das ist nicht notwendig, da der Empfänger/ Sender der Auftragsdaten stets der Einreicher des Auftrags ist. Die Teilnehmer-/KundenID des Einreichers ist bereits Bestandteil der Steuerdaten des ersten EBICS-Requests einer jeden EBICS-Transaktion und ist der EBICS-Transaktion fest zugeordnet.

Das Element `EncryptionPubKeyDigest` enthält neben dem Hashwert des öffentlichen RSA-Schlüssels auch die Version des verwendeten Verschlüsselungsverfahrens und den Identifikator des verwendeten Hashalgorithmus.

Es ist keine Änderung von `DataEncryptionInfoType` notwendig, wenn eine neue Version „E00n“ mit veränderten Parametern definiert wird. Das gilt insbesondere für Versionen, die als eine oder mehrere der Hashfunktionen SHA-224, SHA-256, SHA-384, SHA-512 zulassen.

Bei der Einstellung des verschlüsselten Transaktionsschlüssels im Element `TransactionKey` erfolgt grundsätzlich kein Auffüllen auf die volle Länge des Modulo des RSA-Schlüssels für die Verschlüsselung.

11.4 Replay-Vermeidung mittels Nonce und Timestamp

11.4.1 Verfahrensbeschreibung

Der erste EBICS-Request, der der Initialisierung einer EBICS-Transaktion dient, enthält die Elemente „**Nonce**“ und „**Timestamp**“, die gemeinsam ein Wiedereinspielen dieses Requests verhindern sollen.

„Nonce“ und „Timestamp“ bilden eine funktionale Einheit zur Replay-Vermeidung:

1. Das Kundensystem generiert einen zufälligen „Nonce“ und setzt „Timestamp“ auf den aktuellen Zeitpunkt des Nachrichtenversands.
2. Das Banksystem vergleicht den empfangenen „Nonce“ mit einer lokal gespeicherten Liste zuvor empfangener „Nonce“-Werte. Außerdem prüft es die Abweichung des „Timestamp“ zur aktuellen Uhrzeit. Ist der gerade empfangene „Nonce“ in der gespeicherten Liste vorhanden oder ist die Abweichung des „Timestamp“ größer als ein vom Kreditinstitut festzulegender Toleranzzeit-

raum, so wird der Request mit dem technischen Fehlercode EBICS_TX_MESSAGE_REPLAY beantwortet.

3. Verlief die „Nonce“- und „Timestamp“-Prüfung ohne Fehler, so speichert das Banksystem das „Nonce“-„Timestamp“-Paar in der lokalen Liste ab und fährt mit der weiteren Verarbeitung der Nachricht fort.

Das Banksystem kann „Nonce“-„Timestamp“-Paare, deren Zeitstempel außerhalb des Toleranzzeitraums liegen, aus seiner Liste löschen: Nachrichten, die ein solches Paar enthielten, würden bereits wegen der überhöhten Abweichung des „Timestamp“ abgewiesen. Der festgelegte Toleranzzeitraum gilt also gleichermaßen für die Prüfung neuer Paare wie für den Löschvorgang gespeicherter Paare.

Mit den Elementen „Nonce“ und „Timestamp“ garantiert dieses Verfahren, dass der erste EBICS-Request einer Transaktion eindeutig ist. Dadurch wird eine bankseitige Initialisierung neuer EBICS-Transaktionen aufgrund alter, erneut eingespielter Nachrichten verhindert. Gleichzeitig begrenzt „Timestamp“ die zeitliche Notwendigkeit zur bankseitigen „Nonce“-Speicherung.

11.4.2 Aktionen des Kundensystems

11.4.2.1 Erzeugung von „Nonce“ und „Timestamp“

Das Kundensystem MUSS in der Transaktionsphase „Initialisation“ folgende Felder befüllen:

- `ebicsRequest/header/static/Nonce` mit einer kryptographisch starken Zufallszahl der Länge 128 Bit
- `ebicsRequest/header/static/Timestamp` mit dem aktuellen Zeitstempel des Versands des EBICS-Requests (Datum und Uhrzeit gemäß ISO 8601).

Ein Beispiel für eine syntaktisch korrekte Befüllung der Werte „Nonce“ und „Timestamp“ gibt folgender XML-Ausschnitt wieder:

```
<?xml version="1.0" encoding="UTF-8"?>
<ebicsRequest
 xmlns="urn:org:ebics:H004"
 xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:org:ebics:H004 ebics_request_H004.xsd "
 Version="H004" Revision="1">
 <header authenticate="true">
   <HostID>EBIXHOST<HostID>
   <Nonce>01A56FF768B3B36C5120E9904A7FB035</Nonce>
   <Timestamp>2005-06-22T17:07:34.123+02:00</Timestamp>
   […]
 </header>
 […]
</ebicsRequest>
```

Weitere Informationen zur korrekten Befüllung der beiden XML-Schema-Elemente sind unter http://www.w3.org/TR/xmlschema-2/#hexBinary (hexBinary) bzw.
http://www.w3.org/TR/xmlschema-2/#dateTime (dateTime)
zu finden.

11.4.2.2 Verhalten bei Fehler-Rückmeldung EBICS_TX_MESSAGE_REPLAY

Mit dem technischen Fehlercode EBICS_TX_MESSAGE_REPLAY signalisiert das Banksystem, dass die gerade vom Client versandte EBICS-Nachricht einen „Nonce“-Wert enthält, der mit einem banksystemseitig gespeicherten übereinstimmt, oder dass der „Timestamp“ außerhalb des Toleranzzeitraums liegt.

Bei Verwendung kryptographisch starker Zufallszahlen als „Nonce“ und bei vernünftig gewähltem Toleranzzeitraum durch das Kreditinstitut (Richtwert: einige Stunden) kann der Fall einer zufälligen Kollision aufgrund der verschwindend geringen Wahrscheinlichkeit seines Auftretens vernachlässigt werden.

Das Kundensystem muss also bei Empfang der Meldung EBICS_TX_MESSAGE_REPLAY die Möglichkeit eines Replay-Angriffs, einer intolerabel ungenauen Uhreneinstellung auf Kunden- oder Banksystemseite oder einer Fehlfunktion der eigenen Transaktionsverwaltung bei der Vergabe von „Nonce“-Werten in Betracht ziehen.
Möchte der Teilnehmer die betroffene EBICS-Nachricht trotzdem erfolgreich absetzen, so muss er zuvor die Felder `ebicsRequest/header/static/Nonce` und `ebicsRequest/header/static/Timestamp` entsprechend Kapitel 11.4.2.1 neu erzeugen. Die restlichen Inhalte können unverändert bleiben.

11.4.3 Aktionen des Banksystems

11.4.3.1 Überprüfung von „Nonce“ und „Timestamp“

Wenn das Banksystem eine initiale EBICS-Nachricht eines Teilnehmers erhält, so MUSS es zur Prüfung auf einen Nachrichten-Replay die folgenden Aktionen durchführen. Können diese Prüfungen sämtlich erfolgreich durchlaufen werden, so liegt kein Nachrichten-Replay vor:

1. **Abgleich des empfangenen „Timestamp“ mit dem lokalen Zeitstempel:** Normiert auf UTC muss sich der empfangene „Timestamp“ innerhalb des Toleranzzeitraums befinden, der um den aktuellen Zeitstempel des Banksystems herum aufgespannt ist. Mit diesem Toleranzzeitraum sollen Differenzen der Ganggenauigkeit der beteiligten Uhren zwischen den Systemen und eventuell auch verfrühte/verspätete Sommer-/Winterzeitumstellungen abgefangen werden. Gleichzeitig bestimmt der Toleranzzeitraum auch, wann das Banksystem gespeicherte „Nonce“-„Timestamp“-Paare löschen kann: Ankommende Nachrichten mit einem „Timestamp“ außerhalb des Toleranz-

zeitraums werden nicht akzeptiert. In der Vergangenheit gespeicherte „Nonce“-„Timestamp“-Paare, deren „Timestamp“ sich mittlerweile außerhalb des Toleranzzeitraums befindet können deshalb gelöscht werden.
Der Toleranzzeitraum muss vom Banksystem einmalig festgelegt werden. Hierbei vergrößern hohe Werte (= große Toleranzzeiträume) den Speicherbedarf für valide „Nonce“-„Timestamp“-Paare, während niedrige Werte (= kleine Toleranzzeiträume) die Gefahr abgelehnter EBICS-Nachrichten in Folge zu großer Uhrendifferenzen zwischen Kunden- und Banksystem erhöhen.
Falls sich der empfangene „Timestamp“ nicht im Toleranzzeitraum befindet, besteht die Gefahr eines Nachrichten-Replays. Das Banksystem MUSS daher mit dem technischen Fehlercode EBICS_TX_MESSAGE_REPLAY antworten.

2. **Vergleich des empfangenen „Nonce“ mit den lokal gespeicherten „Nonce“-Werten:** Banksystemseitig sind sämtliche „Nonce“-„Timestamp“-Paare gespeichert, die aus validen EBICS-Requests im Toleranzzeitraum stammen. Falls der empfangene „Nonce“ mit einem gespeicherten „Nonce“ übereinstimmt, MUSS das Banksystem mit dem technischen Fehlercode EBICS_TX_MESSAGE_REPLAY antworten.

11.5 Initialisierungsbriefe

Initialisierungsbriefe für INI enthalten den öffentlichen bankfachlichen Teilnehmerschlüssel, Initialisierungsbriefe für HIA den öffentlichen Authentifikationsschlüssel des Teilnehmers sowie den öffentlichen Verschlüsselungsschlüssel des Teilnehmers.

11.5.1 Initialisierungsbrief für INI (Beispiel)

11.5.1.1 Mit Elektronischer Unterschrift der Version A004

Benutzername	Hans Mustermann
Datum	TT.MM.JJJJ
Uhrzeit	HH:MM:SS
Empfänger	DFÜ-Bank
User-ID	Xxxxxxxx
Kunden-ID	Yyyyyyyy
EU-Version	A004

Öffentlicher Schlüssel (Public-Key) für die Elektronische Unterschrift:

Exponent 1024

00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	01	00	01

Modulo 1024

FF	12	03	26	E6	30	90	A5	06	01	EF	16	10	21	EE	D4
77	23	27	A9	14	17	07	F1	71	25	22	D5	91	00	41	0A
D7	4A	2F	D5	6C	16	4E	C3	2D	82	F3	02	31	CD	FF	FB
45	77	E4	7E	E5	B2	CB	7B	9A	5F	75	7B	32	7C	16	E5
FB	16	41	0B	4A	39	0F	50	47	68	9C	9B	27	D2	A0	9C
CA	23	A8	C3	1C	AB	A5	ED	72	75	9D	0A	B8	9B	37	BA
00	CB	68	BB	AC	C8	D1	C8	D3	35	C8	BF	1F	A3	06	CF
24	5A	DC	EB	84	64	86	D0	97	8F	E4	67	08	78	81	07

Hash	D2	FD	56	F3	1E	5C	76	D2	B8	2C
	0B	1E	4C	6A	13	9E	85	87	E8	D3

Ich bestätige hiermit den obigen öffentlichen Schlüssel für meine Elektronische Unterschrift.

Ort / Datum Firma / Name Unterschrift

11.5.1.2 Mit Elektronischer Unterschrift der Version A005/A006

Benutzername	Hans Mustermann
Datum	TT.MM.JJJJ
Uhrzeit	HH:MM:SS
Empfänger	DFÜ-Bank
User-ID	Xxxxxxxx
Kunden-ID	Yyyyyyyy
EU-Version	A005

Öffentlicher Schlüssel (Public-Key) für die Elektronische Unterschrift:

Exponent 1024

00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	01	00	01

Modulo 1536

9B	86	4D	2E	72	9E	9E	03	94	78	EB	96	41	E6	27	C6
F2	98	B9	B5	4D	AC	B2	B8	99	C6	13	7C	6A	67	A3	93
56	B0	C0	E2	BB	22	D5	F1	4A	4E	3E	B5	E0	50	9A	41
6E	A5	95	8F	75	CF	A3	04	F9	BA	32	18	BF	ED	24	EC
B6	06	5E	62	80	42	F9	7A	C1	32	2C	F3	75	3F	D5	92
72	2C	A2	83	E8	B5	47	12	59	F6	4B	CD	A6	4E	D8	7F
7B	56	DA	D9	57	32	79	B4	7B	66	79	C9	F7	18	40	7E
CF	AC	5C	46	14	6A	B7	70	1D	47	D0	51	E7	81	62	2B
49	D7	09	5F	47	A4	4C	A3	3F	67	04	02	4B	40	3D	71
AA	5F	3E	A2	30	53	77	30	71	0A	96	DD	62	BE	6C	BF
40	27	28	0C	9F	FF	E0	6D	0A	8C	5E	E0	75	E2	30	A4
49	13	65	08	E5	A9	11	E3	7D	1C	FF	7F	B9	31	18	1F

SHA-256-Hashwert:

D4	7A	24	27	5C	5F	D8	0D
50	1B	CF	28	C5	38	FE	1F
51	DD	24	8B	3E	5C	72	D5
CD	47	9D	82	79	0C	EF	52

Ich bestätige hiermit den obigen öffentlichen Schlüssel für meine Elektronische Unterschrift.

