Elektronische Gesundheitskarte und Telematikinfrastruktur

Errata 2 zum Konnektor PTV 3 (eMP/AMTS, NFDM)

Version:	1.0.0
Stand:	06.06.2019
Status:	freigegeben
Klassifizierung:	öffentlich
Referenzierung:	[gemErrata_2_Kon_PTV3]

ID	Dokument	Quelle Dokument und/oder Kapitel	Beschreibung der Änderung	Anpassungen an Afos; TUCs, Tabellen, Korrekturen	von Änderung betroffene Dokumente
C_6754	gemRL_QES_NFDM gemSpec_Kon	TAB_06_SR_DF_NFDM_NOTFALLDATEN TAB_KON_065	**Korrektur Signaturrichtlinie NFDM** Es gibt drei Probleme, die im Zusammenhang mit der Adressierung von XML-Elementen bei der Platzierung von Signaturen in XML-Elementen auftreten: 1. Der XPath-Ausdruck in der NFDM-Signaturrichtlinie berücksichtigt nicht den Namespace der zu selektierenden Knoten. Da die Schnittstelle keine Möglichkeit vorsieht, einen Default-Namespace für den XPATH-Ausdruck vorzugeben, wird der Ausdruck angepasst, so dass der Namespacekontext ausgeblendet wird. Damit ist der Ausdruck auch robust gegenüber möglichen Namespace-Prefixes. 2. Die Spezifikation [OASIS-DSS] verlangt eine Möglichkeit zur Zuordnung von Namespace-Prefixes zu Namespaces, die nicht der üblichen Umsetzungspraxis in Frameworks entspricht. 3. Damit Pfadausdruck im Ausdruck VerifyDocument dss:Signatur identifiziert muss er das Element Signatur umfassen.	In gemRL_QES_NFDM werden folgende Werte geändert: in Tabelle TAB_06_SR_DF_NFDM_NOTFALLDATEN der Wert zu /SIG:SignDocument/SIG:SignRequest/SIG:OptionalInputs/dss:SignaturePlacement/XPathFirstChildOf alt: "/NFD_Document/SignatureArzt" neu: "/*[local-name()='NFD_Document']/*[local-name()='SignatureArzt']" in Tabelle TAB_07_SR_DV_NFDM_NOTFALLDATEN der Wert zu /SIG:VerifyDocument/dss:SignatureObject/ds:SignaturePtr/@XPath alt: "/NFD_Document/SignatureArzt" neu: "/*[local-name()='NFD_Document']/*[local-name()='SignatureArzt']/*[local-name()='Signature']" In gemSpec_Kon wird in der Tabelle TAB_KON_065 die Beschreibung zu dss: SignaturePlacement geändert: alt: Durch dieses in [OASIS-DSS] (Abschnitt 3.5.8) definierte Element kann bei XML-basierten Signaturen gemäß [RFC3275] die Platzierung der Signatur im Dokument angegeben werden. Bei anderen Signaturtypen wird das Element ignoriert und eine Warnung (Fehlercode 4197, Parameter SignaturePlacement wurde ignoriert) zurückgeliefert. neu: Durch dieses in [OASIS-DSS] (Abschnitt 3.5.8) definierte Element kann bei XML-basierten Signaturen gemäß [RFC3275] die Platzierung der Signatur im Dokument angegeben werden. Die in [OASIS-DSS] (Abschnitt 2.5, XPath c) beschriebene Deklaration von Namespace-Prefixes im dss:SignaturePlacement-Element wird nicht unterstützt. Bei anderen Signaturtypen wird das Element ignoriert und eine Warnung (Fehlercode 4197, Parameter SignaturePlacement wurde ignoriert) zurückgeliefert.	gemRL_QES_NFDM_V1.2.0 gemSpec_Kon_V5.4.0 gemProdT_Kon_PTV3
C_6820	gemSpec_Kon	TIP1-A_5447	**QES-XAdES-Signatur nur mit Signaturrichtlinie** Die aktuell spezifizierte Funktion zur Signaturerstellung und -prüfung ist nach Einschätzung des BSI nicht zertifizierbar. Um den Zeitplan für die Zertifizierung und Zulassung des PTV3-Konnektors abzusichern, wird der Funktionsumfang der qualifizierten elektronischen Signatur von XML-Daten nach XAdES auf bekannte Signaturrichtlinien eingeschränkt.	siehe C_6820_C_6890_Anlage	gemSpec_Kon_V5.4.0 gemKPT_Arch_TIP_V2.5.0
C_6827	gemSpec_InfoNFDM	3.4.1.9 element "NFDM:Geschlecht", 5.5.1.8 element "NFDM:Geschlecht"	**Hinzufügen des dritten Geschlechts "divers" - NFDM** Entsprechend der Änderung des Personenstandsgesetzes (PStG) wird eine Ergänzung der zulässigen Werte für das Element "Geschlecht" notwendig. Eine Erweiterung des genutzten Wertebereiches für das dritte Geschlecht soll wie folgt umgesetzt werden: * Die Erweiterung des genutzten Wertebereiches erfolgt mit dem Wert „D“. * Die Beschreibung des Wertebereiches für das Merkmal Geschlecht um "D“ = divers wird erweitert. Die fachliche Notwendigkeit zur Änderung des NFDM-Schema wird außerdem dafür genutzt, die Kodierung der Schema-Datei von ISO-8859-15 auf UTF-8 zu ändern. Das XML des NFD selbst bleibt bei ISO-8859-15. Mit dieser Änderung werden dann die Info-Modelle von VSDM, eMP/AMTS und NFDM einheitlich in UTF-8 dargestellt.	Übersicht * Einführung der Versionierung im Dateinamen für NFD_Common, d.h. Umbenennung nach NFDM_COMMON_v1_1.xsd * Neue Version der Dateien zum Info-Modell für NFDM: ** Umbenennung NFD_Document_v1_3.xsd => NFD_Document_v1_4.xsd ** alte Versionen werden entfernt * Analoge Änderungen und Anpassung der Referenzen in gemSpec_InfoNFDM In der NFDM_Common_v1_1.xsd (und Referenzen in NFD_Document_v1_4.xsd): neu: <xs:element name="Geschlecht"> <xs:simpleType> <xs:restriction base="xs:string"> <xs:maxLength value="1"/> <xs:pattern value="M|W|D|X"/> </xs:restriction> </xs:simpleType> </xs:element>	gemSpec_InfoNFDM_V1.4.0 gemProdT_Kon_PTV3 NFDM_Common.xsd NFD_Document_v1_3.xsd
C_6828	gemSpec_FM_AMTS	attribute MP/P/@s	**Hinzufügen des dritten Geschlechts "divers" - AMTS** Entsprechend der Änderung des Personenstandsgesetzes (PStG) wird eine Ergänzung der zulässigen Werte für das Element "Geschlecht" notwendig. Eine Erweiterung des genutzten Wertebereiches für das dritte Geschlecht soll wie folgt umgesetzt werden * Die Erweiterung des genutzten Wertebereiches erfolgt mit dem Wert „D“. * Die Beschreibung des Wertebereiches für das Merkmal Geschlecht um "D“ = divers wird erweitert.	**Übersicht** * Neue Version der Dateien zum Info-Modell für AMTS: ** Umbenennung AMTS_Document_v1_4.xsd => AMTS_Document_v1_5.xsd ** alte Version wird entfernt * Analoge Änderungen und Anpassung der Referenzen in gemSpec_Info_AMTS **In AMTS_Document_v1_5.xsd**, Attribut "Administrative Geschlecht des Versicherten" (+ Dok. in den Annotation): neu: <xs:enumeration value="D"> <xs:annotation> <xs:documentation>divers</xs:documentation> </xs:annotation> </xs:enumeration>	gemSpec_Info_AMTS_V1.3.0 gemProdT_Kon_PTV3 AMTS_Document_v1_4.xsd