________________ ____________________ ___________________

Ort / Datum Firma / Name Unterschrift

11.5.2 Initialisierungsbrief für HIA (Beispiel)

Benutzername	Hans Mustermann
Datum	TT.MM.JJJJ
Uhrzeit	HH:MM:SS
Empfänger	DFÜ-Bank
User-ID	Xxxxxxxx
Kunden-ID	Yyyyyyyy
Version Authentifikationssignatur	X002
Version Verschlüsselung	E002

Öffentlicher Authentifikationsschlüssel:

Exponent : 1024 Bit Länge

00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	01	00	01

Modulus: 1024 Bit Länge

B7	9D	3A	F0	68	15	AC	6E	AB	BF	F3	A1	D4	38	A3	D1
4D	D6	74	2C	CB	6D	00	52	D5	0C	A2	B0	BF	22	BD	08
8F	F4	5B	3E	B5	67	B5	F5	AE	D6	39	69	01	41	D0	69
8B	D5	F6	EA	03	F1	4B	59	56	84	DE	93	13	D8	07	FB
26	13	05	4B	04	F2	27	65	DA	26	51	35	48	50	64	B3
68	CA	7C	E7	FD	B0	12	34	CF	37	94	EE	CE	7B	B6	2D
79	73	09	82	0A	96	D9	13	75	26	D6	AC	19	40	F8	3E
6C	FD	A0	31	42	2C	F0	A4	EB	30	A0	69	08	A7	61	78
79	9A	67	25	DC	44	CB	66	39	30	11	9A	A5	13	CA	E7
84	53	1A	4C	27	AB	66	62	83	43	E1	B2	81	D6	70	83

SHA-256-Hashwert:

B8	3C	B0	19	66	C9	9C	6E
2C	A5	BA	6A	2B	56	01	92
35	2A	B4	91	53	E9	0B	BA
34	C1	5E	B5	9F	4A	64	F7

Öffentlicher Verschlüsselungsschlüssel:

Exponent : 1024 Bit Länge

00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	01	00	01

Modulus: 2048 Bit Länge

C1	C6	41	30	AF	7A	7C	4C	37	07	48	B0	BD	C1	FD	06
8B	56	06	52	CF	2C	88	9A	FB	24	03	99	A4	22	17	63
56	8D	EA	84	FE	53	40	2E	B1	D9	FF	3A	8E	DD	7A	F4
94	95	53	44	A3	D9	B5	26	60	EC	B1	09	FE	D9	70	F3
D9	6E	40	74	77	16	9B	85	1F	53	65	A3	45	2D	C6	97
5D	7F	9D	0B	22	D9	1B	3C	6F	C9	1B	7B	44	11	C2	69
F0	B8	2C	B6	53	BD	02	11	DB	FF	5D	B1	C4	A7	A7	6C
1B	25	CA	13	0D	46	A2	E7	31	E0	78	11	4A	07	DA	05
C7	CE	A2	C9	39	8C	AB	A7	0D	3B	42	8B	D7	30	9F	B2
A3	48	8A	62	A3	81	0B	FD	1A	8C	05	F7	6A	8D	43	6C
42	74	A1	F9	5A	74	03	63	EA	CB	BF	E9	35	83	60	93
59	B6	2C	95	99	2A	B7	44	32	9D	72	48	32	B6	01	5E
3F	D5	B8	2B	F8	5A	A2	2A	A1	DE	1F	C5	28	21	09	A0
13	93	62	64	4B	C4	82	E8	41	E7	9C	39	01	11	AB	36
40	4D	BF	6B	1B	A5	FB	E1	E6	F9	A3	5F	73	C8	29	37
EC	C7	2A	D7	89	A3	3B	62	9D	2F	B0	03	D7	6B	96	9D

SHA-256-Hashwert:

9D	2D	C0	AF	55	6E	D4	D9
04	00	BB	23	AF	C8	1B	AB
91	A3	7A	2E	97	A9	31	6D
D0	01	79	5F	C6	D0	CD	54

Ich bestätige hiermit die obigen öffentlichen Schlüssel.

____________________ ______________________________

Ort / Datum Unterschrift

11.6 Erzeugen der TransaktionsIDs

TransaktionsIDs sind kryptographisch starke Zufallszahlen der Länge 128 Bit. Dadurch wird erreicht, dass für zwei beliebige Banksysteme die Wahrscheinlichkeit, zu einem Zeitpunkt dieselbe TransaktionsID zu verwenden, hinreichend gering ist.

TransaktionsIDs werden von kryptographischen Pseudozufallszahlengeneratoren (PRNG) erzeugt, welche mit einer echten Zufallszahl (Seed) initialisiert wurden. Die Entropie des Seed sollte mindestens 100 Bit betragen.

12 Übersicht ausgewählter EBICS-Details

12.1 Optionale EBICS-Features

Bei EBICS sind nicht alle Funktionen als verpflichtend definiert. Für einige Auftragsarten sowie Features innerhalb des Transaktionsablaufs steht einem Kreditinstitut, das den EBICS-Standard implementiert, die Unterstützung frei.

12.1.1 Optionale Auftragsarten

Folgende EBICS-Auftragsarten KÖNNEN aus Sicht eines Kreditinstituts unterstützt werden (d.h. sie sind optional):

- HAA (abrufbare Auftragsarten abholen)
- HKD (Kunden- und Teilnehmerdaten des Kunden abholen)
- HTD (Kunden- und Teilnehmerdaten des Teilnehmers abholen)
- HSA (Teilnehmerinitialisierung für Teilnehmer mit Zugang zu DFÜ über FTAM).

12.1.2 Optionale Funktionalitäten im Transaktionsablauf

Folgende EBICS-Funktionalitäten KANN ein Kreditinstitut oder ein Kundenprodukt unterstützen (d.h. sie sind für beide Seiten optional):

- Vorabprüfung (siehe Kapitel 3.6 und 5.3)
- Recovery (siehe Kapitel 3.4 und 5.4).

12.2 EBICS-Bankparameter

Mit der EBICS-Auftragsart HPD (siehe auch Kapitel 9.2) kann sich der Teilnehmer über die spezifischen Zugangs- (`AccessParams`) und Protokollparameter (`ProtocolParams`) des Kreditinstituts informieren.

Zugangsparameter (`AccessParams`):

Parametername	#	Bedeutung	Beispiel
`URL`	1..∞	URL oder IP-Adresse für den elektronischen Zugang zum Kreditinstitut Es besteht die Möglichkeit, mehrere URLs anzugeben. Jede URL mit erreichtem valid_from-Datum (oder wenn dieses Feld leer ist) ist gültig. Ist eine URL nicht erreichbar, kann der Kunde eine andere gültige Adresse nutzen.	„http://www.die-bank.de“

URL@valid_from	0..1	Gültigkeitsbeginn der URL/IP. Falls nicht angegeben, ist die Angabe ab sofort gültig	„2005-01-30T15:30:45.123Z“
Institute	1	Bezeichnung des Kreditinstituts	„Die Bank“
HostID	0..1	ID des EBICS Banksystems	„bank01“

Protokollparameter (ProtocolParams):

Parametername	#	Bedeutung	abh. Auftrags-arten
Version	1	zulässige Versionen (jeweils als Auflistung) für EBICS-Protokoll (Protocol), Verschlüsselung (Encryption), Signatur (Signature) und Authentifikation (Authentication)	alle
Recovery	0..1	Unterstützung für das Wiederaufsetzen von Transaktionen	alle
PreValidation	0..1	Unterstützung der Vorabprüfung. Falls dieser Parameter gesetzt ist, sichert das Kreditinstitut damit zu, dass es zumindest einen Teil der vom Teilnehmer im Rahmen der Vorabprüfung übermittelten Daten prüft. Das Kreditinstitut ist aber nicht verpflichtet, die Daten in vollem Umfang zu prüfen	Uploads
X509Data	0..1	Unterstützung für X.509-Daten wie z.B. Zertifikate aus dem XML-Feld ebicsRequest/body/X509Data. Mit dem Attributflag @persistent kann das Kreditinstitut angeben, ob es die X.509-Daten der Teilnehmer im Zustand „Bereit“ persistent vorhält. In diesem Fall müsste der Teilnehmer sie dann nicht bei jeder Transaktionsinitialisierung erneut übermitteln. Falls nicht angegeben, unterstützt das Kreditinstitut die persistente X.509-Datenhaltung nicht	alle
ClientDataDownload	0..1	Unterstützung der Auftragsarten HKD (Download von Kundendaten, Kapitel 9.3) und HTD (Download von Teilnehmerdaten, Kapitel 9.4)	HKD, HTD
DownloadableOrder» Data	0..1	Unterstützung der Auftragsart HAA (abrufbare Auftragsarten abholen, Kapitel 9.1)	HAA

12.3 Auftragsattribute

Das Auftragsattribut (5 Bytes alphanummerisch) hat in EBICS folgende zulässige Belegungen:

Position	Bedeutung	Zulässige Werte
1	Art der übermittelten Daten	**O** = Auftragsdaten und EUs **U** = bankfachliche EUs **D** = Auftragsdaten und Transport-EU (**D** wird außerdem für HIA, INI, HPB verwendet)
2	Komprimierungsart für Auftragsdaten und/ oder EUs	**Z** = ZIP-Komprimierung
3	Verschlüsselungsart für Auftragsdaten und/ oder EUs	**N** = Keine Verschlüsselung **H** = Hybrid-Verfahren AES/RSA
4	*Reserve*	

5	*Reserve*	

Je nach Auftragsart und evtl. weiteren Rahmenbedingungen MUSS ein EBICS-Client die folgenden Auftragsattribute in die Steuerdaten des ersten EBICS-Requests einer EBICS-Transaktion eintragen (`ebicsRequest/header/static/OrderDetails/OrderAttribute`):

Auftragsart	Rahmenbedingung	Auftrags-attribute
INI	-	DZNNN
HIA	-	DZNNN
HSA	-	OZNNN
HPB	-	DZHNN
PUB	-	OZHNN
HCA	-	OZHNN
HCS	-	OZHNN
SPR	-	UZHNN
HVE	-	UZHNN
HVS	-	UZHNN
H3K	-	OZNNN
sonstige Upload-Auftragsarten	Auftragsdaten und EU(s)	OZHNN
sonstige Upload-Auftragsarten	nur bankfachliche EU(s), keine Auftragsdaten	UZHNN
sonstige Upload-Auftragsarten	Auftragsdaten mit Transport-unterschrift (Freigabe des Auftrags mit Begleitzettel anstelle bankfachlicher EU)	DZHNN
sonstige Download-Auftragsarten	Anforderung der Abholdaten mit bankfachlicher EU des Kreditinstituts	OZHNN
sonstige Download-Auftragsarten	Anforderung der Abholdaten ohne bankfachliche EU des Kreditinstituts	DZHNN

12.4 Sicherheitsmedien bankfachlicher Schlüssel

Für die Angabe des Sicherheitsmediums (geheimer) bankfachlicher Schlüssel definiert EBICS die folgenden Wertekategorien:

Sicherheitmedium	Belegung
Keine Angabe	0000
Diskette	01dd
Chipcard	02dd
Sonstiges Wechselmedium	03dd
Nicht wechselbares Medium	04dd

In der obigen Tabelle steht „dd“ für eine beliebige Zahlenkombination, die institutsindividuell festgelegt wird.

12.5 Muster für TeilnehmerIDs, KundenIDs, AuftragsIDs

Die nachfolgende Tabelle legt die in EBICS zulässigen Muster unterschiedlicher IDs fest. Für jede ID werden zudem all die XML-Typen aufgelistet, die in EBICS verwendet werden, um entsprechende IDs aufzunehmen.

ID	**TeilnehmerID / ID Technischer Teilnehmer**	**KundenID**	**AuftragsIDs**
Pattern	[a-zA-Z0-9,=]{1,35}	[a-zA-Z0-9,=]{1,35}	[A-Z]{1}[A-Z0-9]{3}
XML-Typ (XML-Schemadatei)	Beide vom Typ `UserIDType` (ebics_types_H004.xsd)	`PartnerIDType` (ebics_types_H004.xsd)	`OrderIDType` ebics_types_H004.xsd

Für die Bankrechnernummer HostID (XML-Typ `HostIDType` in der Schemadatei ebics_types_H004.xsd) ist kein Pattern definiert.

13 Anhang: Auftragsartenkennungen

Die Auftragsarten in der folgenden Tabelle werden in den Kapiteln 8.3, 9 und 10 ausführlich erläutert:

Kennung	Über-tragungs-richtung	Text	Optional / Verpflichtend zu unterstützen
FDL	D	Datei mit beliebigem Format abholen	Optional
FUL	U	Datei mit beliebigem Format senden	Optional
HAA	D	Abrufbare Auftragsarten abholen	Optional
HAC	D	Abholen Kundenprotokoll (XML-Format)	Verpflichtend
HCA	U	Änderung der Teilnehmerschlüssel für Authentifikation und Verschlüsselung senden	Verpflichtend
HCS	U	Änderung der Teilnehmerschlüssel für EU, Authentifikation und Verschlüsselung senden	Verpflichtend
HEV	D	Unterstützte EBICS-Versionen abrufen	Verpflichtend
HIA	U	Übermittlung der Teilnehmerschlüssel für Authentifikation und Verschlüsselung im Rahmen der Teilnehmerinitialisierung	Verpflichtend
HKD	D	Kunden- und Teilnehmerdaten des Kunden abholen	Optional
HPB	D	Transfer der öffentlichen Bankschlüssel (Abholen)	Verpflichtend
HPD	D	Bankparameter abholen	Verpflichtend
HSA	U	Übermittlung der Teilnehmerschlüssel für Authentifikation und Verschlüsselung im Rahmen der Teilnehmerinitialisierung für Teilnehmer, die über einen DFÜ-Zugang über FTAM verfügen	Optional
HTD	D	Kunden- und Teilnehmerdaten des Teilnehmers abholen	Optional
HVD	D	VEU-Status abrufen	Konditional[1]
HVE	U	VEU-Unterschrift hinzufügen	Konditional[1]
HVS	U	VEU-Storno	Konditional[1]
HVT	D	VEU-Transaktionsdetails abrufen	Konditional[1]
HVU	D	VEU-Übersicht abholen	Konditional[1]
HVZ	D	VEU-Übersicht mit Zusatzinformationen abholen	Konditional[1]
H3K	U	Übermittlung aller drei Teilnehmerschlüssel (für EU, Authentifikation und Verschlüsselung im Rahmen der Teilnehmerinitialisierung im Falle der Verwendung von Zertifikaten)	Optional

[1] Für deutsche Banken verpflichtend, zurzeit noch nicht von französischen Banken unterstützt.

Weitere Auftragsarten des Schlüsselmanagements:

Kennung	Über-tragungs-richtung	Text	Format
INI	U	Senden Passwort-Initialisierung	Public Key des Kunden für die EU (siehe Anhang Kapitel 14)
PTK	D	Abholen Kundenprotokoll	Format siehe Implementation Guide (chapter 4.3.1, verpflichtend für deutsche Banken)
PUB	U	Senden Public Key zur Unterschriftenverifizierung	Public Key des Kunden für die EU (siehe Anhang Kapitel 14)
SPR	U	Senden Sperre der Zugangsberechtigung	Übertragung einer EU-Datei mit der Signatur für eine Dummy-Datei, die nur ein Leerzeichen enthält (verpflichtend)

Weitere Auftragsartenkennungen (die sowohl vom Schlüsselmanagementprozess als auch vom EBICS-schema unabhängig sind) finden sich im Dokument „EBICS Anhang 2 – Auftragsartenkennungen“

14 Anhang: Signaturverfahren für die Elektronische Unterschrift

Die zum Einsatz kommenden Sicherheitsverfahren müssen die Elektronische Unterschrift für die zu übertragenden Daten leisten. Hierbei ist folgendes Anforderungsprofil zu erfüllen:

- Die Unterschrift darf nur vom Unterzeichner geleistet werden können, so dass der Unterzeichner die Unterschrift nicht leugnen kann bzw. dass beweisbar ist, dass der Ursprung eines eventuellen Missbrauchs nur im Verantwortungsbereich des Unterzeichners liegen kann.

- Jeder mögliche Empfänger muss die Echtheit der Unterschrift prüfen können, wobei zusätzlich gewährleistet sein muss, dass diese Prüfung auch zu einem späteren Zeitpunkt (z. B. durch juristische Instanzen) möglich ist.

- Die Unterschrift muss in einem direkten Zusammenhang zu dem unterschriebenen Dateiinhalt stehen, so dass sie gleichzeitig den entsprechenden Dateiinhalt authentifiziert und so jeder mögliche Empfänger (insbesondere juristische Instanzen auch noch zu einem späteren Zeitpunkt) anhand der Unterschrift auch den Dateiinhalt verifizieren kann (Prüfung der Dateiintegrität).

- Die Unterschriftslösung muss auf jeden beliebigen Kontext anwendbar sein.

- Das Unterschriftsverfahren muss unter Performancegesichtspunkten auch auf weniger leistungsfähigen PC mit vertretbarer Rechenzeitintensität einsetzbar sein.

- Der Verwaltungsaufwand für die notwendige Aufbewahrung der zur Unterschriften-erzeugung und insbesondere -prüfung benötigten Daten (Kennungen) muss möglichst gering sein (einfaches Key Management).

- Die konkrete technische Lösung muss für die gängigen Betriebssysteme, die beim Unterzeichner bzw. Empfänger zum Einsatz kommen können, kompatibel einsetzbar sein.

- The betriebssystemabhängigem Zeichen CR, LF und Ctrl-Z werden in die Hashwertbildung der A005/A006-EU nicht einbezogen (analog zur A004-EU).

Dieses Anforderungsprofil kann nur durch den Einsatz asymmetrischer kryptographischer Verfahren erfüllt werden.

Die Verwendung der Elektronischen Unterschrift wird für alle Datenübertragungen, die nicht der reinen Informationsbeschaffung dienen, dringend empfohlen, soweit in den besonderen Vereinbarungen für einzelne Verfahren nichts Abweichendes bestimmt ist.

Für jedes konkret zum Einsatz kommende Sicherheitsverfahren muss eine detaillierte Beschreibung der darin verwendeten mathematischen Verfahren sowie der verwendeten Datenstrukturen kostenlos offengelegt werden. Diese Beschreibung muss geeignet sein, ein funktional kompatibles Produkt durch beliebige Hersteller erstellen zu lassen. Außerdem muss ein positives Gutachten eines vom Kreditgewerbe bestimmten Gutachters zu dem jeweiligen Gesamtverfahren und speziell zu den darin zum Einsatz kommenden mathematischen Prozeduren vorgelegt werden.

14.1 Die Elektronische Unterschrift der Version A005/A006

Unter Beachtung der unter Kapitel 14 formulierten Anforderungen sind die nachfolgend beschriebenen Verfahren für die Elektronische Unterschrift ab dem 1. September 2009 bankseitig verpflichtend zu unterstützen.
Für die Signaturverfahren A005 und A006 wird ein Intervall von 1536 Bit (Minimum) und 4096 Bit (Maximum) für die Schlüssellänge festgelegt.

14.1.1 Vorbemerkung und Einleitung

Das Kapitel 14.1 beschreibt in den folgenden Unterkapiteln zwei neue Signaturverfahren beschrieben, die beide auf den Vorgaben aus [PKCS1] und dem Einsatz von SHA-256 als Hash-Algorithmus basieren. Die beiden Verfahren unterscheiden sich jedoch bezüglich des zum Einsatz kommenden Padding-Verfahrens aus [PKCS1].
Seit der Fertigstellung von [A005] wurde die Namensgebung für die beiden Signaturverfahren geändert. Wurden die beiden Signaturverfahren in [A005] noch als A005_V1.5 und A005_PSS bezeichnet, werden sie zukünftig als A005 und A006 bezeichnet. Die folgende Tabelle zeigt die Zuordnung zwischen neuen und alten Namen und zu den Signaturverfahren aus [PKCS1]:

Zukünftiger Name	Name in [A005]	[PKCS1]
A005	A005_V1.5	EMSA-PKCS1-v1_5 mit SHA-256
A006	A005_PSS (mit vorher erzeugtem SHA-256 Hash-Wert als Eingabewert)	EMSA-PSS mit SHA-256 (mit vorher erzeugtem SHA-256 Hash-Wert als Eingabewert)

Die folgende Beschreibung der beiden Signaturverfahren basiert auf den entsprechenden Abschnitten aus der Spezifikation von SECCOS 6 [SECCOS6]. Beide Signaturverfahren werden durch eine ZKA Signaturkarte unterstützt, die auf SECCOS 6 basiert und die zugehörige ZKA Signatur Anwendung [ZKASigAnw] (oder eine höhere Version) enthält.
Da bei kreditwirtschaftlichen Anwendungen zukünftig sowohl der AUT-Schlüssel als auch der DS-Schlüssel einer auf SECCOS basierenden ZKA-Signaturkarte mit Signatur Anwendung gemäß [ZKASigAnw] für das Erstellen einer Signatur genutzt werden sollen, muss die folgende Besonderheit berücksichtigt werden:

- Bei Nutzung des AUT-Schlüssels wird das Kommando INTERNAL AUTHENTICATE verwendet. Soll dieser Schlüssel mit dem PSS-Padding aus [PKCS1] verwendet werden, erzeugt die SECCOS-Chipkarte bei Aufruf des Kommandos INTERNAL AUTHENTICATE immer intern den benötigten Hash-Wert über die Eingabedaten. Da die aufrufende Anwendung aber vor Aufruf ebenfalls einen Hash-Wert über die

eigentliche Nachricht M erzeugt, kommt es hierdurch zu einer doppelten Berechnung des Hash-Wertes, d. h. es wird der Wert Hash(Hash(M)) berechnet.

- Aus diesem Grund wird A006 so definiert, dass in das eigentliche Signaturverfahren bereits ein vorher berechneter Hash-Wert über die Nachricht M als Eingabewert eingeht, und nicht die Nachricht M selber.

Die von der ZKA-Chipkarte bereitgestellten asymmetrischen kryptographischen Algorithmen basieren auf dem RSA-Algorithmus mit ungeraden öffentlichen Exponenten ([RSA]).
In Kapitel 14.1.2 des vorliegenden Dokuments werden das Konstruktionsprinzip und die **Schlüsselkomponenten** der öffentlichen und privaten RSA-Schlüssel gemäß Anhang F von [EMV CA] und [PKCS1] für ungerade öffentliche Exponenten erläutert.
Ein ***Signatur-Algorithmus*** besteht aus einem Algorithmus zur Signaturerzeugung und einem hierzu inversen Algorithmus zur Klartextrückgewinnung. Der als Standard von der ZKA-Chipkarte unterstützte Signatur-Algorithmus wird in Kapitel 14.1.3 beschrieben.
Der beschriebene Signatur-Algorithmus auf Basis des RSA-Algorithmus wird durch die ZKA-Chipkarte nur im Rahmen von Signaturverfahren eingesetzt. Ein ***Signaturverfahren*** legt fest, in welcher Weise eine Nachricht M zu einer Bytefolge aufzubereiten ist, die dann in die Signaturerzeugung eines Signatur-Algorithmus eingeht. Die durch ein Signaturverfahren erzeugte Bytefolge wird als ***Digital Signature Input*** (***DSI***) bezeichnet.
Die ZKA Signatur Anwendung [ZKASigAnw] unterstützt mehrere verschiedene Signaturverfahren. In Kapitel 14.1.4 des vorliegenden Dokuments werden die beiden neuen Signaturverfahren A005 und A006 beschrieben, die beide auf den Padding-Verfahren aus [PKCS1] und dem Hash-Algorithmus SHA-256 aufbauen.

14.1.2 RSA

Ein RSA-Schlüsselpaar besteht aus

- einem ***öffentlichen Schlüssel*** P_K und
- einem ***privaten Schlüssel*** S_K.

Öffentlicher und privater Schlüssel bestehen aus ***Schlüsselkomponenten***. RSA-Schlüssel werden auch als ***asymmetrische Schlüssel*** bezeichnet.
Zur Erzeugung eines RSA-Schlüsselpaares mit einem ungeraden ***öffentlichen Exponenten e*** werden zwei verschiedene ***Primzahlen p und q (Primfaktoren)*** verwendet, für die e teilerfremd zu (p-1) und (q-1) ist.
Der zugehörige ***private Exponent d*** ist dann bestimmt durch

$$e*d \equiv 1 \bmod kgV(p-1, q-1).$$

Die Primzahlen p und q sowie der private Exponent d müssen geheim gehalten werden.
Das Produkt der Primzahlen n = p*q wird als ***Modulus*** bezeichnet.
Der ***öffentliche Schlüssel*** P_K des RSA-Schlüsselpaares besteht aus den Komponenten

- ***Modulus n*** und
- ***öffentlicher Exponent e***.

Der ***private Schlüssel S_K*** des RSA-Schlüsselpaares kann auf zwei Arten durch Komponenten dargestellt werden (vgl. [PKCS1]):

1. Darstellung von S_K durch die Komponenten
   - ***Modulus n*** und
   - ***privater Exponent d***, oder
2. Darstellung von S_K durch die Komponenten
   - ***Primfaktor p,***
   - ***Primfaktor q,***
   - $d_p = d \bmod (p-1)$,
   - $d_q = d \bmod (q-1)$ ***und***
   - $qInv = q^{-1} \bmod p$.

Bei der ersten Darstellung muss nur die Komponente d geheim gehalten werden. Die Komponenten der 2. Darstellung werden als ***Chinese Remainder Theorem-Parameter*** (***CRT-Parameter***) bezeichnet. Die CRT-Parameter müssen sämtlich geheim gehalten werden.
Die ZKA-Chipkarte unterstützt den RSA-Algorithmus mit beliebigen ungeraden öffentlichen Exponenten. In der Regel wird aber einer der ungeraden öffentlichen Exponenten 3 oder $F_4 = 2^{16}+1$ verwendet.

In dem vorliegenden Dokument wird die folgende Notation verwendet:
k bezeichnet die Bit-Länge des Modulus n eines RSA-Schlüsselpaares.
k ist durch die Gleichung $2^{k-1} <= n < 2^k$ eindeutig definiert.
n lässt sich darstellen als Folge von Bits

$n = b_k\ b_{k-1} \ldots b_1$, wobei $b_k <> 0$ ist.

Die durch n repräsentierte ganze Zahl wird dadurch bestimmt, dass das erste, linke Bit b_k das höchstwertige Bit und das letzte, rechte Bit b_1 das niedrigstwertige Bit in der Binärdarstellung von n ist.
Zu k existieren eindeutige Zahlen N >= 1 und 8 >= r >=1 mit k = 8*(N-1) + r. Dann lässt sich n auch als Folge von Bits schreiben als

$n = b_r\ b_{r-1} \ldots b_1\ b_{8*(N-1)} \ldots b_{8*(N-2)+1} \ldots b_8 \ldots b_1$.

Wenn r = 8 ist, lässt sich n direkt als Folge von N Bytes schreiben:

$n = B_N\ B_{N-1} \ldots B_1$, wobei $B_N <>$ '00' ist.

Falls r < 8 ist, stellt man 8-r binäre 0 der Bitfolge $b_r\ b_{r-1} \ldots b_1\ b_{8*(N-1)} \ldots b_{8*(N-2)+1} \ldots b_8 \ldots b_1$ voran:

$n = 0 \ldots 0\ b_r\ b_{r-1} \ldots b_1\ b_{8*(N-1)} \ldots b_{8*(N-2)+1} \ldots b_8 \ldots b_1$.

und erhält wiederum eine Bytefolge

$n = B_N\ B_{N-1} \ldots B_1$, wobei B_N <> '00' ist.

Der ganzzahlige Wert von n wird durch führende 0 in der Binärdarstellung nicht geändert, so dass die Darstellung von n als Folge von N Bytes den durch eine Folge von k Bits dargestellten Zahlwert unverändert lässt.
N ist die Byte-Länge von n.
N ist durch die Gleichung $2^{8*(N-1)} <= n < 2^{8*N}$ eindeutig definiert.

14.1.3 Signaturalgorithmus

14.1.3.1 Standardfunktion für die Signaturerzeugung

Sei S_K ein privater RSA-Schlüssel bestehend aus dem Modulus n und dem privaten Exponenten d oder bestehend aus den CRT-Parametern. Der zugehörige öffentliche RSA-Schlüssel P_K bestehe aus dem Modulus n und dem öffentlichen Exponenten e.
Dann können mit S_K binär kodierte Bytefolgen x signiert werden, deren sich aus der Binärdarstellung von x ergebender ganzzahliger Wert zwischen 0 und n-1 liegt. x lässt sich somit darstellen als Bytefolge mit einer Länge von N Bytes und als Bitfolge mit einer Länge von k Bits. Das k-te Bit in der darstellenden Byte- oder Bitfolge kann, muss aber nicht, den Wert 1 haben. Sofern vorhanden haben die Bits $b_{8*N} \ldots b_{k+1}$ in der darstellenden Bytefolge den Wert 0.
Für die Signaturerzeugung mit dem aus n und d bestehenden privaten Schlüssel wird die folgende Notation verwendet:

$sign(S_K)[x] = x^d \bmod n$

Falls der private Schlüssel S_K aus den CRT-Parametern besteht, berechnet sich $sign(S_K)[x] = x^d \bmod n$ wie folgt:

$sign(S_K)[x] = s_2 + h*q$

wobei s_2 und h wie folgt berechnet werden:

$s_1 = x^{dp} \bmod p$,

$s_2 = x^{dq} \bmod q$,

$h = qInv * (s_1 - s_2) \bmod p$.

Hierbei werden die Potenzierungen $x^d \bmod n$, $x^{dp} \bmod p$, $x^{dq} \bmod q$ mit der ganzen Zahl ausgeführt, deren Wert sich aus der Binärdarstellung von x ergibt.
Das Ergebnis der Signaturerzeugung ist wiederum eine Bytefolge s, die sich als Binärdarstellung des ganzzahligen Wertes der Potenz $x^d \bmod n$ bzw. von $s_2 + h*q$ ergibt. Dieser ganzzahlige Wert liegt wieder zwischen 0 und n-1. s lässt sich somit darstellen als Bytefolge mit einer Länge von N Bytes und als Bitfolge mit einer Länge von k Bits. Das k-te

Bit in der darstellenden Byte- oder Bitfolge kann, muss aber nicht, den Wert 1 haben. Sofern vorhanden haben die Bits b_{8*N} ... b_{k+1} in der darstellenden Bytefolge den Wert 0.

14.1.3.2 Standardfunktion für die Klartextrückgewinnung

Sei P_K ein öffentlicher RSA-Schlüssel bestehend aus dem Modulus n und dem öffentlichen Exponenten e.
Dann kann mit P_K aus einer binär kodierten Bytefolge s Klartext zurück gewonnen werden, wenn der sich aus der Binärdarstellung von s ergebende ganzzahlige Wert zwischen 0 und n-1 liegt. s lässt sich somit darstellen als Bytefolge mit einer Länge von N Bytes und als Bitfolge mit einer Länge von k Bits. Das k-te Bit in der darstellenden Byte- oder Bitfolge kann, muss aber nicht, den Wert 1 haben. Sofern vorhanden haben die Bits b_{8*N} ... b_{k+1} in der darstellenden Bytefolge den Wert 0.
Für die Klartextrückgewinnung wird die folgende Notation verwendet:

$$recover(P_K)[s] = s^e \bmod n$$

Hierbei wird die Potenzierung s^e mod n mit der ganzen Zahl ausgeführt, deren Wert sich aus der Binärdarstellung von s ergibt.
Das Ergebnis der Klartextrückgewinnung ist eine ganze Zahl, deren Wert zwischen 0 und n-1 liegt. Es lässt sich somit darstellen als Bytefolge mit einer Länge von N Bytes und als Bitfolge mit einer Länge von k Bits. Das k-te Bit in der darstellenden Byte- oder Bitfolge kann, muss aber nicht, den Wert 1 haben. Sofern vorhanden haben die Bits b_{8*N} ... b_{k+1} in der darstellenden Bytefolge den Wert 0.
Für ein RSA-Schlüsselpaar P_K und S_K gilt

$$recover(P_K)[sign(S_K)[x]] = x$$

14.1.4 ZKA Signaturverfahren A005 und A006

Die Signaturverfahren A005 und A006 basieren beide auf dem Industriestandard PKCS#1 [PKCS1] und dem Einsatz von SHA-256 als Hash-Algorithmus. Beides sind Signaturverfahren ohne Nachrichtenrückgewinnung.
Ein Hash-Algorithmus bildet Bitfolgen beliebiger Länge (***Eingabe-Bitfolgen***) auf Bytefolgen einer festen, durch den Hash-Algorithmus festgelegten Länge ab. Das Ergebnis der Anwendung eines Hash-Algorithmus auf eine Bitfolge wird als ***Hash-Wert*** bezeichnet.
Der Hash-Algorithmus SHA-256 wird in [FIPS H2] spezifiziert. SHA-256 bildet Eingabe-Bitfolgen beliebiger Länge auf eine Bytefolge von 32 Bytes Länge ab. Das Padding der Eingabe-Bitfolge auf ein Vielfaches von 64 Bytes ist Bestandteil des Hash-Algorithmus. Das Padding erfolgt auch dann, falls die Länge der Eingabe-Bitfolge bereits ein Vielfaches von 64 Bytes ist.
SHA-256 verarbeitet die Eingabe-Bitfolge in Blöcken von 64 Bytes Länge.
Der Hash-Wert einer Eingabe-Bitfolge x unter dem Hash-Algorithmus SHA-256 wird wie folgt bezeichnet:

SHA-256(x)

Für das Generieren des Eingabewertes für die eigentliche Signaturerzeugung **(Digital Signature Input DSI)** unterscheidet PKCS#1 [PKCS1] die beiden Methoden EMSA-PKCS1-v1_5 und EMSA-PSS. Entsprechend werden die zwei Signaturverfahren des ZKA unterschieden, genannt A005 und A006.

14.1.4.1 Signaturverfahren A005

Für das Berechnen und das Verifizieren einer digitalen Signatur mit dem in [PKCS1] beschriebenen Signaturverfahren mit Nutzung der Methode EMSA-PKCS1-v1_5 müssen die folgenden Punkte angegeben werden:

- der zu verwendende Hash-Algorithmus HASH,
- die Länge H des generierten Hash-Wertes (in Bytes),
- der zum Einsatz kommende Signatur-Algorithmus und
- die maximale Länge N des erzeugten DSI als Eingabe für den Signatur-Algorithmus.

Das Signaturverfahren A005 ist identisch mit EMSA-PKCS1-v1_5, wobei der Hash-Algorithmus SHA-256 verwendet wird. Die Länge H des Hash-Wertes beträgt 32 Bytes.
Als Signatur-Algorithmus wird bei ZKA Chipkarten RSA verwendet. Der Wert N entspricht daher der Bytelänge des Modulus n des verwendeten RSA-Schlüsselpaares.
In den folgenden Abschnitten wird das Berechnen und das Verifizieren einer digitalen Signatur mit dem Signaturverfahren A005 beschrieben. Dabei werden die in Kapitel 14.1.2 definierten Abkürzungen verwendet.

14.1.4.1.1 Berechnen der digitalen Signatur

Gemäß [PKCS1] müssen (bei Nutzung der Methode EMSA-PKCS1-v1_5) die folgenden Schritte für das Berechnen einer digitalen Signatur über eine Nachricht M mit der Bitlänge m ausgeführt werden.

1. Der Hash-Wert HASH(M) mit einer Länge von H Bytes muss berechnet werden. Für das Signaturverfahren A005 SHA-256(M) mit einer Länge von 32 Bytes.

2. Der Eingabewert DSI für den Signatur-Algorithmus muss erzeugt werden.

   Der DSI ist eine Bytefolge von N-1 Bytes, die wie folgt aufgebaut wird:

Bezeichnung	**Länge (Bytes)**	**Wert**
Block-Typ	1	'01'
Padding-Feld	N-3-D	'FF..FF'
Separator	1	'00'
Digest-Info	D	BER-TLV kodiertes Datenobjekt mit der OID und den Parametern des Hash-Algorithmus HASH und dem Hash-Wert HASH(M)

   Bei einer Nutzung von SHA-256 ist das Digest-Info wie folgt aufgebaut:

Tag	**Length** (in byte)	**Value**	**Description**
'30'	'31'		Tag und Länge SEQUENCE
'30'	'0D'		Tag und Länge SEQUENCE
'06'	'09'	'60 86 48 01 65 03 04 02 01'	OID von SHA-256 (2 16 840 1 101 3 4 2 1)
'05'	'00'	-	TLV Kodierung von ZERO (keine Parameter)
'04'	'20'	'XX..XX'	Hash-Wert

   Das Digest-Info hat bei Nutzung von SHA-256 eine Länge D von 51 Bytes. Das Padding-Feld hat eine Länge von N-54 Bytes. Da N mindestens den Wert 128 hat (entspricht einer minimalen Schlüssellänge von 1024 Bit), müssen mindestens 74 Bytes 'FF' als Padding eingefügt werden.

3. Die digitale Signatur wird mit der Standardfunktion für die Signaturerzeugung über die Eingabebytefolge DSI wie in Abschnitt 14.1.3.1 beschrieben berechnet.

   Da der DSI eine Bytefolge von N-1 Bytes ist, ist die durch den DSI (betrachtet als Binärdarstellung) gegebene ganze Zahl immer kleiner als der Wert n des Modulus.

Die berechnete digitale Signatur kann als Bytefolge der Länge N dargestellt werden. Bei der Darstellung des Modulus n als Bytefolge hat das Bit b_k den Wert 1 und die Bits b_{8*N} b_{8*N-1} ... b_{k+1} haben, falls vorhanden, den Wert 0. In der Darstellung der digitalen Signatur als Bytefolge müssen daher die Bits b_{8*N} b_{8*N-1} ... b_{k+1} ebenfalls den Wert 0 haben.

14.1.4.1.2 Verifizieren der digitalen Signatur

Gemäß [PKCS1] müssen (bei Nutzung der Methode EMSA-PKCS1-v1_5) die folgenden Schritte für das Verifizieren einer digitalen Signatur ausgeführt werden. Die zu verifizierende Signatur und die Nachricht M' müssen dabei als Bytefolgen vorliegen.

1. Die digitale Signatur muss eine Bytefolge der Länge N sein. Dabei müssen die Bits b_{8*N} b_{8*N-1} ... b_{k+1}, falls vorhanden, den Wert 0 haben. Ist dies nicht der Fall, muss die Signatur abgelehnt werden.

   Die durch die Signatur (betrachtet als Binärdarstellung) gegebene ganze Zahl muss kleiner sein als der Modulus n. Ist dies nicht der Fall, wird die digitale Signatur abgelehnt.

2. Die Standardfunktion für die Klartextrückgewinnung muss wie in Abschnitt 14.1.3.2 beschrieben auf die Signatur angewendet werden. Das Ergebnis muss eine Bytefolge der Länge N-1 sein. Ist dies nicht der Fall, muss die Signatur abgelehnt werden.

3. Der Vergleichswert DSI' mit einer Länge von N-1 Bytes wird aus der Nachricht M' wie in den Schritten 1 und 2 von Abschnitt 14.1.4.1.1 beschrieben erzeugt.

4. DSI' muss mit dem in Schritt 2 aus der Signatur zurück gewonnenen Klartext verglichen werden. Stimmen beide Werte überein, wurde die Signatur erfolgreich verifiziert. Andernfalls muss die Signatur abgelehnt werden.

14.1.4.1.3 Notation

Für die Berechnung einer Signatur zu einer Nachricht M mit dem Signaturverfahren A005 und dem privaten RSA-Schlüssel S_K wird die folgende Notation verwendet:

$s = sign_{A005}(S_K)[M]$.

Für die Prüfung einer Signatur s zu einer Nachricht M mit dem Signaturverfahren A005 und dem öffentlichen RSA-Schlüssel P_K wird die folgende Notation verwendet:

$verify_{A005}(P_K)[s,M]$.

14.1.4.2 Signaturverfahren A006

Für das Berechnen und das Verifizieren einer digitalen Signatur mit dem in [PKCS1] beschriebenen Signaturverfahren mit Nutzung der Methode EMSA-PSS müssen die folgenden Punkte angegeben werden:

- der zu verwendende Hash-Algorithmus HASH,
- die Länge H des generierten Hash-Wertes (in Bytes),
- die Länge S des zu nutzenden Zufallswerts salt (in Bytes),
- die zum Einsatz kommenden Maskierfunktion (mask generation function),
- der zum Einsatz kommende Signatur-Algortihmus,
- die maximale Länge k (in Bits) des erzeugten DSI als Eingabe für den Signatur-Algorithmus sowie
- die maximale Länge N (in Bytes) des erzeugten DSI als Eingabe für den Signatur-Algorithmus.

Das Signaturverfahren A006 basiert auf EMSA-PSS, wobei der Hash-Algorithmus SHA-256 verwendet wird. Die Länge H des Hash-Wertes beträgt 32 Byte.
Die Länge S des Zufallswerts salt wird durch den zum Einsatz kommenden Hash-Algorithmus definiert, d.h. die Länge S des Zufallswerts salt ist identisch mit der Länge H des Hash-Wertes.
Im Rahmen des Signaturverfahrens A006 wird nur die in [PKCS1] definierte Maskierfunktion MGF1 verwendet.
Notation: k ist die Länge des Modulus n (in Bits) des verwendeten RSA-Schlüssels. Die Länge des erzeugten DSI (in Bits) ist k-1 und wird mit emBits bezeichnet. Die Länge des Modulus n (in Bytes) wird mit N bezeichnet. Die Länge des DSI (in Bytes) wird mit emLen bezeichnet.

14.1.4.2.1 Maskierfunktion MGF1

Das in den Abschnitten 8.1 und 9.1 von [PKCS1] beschriebene Signaturverfahren nutzt die Maskierfunktion MGF1, die in Abschnitt B.2 von [PKCS1] beschrieben wird.
Die Maskierfunktion MGF1 basiert auf einem Hash-Algorithmus HASH, der Hash-Werte der Länge H (in Bytes) erzeugt. MGF1 erzeugt dann wie folgt aus einem Startwert **mgfSeed** eine Bytefolge der gegebenen Länge **maskLen**:

1. Sei T eine leere Bytefolge.
2. Für einen Zähler von 0 bis $\lceil maskLen / H \rceil - 1$ führe das folgende durch:
   a. Konvertiere den Zähler in eine Bytefolge C der Länge 4 Bytes.
   b. Berechne den Hashwert HASH(mgfSeed | C) und konkateniere diesen mit der aktuellen Bytefolge T, d.h.:

T = T | HASH(mgfSeed | C)

3. Das Ergebnis MGF1(mgfSeed, maskLen) sind die höchstwertigen ("leftmost") maskLen Bytes der Bytefolge T.

Hinweis: ⌈maskLen / H⌉ ist definiert als die kleinste ganze Zahl, die größer als oder gleich (masklen / H) ist.

14.1.4.2.2 Berechnen der digitalen Signatur mit EMSA-PSS

Gemäß den Abschnitten 8.1.1 und 9.1.1 von [PKCS1] müssen (bei Nutzung der Methode EMSA-PSS) die folgenden Schritte für das Berechnen einer digitalen Signatur über eine Nachricht M mit Bitlänge m ausgeführt werden:

1. Der Hash-Wert HASH(M) mit der Länge H (in Bytes) muss berechnet werden. Wird EMSA-PSS als Basis für das Signaturverfahren A006 genutzt, wird SHA-256(M) mit einer Länge von 32 Bytes berechnet.

2. Der Eingabewert DSI für den Signatur-Algorithmus muss wie folgt erzeugt werden:

   Erzeuge eine Zufallszahl der Länge S Bytes, die als Zufallswert salt genutzt wird.

   Erzeuge die Nachricht M' wie folgt:

   M' = '00 00 00 00 00 00 00 00' | HASH(M) | salt

   Berechne über M' den Hash-Wert HASH(M') der Länge H.

   Bilde eine Padding-Bytefolge PS der Länge emLen – H – S – 2 Bytes bestehend aus '00' Bytes.

   Setze DB = PS | '01' | salt; DB ist eine Bytefolge der Länge emLen – H – 1.

   Sei dbMask = MGF(HASH(M'), emLen – H – 1) das Ergebnis der Maskierfunktion. Hinweis: Wird EMSA-PSS als Basis für A006 genutzt, wird an dieser Stelle MGF1 aus Abschnitt 14.1.4.2.1 verwendet.

   Setze maskedDB = DB ⊕ dbMask.

   Setze die 8*emLen – emBits höchstwertigen ("leftmost") Bits des höchstwertigen ("leftmost") Byte von maskedDB auf 0.

   Setze DSI = maskedDB | HASH(M') | 'BC'.

3. Die digitale Signatur wird mit der Standardfunktion für die Signaturerzeugung über die Eingabebytefolge DSI wie in Abschnitt 14.1.3.1 beschrieben berechnet.

Die Bytefolge DSI hat eine Länge von emLen Bytes. Die durch den DSI (betrachtet als Binärdarstellung) gegebene ganze Zahl ist immer kleiner als der Wert n des Modulus, da die Bitlänge emBits des DSI kleiner ist als die Bitlänge des Modulus.

Die berechnete digitale Signatur kann als Bytefolge der Länge N dargestellt werden. Bei der Darstellung des Modulus n als Bytefolge hat das Bit b_k den Wert 1 und die Bits $b_{8*N}\, b_{8*N-1}\, ...\, b_{k+1}$ haben, falls vorhanden, den Wert 0. In der Darstellung der digitalen Signatur als Bytefolge müssen daher die Bits $b_{8*N}\, b_{8*N-1}\, ...\, b_{k+1}$ ebenfalls den Wert 0 haben.

14.1.4.2.3 Verifizieren der digitalen Signatur

Gemäß den Abschnitten 8.1.2 und 9.1.2 von [PKCS1] müssen (bei Nutzung der Methode EMSA-PSS) die folgenden Schritte für das Verifizieren einer digitalen Signatur ausgeführt werden. Die zu verifizierende Signatur und die Nachricht M müssen dabei als Bytefolgen vorliegen.

1. Die digitale Signatur muss eine Bytefolge der Länge N sein. Dabei müssen die Bits $b_{8*N}\, b_{8*N-1}\, ...\, b_{k+1}$, falls vorhanden, den Wert 0 haben. Ist dies nicht der Fall, muss die Signatur abgelehnt werden.

   Die durch die Signatur (betrachtet als Binärdarstellung) gegebene ganze Zahl muss kleiner sein als der Modulus n. Ist dies nicht der Fall, wird die digitale Signatur abgelehnt.

2. Die Standardfunktion für die Klartextrückgewinnung muss wie in Abschnitt 14.1.3.2 beschrieben auf die Signatur angewendet werden. Das Ergebnis muss eine Bytefolge der Länge emLen sein. Ist dies nicht der Fall, muss die Signatur abgelehnt werden.

3. Der zurück gewonnene Klartext muss wie folgt überprüft werden:

   Berechne den Hash-Wert HASH(M) mit der Länge H Bytes.

   Das niedrigstwertige Byte des Klartexts muss den Wert 'BC' haben. Ist dies nicht der Fall, muss die Signatur abgelehnt werden.

   Seien maskedDB die emLen – H – 1 höchstwertigen ("leftmost") Bytes des zurück gewonnenen Klartexts und HM' die nächsten H Bytes des Klartexts.

   Falls die 8*emLen – emBits höchstwertigen ("leftmost") Bits des höchstwertigen ("leftmost") Byte von maskedDB nicht den Wert 0 haben, muss die Signatur abgelehnt werden.

   Setze dbMask = MGF (HM', emLen – H – 1) mit der Maskierfunktion MGF1.

   Sei DB = maskedDB $\oplus$ dbMask.

Setze die 8*emLen – emBits höchstwertigen ("leftmost ") Bits des höchstwertigen Byte von DB auf 0.

Falls die emLen – H – S – 2 höchstwertigen ("leftmost") Bytes von DB nicht alle den Wert '00' haben oder das Byte an der Position emLen – H – S – 1 nicht den Wert '01' hat, muss die Signatur abgelehnt werden.

Sei salt die S niedrigstwertigen ("rightmost") Bytes von DB.

Setze M' = '00 00 00 00 00 00 00 00' | HASH(M) | salt

und berechne den Hash-Wert HASH(M') der Länge H.

Falls HM' = HASH(M') gilt, wurde die Signatur erfolgreich verifiziert. Andernfalls muss die Signatur abgelehnt werden.

14.1.4.2.4 Notation für EMSA-PSS

Für die Berechnung einer Signatur zu einer Nachricht M mit der Methode EMSA-PSS gemäß [PKCS1] und dem privaten RSA-Schlüssel S_K wird die folgende Notation verwendet:

$s = sign_{EMSA\text{-}PSS}(S_K)[M]$.

Für die Prüfung einer Signatur s zu einer Nachricht M mit der Methode EMSA-PSS gemäß [PKCS1] und dem öffentlichen RSA-Schlüssel P_K wird die folgende Notation verwendet:

$verify_{EMSA\text{-}PSS}(P_K)[s,M]$.

14.1.4.2.5 Berechnen der digitalen Signatur mit A006

Wie bereits in den Vorbemerkungen erwähnt, soll bei kreditwirtschaftlichen Anwendungen auch der (eigentlich für Authentikationszwecke konzipierte) AUT-Schlüssel für die Berechnung von Signaturen eingesetzt werden. Dabei berechnet eine SECCOS-Chipkarte bei Nutzung des Kommandos INTERNAL AUTHENTICATE (AUT-Schlüssel) im Rahmen des Signaturverfahrens EMSA-PSS immer intern den Hash-Wert über den Eingabewert. Bei diesen Anwendungen sind die Nachrichten, über die die Signaturen berechnet werden, im allgemein sehr lang. Sie können daher nicht direkt als Eingabewert mit dem Kommando INTERNAL AUTHENTICATE an die SECCOS-Chipkarte übergeben werden. Unter anderem aus diesem Grunde berechnet die Anwendung vor Aufruf des SECCOS-Kommandos ebenfalls einen Hash-Wert über die Nachricht und übergibt diesen dann als Eingabewert mit dem Kommando. Folglich kommt es bei Nutzung des AUT-Schlüssels immer zu einer doppelten Berechnung des Hash-Werts. Diesen Umstand trägt die Definition des Signaturverfahrens A006 Rechnung.
Um über eine Nachricht M eine Signatur s mit dem privaten Schlüssel S_K und dem Signaturverfahren A006 zu berechnen, führe die beiden folgenden Schritte durch:

- Berechne den Hashwert HM = SHA-256(M).
- Berechne dann die Signatur $s = sign_{EMSA\text{-}PSS}(S_K)[HM]$.

14.1.4.2.6 Verifizieren der digitalen Signatur mit A006

Um eine Signatur s zu einer Nachricht M mit dem öffentlichen Schlüssel P_K und dem Signaturverfahren A006 zu überprüfen, führe die beiden folgenden Schritte durch:

- Berechne den Hashwert HM = SHA-256(M).
- Überprüfe dann s mittels $verify_{EMSA\text{-}PSS}(P_K)[s,HM]$.

14.1.4.2.7 Notation für A006

Für die Berechnung einer Signatur zu einer Nachricht M mit dem Signaturverfahren A006 und dem privaten RSA-Schlüssel S_K wird die folgende Notation verwendet:

$s = sign_{A006}(S_K)[M]$.

Für die Prüfung einer Signatur s zu einer Nachricht M mit dem Signaturverfahren A006 und dem öffentlichen RSA-Schlüssel P_K wird die folgende Notation verwendet:

$verify_{A006}(P_K)[s,M]$.

14.1.5 Referenzen

[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-Struktur der Unterschriftsversionen A005/A006

Die bankfachliche Elektronische Unterschrift (EU) in strukturierter Form hat folgenden Aufbau:

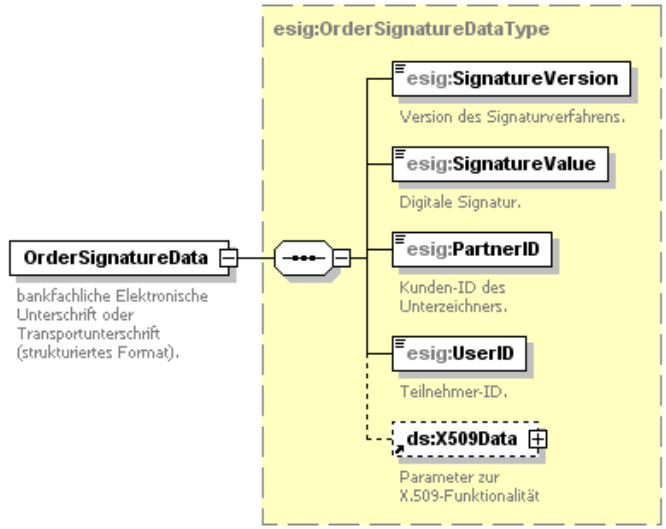


Abbildung 105: OrderSignatureData – strukturierte Elektronische Unterschrift

`OderSignatureData` darf immer nur als Bestandteil eines XML Dokuments mit dem Rootelement `UserSignatureData` übertragen werden. Weitere Erläuterungen und Abbildung siehe Kapitel 3.5.3.

Mit der Intention, die strukturierte Form der EU auch außerhalb von EBICS nutzen zu können, wurden alle notwendigen Datenstrukturen in einer eigenständigen XSD-Datei (ebics_signature.xsd) definiert. Sie ist unter http://www.ebics.de (siehe „Spezifikation“) abrufbar.
Für den Transport der öffentlichen Signaturschlüssel wird das Format `SignaturePubKeyInfoType` verwendet (siehe Kapitel 4.2).

14.2 Die Elektronische Unterschrift der Version A004

Das nachfolgend beschriebene Verfahren für die Elektronische Unterschrift ist seit 1. September 2009 nicht mehr bankseitig verpflichtend zu unterstützen.

14.2.1 Einleitung

Die für die Version A004 der Elektronischen Unterschrift genutzten asymmetrischen kryptographischen Algorithmen basieren auf dem RSA-Algorithmus mit ungeraden öffentlichen Exponenten ([RSA], siehe Kapitel 14.2.5.5).

Im Kapitel 14.2.2 werden das Konstruktionsprinzip und die **Schlüsselkomponenten** der öffentlichen und privaten RSA-Schlüssel gemäß Anhang B von [EMV B2], Anhang A von [ISO DS2] und [PKCS1] für ungerade öffentliche Exponenten erläutert.

Ein **Signatur-Algorithmus** besteht aus einem Algorithmus zur Signaturerzeugung und einem hierzu inversen Algorithmus zur Klartextrückgewinnung. Der als Standard unterstützte Signatur-Algorithmus wird in Kapitel 14.2.3 unter „Signaturerzeugung“ beschrieben.

Der beschriebene Signatur-Algorithmus auf Basis des RSA-Algorithmus wird nur im Rahmen von Signaturverfahren eingesetzt. Ein **Signaturverfahren** legt fest, in welcher Weise eine Nachricht M zu einer Bytefolge aufzubereiten ist, die dann in die Signaturerzeugung eines Signatur-Algorithmus eingeht. Die durch ein Signaturverfahren erzeugte Bytefolge wird als **Digital Signature Input (DSI)** bezeichnet.

In Kapitel 14.2.5 wird das Signaturverfahren beschrieben, das zur Erzeugung digitaler Signaturen unterstützt wird. Dieses Signaturverfahren entspricht der DIN-Spezifikation [DINSIG] (siehe Kapitel 14.2.5.5) einer Signatur-Anwendung/Funktion nach SigG und SigV.

14.2.2 RSA-Schlüsselkomponenten

Ein RSA-Schlüsselpaar besteht aus

- einem öffentlichen Schlüssel P_K und
- einem privaten Schlüssel S_K

Öffentlicher und privater Schlüssel bestehen aus ***Schlüsselkomponenten***. RSA-Schlüssel werden auch als ***asymmetrische Schlüssel*** bezeichnet.

Zur Erzeugung eines RSA-Schlüsselpaares mit einem ungeraden ***öffentlichen Exponenten e*** werden zwei verschiedene ***Primzahlen p und q (Primfaktoren)*** verwendet, für die e teilerfremd zu (p-1) und (q-1) ist.

Der zugehörige ***private Exponent d*** ist dann bestimmt durch

$e*d \equiv 1 \bmod kgV(p-1, q-1)$.

Die Primzahlen p und q sowie der private Exponent d müssen geheimgehalten werden.

Das Produkt der Primzahlen n = p*q wird als ***Modulus*** bezeichnet.

Der ***öffentliche Schlüssel*** P_K des RSA-Schlüsselpaares besteht aus den Komponenten

- Modulus n und
- öffentlicher Exponent e.

Der ***private Schlüssel*** S_K des RSA-Schlüsselpaares kann auf zwei Arten durch Komponenten dargestellt werden (vgl. [PKCS1], siehe Kapitel 14.2.5.5):

1. Darstellung von S_K durch die Komponenten:
   - Modulus n und
   - privater Exponent d,
2. Darstellung von S_K durch die Komponenten:
   - Primfaktor p,
   - Primfaktor q,
   - $d_p = d \bmod (p-1)$,
   - $d_q = d \bmod (q-1)$ und
   - qInv = $q^{-1} \bmod p$.

Nur die Komponente d der ersten Darstellung muss geheimgehalten werden. Die Komponenten der 2. Darstellung werden als ***Chinese Remainder Theorem-Parameter*** (***CRT-Parameter***) bezeichnet. Die CRT-Parameter müssen sämtlich geheimgehalten werden.

Es wird der RSA-Algorithmus mit beliebigen ungeraden öffentlichen Exponenten unterstützt, die die Länge des Modulus nicht überschreiten. In der Regel wird der ungerade öffentliche Exponent$F_4 = 2^{16}+1$ verwendet.In dem vorliegenden Dokument wird die folgende Notation verwendet:

k bezeichnet die Bit-Länge des Modulus n eines RSA-Schlüsselpaares.

k ist durch die Gleichung $2^{k-1} <= n < 2^k$ eindeutig definiert.

n lässt sich darstellen als Folge von Bit

$n = b_k\ b_{k-1} ... b_1$, wobei $b_k <> 0$ ist.

Der Wert von n als ganzer Zahl wird dadurch bestimmt, dass das erste, linke Bit b_k das höchstwertige Bit und das letzte, rechte Bit b_1 das niedrigstwertige Bit in der Binärdarstellung von n ist.

Zu k existieren eindeutige Zahlen N >= 1 und 8 >= r >=1 mit k = 8*(N-1) + r. Dann lässt sich n auch als Folge von Bit schreiben als

$n = b_r\ b_{r-1} ... b_1\ b_{8*(N-1)} ... b_{8*(N-2)+1} ... b_8 ... b_1$.

Wenn r = 8 ist, lässt sich n direkt als Folge von N Bytes schreiben:

$n = B_N\ B_{N-1} ... B_1$, wobei $B_N <>$ '00' und $B_N >=$ '80' ist.

Falls r < 8 ist, stellt man 8-r binäre 0 der Bitfolge $b_r\ b_{r-1} ... b_1\ b_{8*(N-1)} ... b_{8*(N-2)+1} ... b_8 ... b_1$ voran:

$n = 0 ... 0\ b_r\ b_{r-1} ... b_1\ b_{8*(N-1)} ... b_{8*(N-2)+1} ... b_8 ... b_1$.

und erhält wiederum eine Bytefolge

$n = B_N\ B_{N-1} ... B_1$, wobei $B_N <>$ '00' und $B_N <$ '80' ist.

Der ganzzahlige Wert von n wird durch führende 0 in der Binärdarstellung nicht geändert, so dass die Darstellung von n als Folge von N Bytes den durch eine Folge von k Bit dargestellten Zahlwert unverändert lässt.

N ist die Byte-Länge von n.

Es werden nur Moduli verwendet, die mindestens 128 Bytes lang sind. Aus technischen Gründen können

- für die Berechnung einer Signatur nur Moduli mit einer maximalen Länge von 256 Bytes und
- für die Prüfung einer Signatur nur Moduli mit einer maximalen Länge von 252 Bytes

verarbeitet werden.

14.2.3 Signatur-Algorithmus

Signaturerzeugung

Es sei S_K ein privater RSA-Schlüssel bestehend aus dem Modulus n und dem privaten Exponenten d oder bestehend aus den CRT-Parametern. Der zugehörige öffentliche RSA-Schlüssel P_K bestehe aus dem Modulus n und dem öffentlichen Exponenten e.

Dann können mit S_K binär kodierte Bytefolgen x signiert werden, deren sich aus der Binärdarstellung von x ergebender ganzzahliger Wert zwischen 0 und n-1 liegt. x lässt sich somit darstellen als Bytefolge mit einer Länge von N Bytes und als Bitfolge mit einer Länge von k Bit. Das k-te Bit in der darstellenden Byte- oder Bitfolge kann, muss aber nicht, den Wert 1 haben. Sofern vorhanden haben die Bit b_{8*N} ... b_{k+1} in der darstellenden Bytefolge den Wert 0.

Für die Signaturerzeugung mit dem aus n und d bestehenden privaten Schlüssel wird die folgende Notation verwendet:

$$sign(S_K)[x] = x^d \bmod n$$

Falls der private Schlüssel S_K aus den CRT-Parametern besteht, berechnet sich $sign(S_K)[x] = x^d \bmod n$ wie folgt:

$$sign(S_K)[x] = s_2 + h*q$$

wobei S_2 und h wie folgt berechnet werden:

$s_1 = x^{dp} \bmod p$
$s_2 = x^{dq} \bmod q$
$h = qInv*(s_1 - s_2) \bmod p.$

Hierbei werden die Potenzierungen $x^d \bmod n$, $x^{dp} \bmod p$, $x^{dq} \bmod q$ mit der ganzen Zahl ausgeführt, deren Wert sich aus der Binärdarstellung von x ergibt.

Das Ergebnis der Signaturerzeugung ist wiederum eine Bytefolge s, die sich als Binärdarstellung des ganzzahligen Wertes der Potenz $x^d \bmod n$ bzw. von $s_2 + h*q$ ergibt. Dieser ganzzahlige Wert liegt wieder zwischen 0 und n-1. s lässt sich somit darstellen als Bytefolge mit einer Länge von N Bytes und als Bitfolge mit einer Länge von k Bit. Das k-te Bit in der darstellenden Byte- oder Bitfolge kann, muss aber nicht, den Wert 1 haben. Sofern vorhanden haben die Bit b_{8*N} ... b_{k+1} in der darstellenden Bytefolge den Wert 0.

Klartextrückgewinnung

Es sei P_K ein öffentlicher RSA-Schlüssel bestehend aus dem Modulus n und dem öffentlichen Exponenten e.

Dann kann mit P_K aus einer binär kodierten Bytefolge s Klartext zurückgewonnen werden, wenn sich der sich aus der Binärdarstellung von s ergebende ganzzahlige Wert zwischen 0 und n-1 liegt. s lässt sich somit darstellen als Bytefolge mit einer Länge von N Bytes und als Bitfolge mit einer Länge von k Bit. Das k-te Bit in der darstellenden Byte- oder Bitfolge kann, muss aber nicht, den Wert 1 haben. Sofern vorhanden haben die Bit b_{8*N} ... b_{k+1} in der darstellenden Bytefolge den Wert 0.

Für die Klartextrückgewinnung wird die folgende Notation verwendet:

$$recover(P_K)[s] = s^e \bmod n$$

Hierbei wird die Potenzierung s^e mod n mit der ganzen Zahl ausgeführt, deren Wert sich aus der Binärdarstellung von s ergibt.

Das Ergebnis der Klartextrückgewinnung ist eine ganze Zahl, deren Wert zwischen 0 und n-1 liegt. Es lässt sich somit darstellen als Bytefolge mit einer Länge von N Bytes und als Bitfolge mit einer Länge von k Bit. Das k-te Bit in der darstellenden Byte- oder Bitfolge kann, muss aber nicht, den Wert 1 haben. Sofern vorhanden haben die Bit b_{8*N} ... b_{k+1} in der darstellenden Bytefolge den Wert 0.

Für ein RSA-Schlüsselpaar P_K und S_K gilt:

$$recover(P_K)[sign(S_K)[x]] = x$$

14.2.4 Signaturverfahren gemäß DIN-Spezifikation

Im Folgenden werden die in ein Signaturverfahren eingehenden Nachrichten M als Bitfolge der Bit-Länge m aufgefasst. Eine Nachricht lässt sich demnach schreiben als Folge von Bit b_i

$$M = b_m\ b_{m-1} \ldots b_1$$

Wenn M als Binärzahl aufgefasst wird, ist das erste, linke Bit b_m das höchstwertige Bit und das letzte, rechte Bit b_1 das niedrigstwertige Bit. Das oder die höchstwertigen Bit einer Nachricht können den Wert 0 haben.

In der Regel ist m ein Vielfaches von 8, so dass sich M auch als Folge von Bytes darstellen lässt. Verfahren zur Kodierung von Nachrichten als Bit- oder Bytefolgen sind nicht Teil der unterstützten Signaturverfahren.

Falls ein Teil der zu signierenden Nachricht M als Bytefolge in dem DSI enthalten ist, kann dieser Teil durch die Klartextrückgewinnung des Signatur-Algorithmus aus der Signatur zurückgewonnen werden. In diesem Fall handelt es sich um ein Signaturverfahren ***mit Nachrichtenrückgewinnung. Der zurückgewinnbare Teil*** der Nachricht M wird als $\mathbf{M_r}$ bezeichnet. Falls die gesamte signierte Nachricht aus der Signatur zurückgewonnen werden kann, handelt es sich um ***vollständige Nachrichtenrückgewinnung*** ($M = M_r$), andernfalls um partielle Nachrichtenrückgewinnung. Der ***nicht zurückgewinnbare Teil*** der Nachricht M wird dann als $\boldsymbol{M_n}$ bezeichnet.

Falls der DSI keinen Teil der Nachricht als Bytefolge enthält, handelt es sich um ein Signaturverfahren ***ohne Nachrichtenrückgewinnung*** ($M = M_n$).

Zur Prüfung einer mit einem Signaturverfahren erzeugten digitalen Signatur wird außer der Signatur der nicht zurückgewinnbare Teil der signierten Nachricht benötigt.

Das im Folgenden beschriebene Signaturverfahren ist in [DINSIG] spezifiziert. Es ist ein Signaturverfahren ohne Nachrichtenrückgewinnung, das auf [ISO DS2] (beide Referenzen siehe Kapitel 14.2.5.5) basiert. Es verwendet zur Erzeugung des DSI

- einen Hash-Algorithmus und
- einen Format-Mechanismus.

Ein Hash-Algorithmus bildet Bitfolgen beliebiger Länge (***Eingabe-Bitfolgen***) auf Bytefolgen einer festen, durch den Hash-Algorithmus festgelegten Länge ab. Das Ergebnis der Anwendung eines Hash-Algorithmus auf eine Bitfolge wird als ***Hashwert*** bezeichnet.
Das zur Erzeugung digitaler Signaturen unterstützte Signaturverfahren verwendet den Hash-Algorithmus ***RIPEMD-160***.
Der Hash-Algorithmus RIPEMD-160 ist in [RIPEMD] und [ISO HF3] (siehe Kapitel 14.2.5.5) spezifiziert. RIPEMD-160 bildet Eingabe-Bitfolgen beliebiger Länge auf einen als Bytefolge dargestellten Hashwert von 20 Bytes Länge ab. Teil des Hash-Algorithmus ist das Padding von Eingabe-Bitfolgen auf ein Vielfaches von 64 Bytes. Das Padding erfolgt auch dann, wenn die Eingabe-Bitfolge bereits eine Länge hat, die ein Vielfaches von 64 Bytes ist. RIPEMD-160 verarbeitet die Eingabe-Bitfolgen in Blöcken von 64 Bytes Länge.
Der Hashwert einer Eingabe-Bitfolge x unter dem Hash-Algorithmus RIPEMD-160 wird wie folgt bezeichnet:

RIPEMD(x).

Im Folgenden wird der Format-Mechanismus des Signaturverfahrens gemäß Spezifikation [DINSIG] ((siehe Kapitel 14.2.5.5) beschrieben. Die verwendeten Abkürzungen sind in Kapitel 14.2.2 definiert.

Berechnen einer Signatur

Die folgenden Schritte werden gemäß Spezifikation [DINSIG] und [ISO DS2] ((siehe Kapitel 14.2.5.5) zum Berechnen einer Signatur zu der Nachricht M der Bit-Länge m ausgeführt.

- Der Hashwert RIPEMD(M) mit einer Länge von 160 Bit wird berechnet.
  Hierbei gehen betriebssystemabhängige Zeichen (*bei Windows CR, LF, CRLF und Control-Z*) in die Bildung des Hashwertes nicht ein.

- Der DSI wird erzeugt.
  Der DSI ist eine Folge von k Bit, der wie folgt aufgebaut ist:

Bezeichnung	Bit-Länge	Wert
Header	2	0 1
More-Data-Bit	1	1, da , M = M_n ist
Paddingfeld	k-235	k-236 Bit 0, gefolgt von einem Bit 1 (Grenzbit)
Datenfeld	64	Zufallszahl: Die Zufallszahl muss bei jeder Signaturberechnung dynamisch erzeugt und in den DSI eingestellt werden.
Hashwert	160	RIPEMD(M)
Trailer	8	'BC'

Die Nachricht M besteht komplett aus dem nicht zurückgewinnbaren Teil M_n.
Die ersten vier Bit des DSI können nur den Wert ‚6' annehmen, da k-236 > 0 ist.

- Aus dem DSI wird mit dem Algorithmus zur Signaturerzeugung gemäß Kapitel 14.2.3 eine Signatur berechnet.

  Hierbei ist zu beachten, dass der DSI als Folge von k Bit darstellbar ist, wobei das erste (höchstwertige) Bit den Wert 0 hat. Der sich aus der Binärdarstellung ergebende ganzzahlige Wert des DSI ist damit kleiner als 2^{k-1} und damit kleiner als der Wert des Modulus n.

  Ferner lässt sich der DSI als Bytefolge darstellen, ggf. indem maximal 7 Bit mit dem Wert 0 dem ersten Bit der Bitfolge vorangestellt werden. Diese Bytefolge hat denselben ganzzahligen Wert wie die den DSI darstellende Bitfolge.

  Die Signatur ist als Bytefolge darstellbar, deren Byte-Länge höchstens N ist. In der Darstellung des Modulus n als Bytefolge hat das Bit b_k den Wert 1 und die Bit b_{8*N} b_{8*N-1} ... b_{k+1} haben, sofern vorhanden, den Wert 0. In der Darstellung der Signatur als Bytefolge haben die Bit b_{8*N} b_{8*N-1} ... b_{k+1} ebenfalls den Wert 0.

Prüfen einer Signatur

Die folgenden Schritte werden gemäß Spezifikationen [DINSIG] und [ISO DS2] (siehe Kapitel

14.2.5.5) zum Prüfen einer Signatur ausgeführt. Hierzu müssen die zu prüfende Signatur s und die Nachricht M' der Bit-Länge m' vorliegen.

- Die Signatur muss als Bytefolge darstellbar sein, deren Byte-Länge höchstens N ist. In der Darstellung von s als Bytefolge müssen die Bit b_{8*N} b_{8*N-1} ... b_{k+1}, sofern vorhanden den Wert 0 haben. Ist das nicht der Fall, wird die Signatur abgewiesen.
  Der sich aus der Binärdarstellung von s ergebende ganzzahlige Wert muss zwischen 0 und n-1 liegen. Ist das nicht der Fall, wird die Signatur abgewiesen.

- Auf die Signatur wird der Algorithmus zur Klartextrückgewinnung gemäß Kapitel 14.2.3 angewandt. Das Ergebnis ist eine Bitfolge b_k ... b_1, die als DSI' bezeichnet wird. DSI' muss den folgenden Anforderungen genügen:

  Das niedrigstwertige Byte bestehend aus b_8 ... b_1 hat den Wert 'BC'.

  Das Bit b_{k-1} hat den Wert 1 und alle höherwertigen Bit haben den Wert 0.

  Das Bit b_{k-2} (‚More Data'-Bit) hat den Wert 1.

  Das Paddingfeld besteht aus (k – 236) Nullen und einer 1 (Grenzbit).

  Sind die Anforderungen nicht erfüllt, wird die Signatur abgewiesen.

- Aus dem DSI' wird ein Hashwert' entnommen. Der Hashwert' besteht aus den 160 Bit, die dem Trailer 'BC' vorausgehen.

- Der Hashwert RIPEMD(M') wird berechnet und mit dem Hashwert' verglichen. Falls die Werte gleich sind, war die Prüfung der Signatur erfolgreich. Andernfalls wird die Signatur abgelehnt.

Notation

Für die Berechnung einer Signatur zu einer Nachricht M mit dem Signaturverfahren gemäß [DINSIG], dem Signatur-Algorithmus RSA und dem privaten RSA-Schlüssel S_K wird die folgende Notation verwendet:

$sign_{DINSIG}(S_K)[M]$.

Für die Prüfung einer Signatur s zu einer Nachricht M mit dem Signaturverfahren gemäß [DINSIG], dem Signatur-Algorithmus RSA und dem öffentlichen RSA-Schlüssel P_K wird die folgende Notation verwendet:

$verify_{DINSIG}(P_K)[s,M]$.

14.2.5 Signaturformat A004

14.2.5.1 Signaturformat

Die Version A004 der Elektronischen Unterschrift basiert auf RSA-Signaturen, die mit Schlüsseln generiert werden, deren Moduli eine Länge von 1024 Bit haben. Die Signatur A004 unterstützt den RSA-Algorithmus mit beliebigen ungeraden öffentlichen Exponenten. In der Regel wird der ungerade öffentliche Exponent $F_4 = 2^{16}+1$ verwendet. Als Paddingformat wird das in der „DIN-Spezifikation der Schnittstelle zu Chipkarten mit digitaler Signatur-Anwendung/Funktion nach SigG und SigV“ beschriebene Paddingformat „ISO 9796 Part 2 mit Zufallszahl“ verwendet.

14.2.5.2 Ermittlung des Hashwertes über die zu unterschreibende Datei

Das in der Version A004 der Elektronischen Unterschrift angewandte Signaturverfahren verwendet den Hash-Algorithmus RIPEMD-160.

RIPEMD-160 bildet Eingabe-Bitfolgen beliebiger Länge auf einen als Bytefolge dargestellten Hashwert von 20 Bytes Länge ab. Teil des Hash-Algorithmus ist das Padding von Eingabe-Bitfolgen auf ein Vielfaches von 64 Bytes. Das Padding erfolgt auch dann, wenn die Eingabe-Bitfolge bereits eine Länge hat, die ein Vielfaches von 64 Bytes ist. RIPEMD-160 verarbeitet die Eingabe-Bitfolgen in Blöcken von 64 Bytes Länge. Das Padding der Nachricht auf die entsprechende Blockgröße wird in der Beschreibung des Hashverfahrens spezifiziert. Der zu verwendende Initialisierungsvektor ist ebenfalls in der Beschreibung des Hashverfahrens festgelegt.

Der im Initialisierungsbrief angegebene Hashwert wird ebenfalls nach diesem Verfahren über die 256 Bytes – bestehend aus öffentlichem Exponenten und Modulus – berechnet.

14.2.5.3 Aufbau der Unterschriftsdatei

Vor dem Versand einer Elektronischen Unterschrift wird diese in eine separat zu übertragende Unterschriftsdatei eingestellt, die den folgenden Aufbau hat:

Inhalt	Länge in Bytes	Daten-format[2]	Belegung	Erläuterung
Versionsnummer	4	an	‚A004'	
Länge des Modulus	4	n	‚1024'	
Auftragsart	3	an	z.B. 'IZV'	Auftragskürzel der Originaldatei
EU	128	binär	'0, ..., 0, SIGNATUR'	rechtsbündig
User-ID	8	an	z.B. 'A2B2C2D2'	
Originaldatei	128	an		Lokaler Dateiname der Originaldatei
Datum/Uhrzeit der Dateierstellung	16	an	jjjjmmttX'20'hhmmssX'20'	
Datum/Uhrzeit der Unterschrift	16	an	jjjjmmttX'20'hhmmssX'20'	
Frei nutzbares Feld	8	binär	X'00'	Zur Zeit nicht benutzt
Reserve	197	binär	X'00'	Zur Zeit nicht benutzt

Das Feld „Originaldatei" muss nicht gefüllt sein. Eine Prüfung findet bei der EU-Prüfung nicht statt.
Das Feld Datum/Uhrzeit der Dateierstellung ist grundsätzlich mit den entsprechenden Zeitangaben zu befüllen. Im Falle der VEU sind die u.U. nicht mehr verfügbar. In diesem Fall ist das Feld mit Blanks zu füllen. Eine Prüfung findet bei der EU-Prüfung nicht statt.

14.2.5.4 Aufbau der Public-Key-Datei (INI-Datei)

Inhalt	Anzahl Bytes	Belegung / Erläuterung
Versionsnummer	4 ASCII	'A004' Dieses Feld dient zur Kennzeichnung des verwendeten EU-Verfahrens (A004).
User-ID	8 ASCII	'A2B2C2D2' Diese Feld enthält die institutsabhängige User-ID. Dies hat zur Folge, dass das Kundensystem vor dem Versenden der Public Key-Datei die entsprechende User-ID eintragen muss.

[2] an = alphanummerisch; n = nummerisch; Werte im ASCII-Format werden linksbündig eingestellt und rechts mit Blanks (X'20') aufgefüllt. Werte im Binär-Format werden rechtsbündig eingestellt und links mit X'00' aufgefüllt.

Inhalt	Anzahl Bytes	Belegung / Erläuterung
LExponent	4 ASCII	‘1024’, Länge des Exponenten, Wert in Bit
Exponent	128 binär	00...010001 (Hex) Dieses Feld kann Werte bis maximal 1024 Bit enthalten.
LModulo	4 ASCII	‘1024’, Länge von Modulo, Wert i n Bit
Modulo	128 binär	0...bc7bdc...87 Dieses Feld kann Werte bis maximal 1024 Bit enthalten.
Reserve	236 ASCII	Auffüllen mit X’20’

Werte im ASCII-Format werden linksbündig eingestellt und rechts mit Blanks X’20’ aufgefüllt.
Werte im Binär-Format werden rechtsbündig eingestellt und links mit X’00’ aufgefüllt.

14.2.5.5 Referenzen

[DINSIG] DIN-Spezifikation der Schnittstelle zu Chipkarten mit Digitaler Signatur-Anwendung/Funktion nach SigG und SigV, DIN NI-17.4, Version 1.0, 15.12.1998

[EMV B2] EMV2000, Integrated Circuit Card Specification for Payment Systems, Book 2, Security and Key Management, Version 4.0, EMVCo, December 2000

[ISO DS2] ISO 9796 - 2, Information technology - Security techniques - Digital signature scheme giving message recovery, Part 2: Mechanisms using a hash function, 1997

[ISO HF3] ISO 10118 - 3, Information technology - Security techniques - Hash-functions, Part 3: Dedicated hash functions, 1998

[PKCS1] PKCS #1: RSA Cryptography Standard, Version 2.0, 1.10.1998

[RIPEMD] H. Dobbertin, A. Bosselaers, B. Preneel, RIPEMD-160: A strengthened version of RIPEMD, 1996

[RSA] R. L. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public key cryptosystems, Communications of the ACM, vol. 1, n. 2, 1978, 120-126

15 Anhang: Verschlüsselungsverfahren V001

15.1 Vorgänge beim Sender

Generierung des geheimen DES-Schlüssels (2-Key-Triple-DES)

Es werden zwei zufällige Bitstrings **DEK_{left}** und **DEK_{right}** mit jeweils einer Länge von 64 Bit generiert. Die Verknüpfung von DEK_{left} und DEK_{right} wird als DEK bezeichnet.
Es sei

$$DEK = DEK_{left} || DEK_{right} = x_{127}, \ldots, x_o$$

mit $DEK_{left} = x_{127}, \ldots, x_{64}$ und $DEK_{right} = x_{63}, \ldots, x_{o.}$

Bei der Interpretation der DES-Keys als natürliche Zahl wird angenommen, dass das jeweils linke Bit (x_{127} bzw. x_{63}) der Schlüssel als höchstwertigstes Bit der Zahl aufgefaßt wird.

Prüfen der geheimen DES-Schlüssel

Die generierten Zufallszahlen, die als rechte und linke Schlüsselhälfte des 2-Key-Triple-DES verwendet werden, sind daraufhin zu überprüfen, dass es sich nicht um einen schwachen oder halbschwachen DES-Schlüssel handelt

Die schwachen Schlüssel des DES							
01	01	01	01	01	01	01	01
FE	FE	FE	FE	FE	FE	FE	FE
1F	1F	1F	1F	0E	0E	0E	0E
E0	E0	E0	E0	F1	F1	F1	F1
Die halbschwachen Schlüssel des DES							
01	FE	01	FE	01	FE	01	FE
FE	01	FE	01	FE	01	FE	01
1F	E0	1F	E0	0E	F1	0E	F1
E0	1F	E0	1F	F1	0E	F1	0E
01	E0	01	E0	01	F1	01	F1
E0	01	E0	01	F1	01	F1	01
1F	FE	1F	FE	0E	FE	0E	FE
FE	1F	FE	1F	FE	0E	FE	0E
01	1F	01	1F	01	0E	01	0E
1F	01	1F	01	0E	01	0E	01
E0	FE	E0	FE	F1	FE	F1	FE
FE	E0	FE	E0	FE	F1	FE	F1

Vorbereitung zur Verschlüsselung von DEK

Der als natürliche Zahl interpretierte 128 Bit DEK wird vor dem höchstwertigsten Bit mit Nullbits auf 768 Bit aufgefüllt. Das Ergebnis heißt **PDEK**.

Verschlüsselung des geheimen DES-Schlüssels

PDEK wird anschließend mit dem öffentlichen Schlüssel des RSA-Schlüsselsystems des Empfängers verschlüsselt und anschließend mit führenden Nullbits auf 1024 Bit erweitert.

Das Ergebnis heißt **EDEK**. Es muss sichergestellt sein, dass EDEK ungleich DEK ist.

Verschlüsselung der Nachrichten

Padding der Nachricht:
Für das Padding der Nachricht wird die Methode **Padding with Octets** nach ANSI X9.23 angewendet, d. h. es werden in jedem Fall Daten an die zu verschlüsselnde Nachricht angefügt.

Anwendung des Verschlüsselungsalgorithmus:
Die Nachricht wird mit dem geheimen Schlüssel DEK nach dem 2-Key-Triple-DES-Verfahren, wie im ANSI X3.92-1981 spezifiziert, im CBC-Mode gemäß ANSI X3.106 verschlüsselt.

Hierbei wird folgender Initialisierungswert „ICV“ verwendet: X ‘00 00 00 00 00 00 00 00’.

15.2 Vorgänge beim Empfänger

Entschlüsselung des DES-Schlüssels
Die führenden 256 Nullbits des EDEK werden entfernt und die verbleibenden 768 Bit mit dem geheimen Schlüssel des RSA-Schlüsselsystems des Empfängers entschlüsselt. Anschließend liegt PDEK vor. Aus den niederwertigsten 128 Bits von PDEK erhält man den geheimen DES-Schlüssel DEK, der in die Einzelschlüssel DEK_{left} und DEK_{right} aufgesplittet wird.

Entschlüsselung der Nachricht
Mit dem geheimen DES-Schlüssel (bestehend aus DEK_{left} und DEK_{right}) wird die verschlüsselte Originalnachricht nach dem 2-Key-Triple-DES-Verfahren im CBC-Mode entschlüsselt. Hierbei wird wiederum der Initialisierungswert ICV verwendet.

Entfernen der Paddinginformationen
Für das Entfernen der Paddinginformationen aus der entschlüsselten Nachricht wird die

Methode „Padding with Octets“ nach ANSI X9.23 angewendet. Anschließend liegt die Originalnachricht unverschlüsselt vor.

16 Anhang: Standards und Verweise

Im EBICS-Feinkonzept wird Bezug auf einige Verfahren, Algorithmen und Formatfestlegungen genommen.

Die zugehörigen Standard-Dokumentkennungen sowie Links zu den referenzierten Dokumenten werden im Folgenden aufgeführt.

Standard	Eigenschaft	Standard-kennung	Verweis
EBICS	multibankfähige Schnittstelle zur internetbasierten Kommunikation	H004	http://www.ebics.org, Kategorie „Spezifikation“ (XML-Schema)
ZIP	universeller Komprimierungs-algorithmus	RFC 1950, RFC 1951	http://www.ietf.org/rfc/rfc1950.txt http://www.ietf.org/rfc/rfc1951.txt
base64	Kodierungsformat zum textuellen Bytecode-Transport	RFC 1421, RFC 2045	http://www.ietf.org/rfc/rfc1421.txt http://www.ietf.org/rfc/rfc2045.txt
UTF-8	Kodierungsformat für Unicode-Zeichen	RFC 3629 (ISO 10646)	http://www.ietf.org/rfc/rfc3629.txt
HTTP 1.1	Internet-Anwendungs-protokoll	RFC 2616	http://www.ietf.org/rfc/rfc2616.txt
TLS	Transportschicht-Verschlüsselung	RFC 2246, RFC 3268 (+AES), RFC 2818 (HTTP über TLS)	http://www.ietf.org/rfc/rfc2246.txt http://www.ietf.org/rfc/rfc3268.txt http://www.ietf.org/rfc/rfc2818.txt
TCP	Internet-Übertragungs-protokoll	RFC 793	http://www.ietf.org/rfc/rfc793.txt
IP(v4)	Internet-Netzwerkprotokoll	RFC 791	http://www.ietf.org/rfc/rfc791.txt
XML	hierarchische Dokumenten-sprache	(W3C-Rec.)	http://www.w3.org/TR/REC-xml/
XML-Signature	Verfahren zur digitalen Signatur	RFC 3275	http://www.ietf.org/rfc/rfc3275.txt http://www.w3.org/TR/xmldsig-core/

	von XML-Dokumenten		
X.509v3	Format und Profil für PKI-Zertifikatsdaten	RFC 3280	http://www.ietf.org/rfc/rfc3280.txt
Länder-codes	Format für Länderkürzel	RFC 1766, ISO 639	http://www.ietf.org/rfc/rfc1766.txt
Zeitstempel	Format für Datums- und Zeitstempel	ISO 8601 (2004)	http://www.iso.org/iso/en/CatalogueDetail»Page.CatalogueDetail?CSNUMBER=40874
SHA-1	Hashalgorithmus	RFC 3174, FIPS 180-2 (SHA allg.)	http://www.ietf.org/rfc/rfc3174.txt http://csrc.nist.gov/publications/fips/fips»180-2/fips180-2withchangenotice.pdf
Triple-DES, 3DES	symmetrischer Verschlüs-selungs-algorithmus	FIPS 46-3	http://csrc.nist.gov/publications/fips/fips46-»3/fips46-3.pdf
AES	symmetrischer Verschlüsse-lungsalgorithmus	FIPS 197	http://csrc.nist.gov/publications/fips/fips197/»fips-197.pdf

17 Anhang: Glossar

3DES	Triple-DES, eine Variante des „Digital Encryption Standard“. DES ist ein symmetrischer Verschlüsselungsalgorithmus mit einer Nutzschlüssellänge von 56 bit. Um dem heutigen Sicherheitsbedürfnis nachzukommen, wird bei 3DES der Algorithmus dreimal hintereinander auf dem gleichen Datenblock ausgeführt und ein Schlüssel mit einer Nutzlänge von 112 bit (2-Key-3DES) oder 168 bit (3-Key-3DES) verwendet. Im EBICS-Kontext findet 3DES Anwendung bei TLS (gemäß RFC 2246).
AES	„Advanced Encryption Standard“: ein symmetrischer Verschlüsselungsalgorithmus, der DES ablösen soll. Im EBICS-Kontext findet AES Anwendung bei TLS sowie bei der Verschlüsselung bankfachlicher Nutzdaten (gemäß RFC 3268).
Auftrag	Bankfachlicher oder systembedingter Geschäftsvorfall, dessen Typ über die so genannte Auftragsart identifiziert wird.
Auftragsart	Dreistelliger alphanummerischer Code, der die Art des Auftrags kennzeichnet. Die standardisierten, systembedingten und reservierten Auftragsarten sind im Anhang (Kapitel 13) und dem Document “EBICS Anhang 2 Auftragsartenkennungen”aufgelistet. Neue EBICS-Auftragsarten sind am ersten Buchstaben „H“ erkennbar. Siehe hierzu auch Kapitel 3.10.
Auftragsattribute	Fünfstelliger alphanumerischer Code, der zu einer EBICS-Transaktion Informationen über die Art der übermittelten Daten (EUs zu einem bestehenden Auftrag, Auftragsdaten zusammen mit bankfachlichen EUs, Auftragsdaten mit Transportunterschrift) sowie über die Art der Komprimierung und der Verschlüsselung dieser Daten enthält. Siehe Anhang (Kapitel 12.3).
Auftragsdaten	siehe „bankfachliche Nutzdaten“
Auftragsnummer	Eindeutige, durch den Bankrechner vergebene Auftragsnummer, die dem Kunden(system) mittels Upload-Response mitgeteilt wird. Sie dient insbesondere zur Synchronisation von Auftragsdaten und elektronischen Unterschriften. Durch die Anwendung ist sicherzustellen, dass pro Kunden-ID und pro Auftragsart eindeutige Auftragsnummern vergeben werden. Aufbau der 4-stelligen Auftragsnummer: Erste Stelle: Alphazeichen (A-Z) frei wählbar Restliche 3 Stellen: Alphanummerisch (A-Z oder 0-9) hochzählen
Auftrags-parameter	Zusätzliche Parameter zum Auftrag, die der Client im ersten Transaktionsschritt an den Server übermittelt. Siehe Kapitel 3.11.
Authentifikations-schlüssel (öffentlich/privat)	RSA-Schlüsselpaar, dessen privater Schlüssel zur Bildung der Authentifkationssignatur und dessen öffentlicher Schlüssel zu deren Prüfung eingesetzt wird.
Authentifikations-	Digitale Signatur zur Sicherung der Authentizität der Steuerdaten einer

signatur	EBICS-Nachricht. Als Signaturformat kommt XML-Signature zur Anwendung.
bankfachliche Elektronische Unterschrift	EU eines Teilnehmers der Unterschriftsklasse „E“, „A“ oder „B“, mit der die Durchführung eines Auftrags autorisiert wird.
bankfachliche Nutzdaten	Daten, die für die Durchführung eines Auftrags benötigt werden. Das Format dieser Daten ist abhängig von der Auftragsart. Die Mehrheit der Datenformate, die in EBICS verwendet werden, ist bereits definiert. Die Datenformate der für EBICS neu definierten Auftragsarten (wie etwa Auftragsarten für die Verteilte Elektronische Unterschrift) sind in EBICS mittels XML-Schema definiert. Die Nutzdaten eines Auftrags werden transparent (in komprimierter verschlüsselter Form) in EBICS-Nachrichten eingebettet.
bankfachlicher Schlüssel (öffentlich/privat)	RSA-Schlüsselpaar, dessen privater Schlüssel zur Bildung der bankfachlichen Elektronischen Unterschrift und dessen öffentlicher Schlüssel zu deren Verifikation verwendet wird.
bankfachliches Zielsystem	Komponente in der Verantwortung des Kreditinstituts, die für die Kunden-/ Teilnehmerverwaltung sowie die Durchführung bankfachlicher Aufträge zuständig ist. Im Rahmen der EBICS-Spezifikation gilt das bankfachliche Zielsystem als „sichere Blackbox“.
Banksystem	Komponenten in der Verantwortung des Kreditinstituts, die an der Durchführung einer EBICS-Transaktion beteiligt sind. Dies umfasst sowohl das bankfachliche Zielsystem als auch den/die HTTP-Server, die die EBICS-Nachricht empfangen und an das bankfachliche Zielsystem weiterreichen.
base64	Kodierungsalgorithmus und -format gemäß RFCs 1421 & 2045. Das Resultat eines base64-Kodierungsdurchlaufs ist komplett in ASCII darstellbar.
CA	Abkürzung für Certificate Authority (Zertifizierungsinstanz)
Client	Kommunikationseinheit, die EBICS-Requests verschickt und EBICS-Responses empfängt. Siehe auch „Kundensystem“.
Download-Transaktion	EBICS-Transaktion zur Übermittlung eines Abholauftrags. Die Transaktionsphasen einer Download-Transaktion sind: Transaktionsinitialisierung, Datentransfer, Quittierung der Abholdaten.
EBICS-Nachricht	EBICS-Request eines Teilnehmers oder EBICS-Response des Kreditinstituts. EBICS-Nachrichten setzen sich im Wesentlichen aus Steuerdaten, der Authentifikationssignatur und aus bankfachlichen Daten zusammen.
EBICS-Request	Request eines Teilnehmers im XML-Format, das in EBICS definiert wurde.
EBICS-Response	Response des Kreditinstituts im XML-Format, das in EBICS definiert wurde.
EBICS-Transaktion	Sequenzielle Abfolge von EBICS-Transaktionsphasen, die notwendig sind, um einen Auftrag an das bankfachliche Zielsystem zu

	übermitteln**.** EBICS-Transaktionen können Upload- oder Download-Transaktionen sein.
EBICS-Transaktions-phase	Abfolge zusammengehöriger EBICS-Transaktionsschritte. Bei EBICS werden folgende Transaktionsphasen unterschieden: Transaktionsinitialisierung („initialisation"), Datentransfer („transfer") und Quittierung („receipt").
EBICS-Transaktions-schritt	Paar aus EBICS-Request und zugehöriger EBICS-Response. Ein EBICS-Request wird stets vom Kundensystem initiiert.
EBICS-Transaktions-verwaltung	Komponente des Banksystems, deren Verantwortlichkeit die Verwaltung von EBICS-Transaktionen ist.
Elektronische Unterschrift (EU)	Willentliche Signatur der Auftragsdaten eines Sendeauftrags durch einen Teilnehmer, mit welcher der entsprechende Auftrag eingereicht oder autorisiert werden kann, oder auch Signatur des Kreditinstituts über Abholdaten. In EBICS werden EUs gemäß Anhang (Kapitel 14) verwendet, die mindestens nach dem Verfahren A004 gebildet sind
EU	siehe „Elektronische Unterschrift"
EU-Signatur-schlüssel	siehe „bankfachlicher Schlüssel (öffentlich/privat)".
Host-ID	EBICS Host-ID zur Indentifikation des EBICS Bankrechners in jeder Request-Nachricht des Kundensystems. Sie muss nicht identisch sein zur Host-ID für das FTAM-Verfahren. Das Kreditinstitut teilt dem Kunden die EBICS Host-ID zusammen mit der URL des Bankzugangs mit.
Kunde	Organisatorische Einheit (Unternehmen oder Privatperson), die mit dem Kreditinstitut seinen Vertrag abschließt. Der Vertrag legt fest, welche Geschäftsvorfälle der Kunde mit dem Kreditinstitut abwickelt, welche Konten betroffen sind, welche Teilnehmer des Kunden mit dem Banksystem arbeiten und welche Berechtigungen die Teilnehmer besitzen.
Kundensystem	Komponente, welche von Teilnehmern verwendet wird, um Aufträge an ein Kreditinstitut zu senden und Information zu Aufträgen oder Teilnehmerkonten vom Kreditinstitut zu erhalten.
OrderAttribute	siehe „Auftragsattribute".
OrderData	siehe „Auftragsart".
Partner	siehe „Kunde".
Schlüssel-verwaltung	Komponente des Banksystems, die für Zuordnung öffentlicher Schlüssel zu Teilnehmern zuständig ist und den Zugriff auf die von ihr verwalteten Schlüssel kontrolliert.
Segmentierung	Aufteilung des Datenvolumens der Auftragsdaten nach Komprimierung, Verschlüsselung und base64-Kodierung in Segmente von max. 1 MB Größe. Siehe auch Kapitel 7.
Server	Kommunikationseinheit, die EBICS-Requests empfängt und EBICS-

	Responses verschickt. Siehe auch „Banksystem“.
Steuerdaten	Daten einer EBICS-Nachricht, die für die Steuerung des Ablaufs einer EBICS-Transaktion benötigt werden. Es handelt sich dabei sowohl um Daten für die Authentisierung des Teilnehmers gegenüber dem bankfachlichen Zielsystem, als auch um Daten zur Identifikation des nächsten auszuführenden Transaktionsschritts oder um technische Returncodes, als auch um Auftragsparameter, Daten zur Vorabprüfung eines Auftrags oder bankfachliche Returncodes.
Teilnehmer	Menschlicher Nutzer („nichttechnischer Teilnehmer“) oder technisches System („technischer Teilnehmer“), der/das einem Kunden zugeordnet ist. Wird über die Kombination aus TeilnehmerID und KundenID identifiziert. Der technische Teilnehmer dient dem Datenaustausch zwischen einem Kunden und einem Kreditinstitut und ist nicht gleichzusetzen mit einer technischen ID für Service-Dienstleister.
Teilnehmer-initialisierung	Verfahren, nach welchem die öffentlichen Teilnehmerschlüssel dem Kreditinstitut übermittelt werden und anschließend vom Kreditinstitut freigegeben werden. Nach dem erfolgreichen Durchlauf der Teilnehmerinitialisierung haben die Teilnehmer im Banksystem den Zustand „Bereit“.
TLS	„Transport Layer Security“: Protokoll gemäß RFCs 2246 & 3268 zur kryptographischen Absicherung von Nachrichten, die TCP/IP als Übertragungsprotokoll benutzen. Im EBICS-Kontext findet TLS als Transportverschlüsselung für HTTP-Nachrichten (HTTPS) Anwendung.
Transaktion	siehe „EBICS-Transaktion“.
Transaktions-phase	siehe „EBICS-Transaktionsphase“.
Transaktions-schlüssel	Symmetrischer Schlüssel, der innerhalb einer EBICS-Transaktion zur Verschlüsselung bankfachlicher Daten verwendet wird.
Transaktions-schritt	siehe „EBICS-Transaktionsschritt“.
Transaktions-verwaltung	siehe „EBICS-Transaktionsverwaltung“.
Transport-unterschrift (TEU)	EU eines Teilnehmers der Unterschriftsklasse „T“, mit der ein Auftrag eingereicht (und nicht seine Durchführung autorisiert) wird.
Unterschrifts-klasse	Bezieht sich auf EUs von Teilnehmern. EBICS definiert die folgenden Unterschriftsklassen: Einzelunterschrift (Typ „E“), Erstunterschrift (Typ „A“), Zweitunterschrift (Typ „B“), Transportunterschrift (Typ „T“). Details siehe Kapitel 3.5.1.
Upload-Transaktion	EBICS-Transaktion zur Übermittlung eines Sendeauftrags. Die Transaktionsphasen einer Upload-Transaktion sind: Transaktionsinitialisierung, Datentransfer.

UTF-8	„Unicode Transformation Format“, ein Zeichenkodierungsstandard gemäß RFC 3629.
Verschlüsselungs-schlüssel (öffentlich/privat)	RSA-Schlüsselpaar, dessen öffentlicher Schlüssel vom Kommunikationspartner zur Verschlüsselung des symmetrischen Transaktionsschlüssels und dessen privater Schlüssel vom Inhaber zur Entschlüsselung desselben Transaktionsschlüssels eingesetzt wird.
Verteilte bankfachliche Unterschrift	siehe „Verteilte Elektronische Unterschrift“.
Verteilte Elektronische Unterschrift	Verfahren, mit dem bankfachliche Elektronische Unterschriften zu einem bestimmten Auftrag orts- und zeitunabhängig voneinander geleistet werden können. Siehe hierzu Kapitel 8. Die Abkürzung lautet „VEU“.
Vertrauensanker	Im Kontext der Zertifikatsprüfung bezeichnet ein Vertrauensanker (Point of Trust) ein Zertifikat, das als vertrauenswürdig gilt. Normalerweise handelt es sich hier um das Zertifikat einer CA (Certification Authority).
VEU	siehe „Verteilte Elektronische Unterschrift“.
ZIP	Verlustfreier Komprimierungsalgorithmus gemäß RFCs 1950 und 1951.

18 Abbildungsverzeichnis