ID	Dokument	Quelle Dokument und/oder Kapitel	Beschreibung der Änderung	**Anpassungen** an Afos; TUCs, Tabellen, Korrekturen	**von Änderung betroffene Dokumente**
C_6834	IdpServiceActiveRequestor.wsdl LocalIdpService.wsdl		**Korrektur Service und targetNamespace TBAuth** Die Spezifikation sieht für die beiden SOAP-Services von TBAuth spezielle targetNamespaces vor. Diese werden in den wsdl jedoch nicht verwendet. Beide WSDL verwenden den gleichen Servicenamen, was zu Konflikten in der Implementierung führt. Beide Fehler werden korrigiert.	**Alt:** targetNamespace=http://docs.oasis-open.org/ws-sx/ws-trust/200512/ <wsdl:service name="SecurityTokenService"> **Neu:** IdpServiceActiveRequestor.wsdl: targetNamespace=http://ws.gematik.de/conn/tbauth/IdpServiceActiveRequestor/v1.0 <wsdl:service name="IdpServiceActiveRequestor"> LocalIdpService.wsdl: targetNamespace=http://ws.gematik.de/conn/tbauth/LocalIdpService/v1.0 <wsdl:service name="LocalIdpService">	IdpServiceActiveRequestor.wsdl LocalIdpService.wsdl gemProdT_Kon_PTV3
C_6835	gemILF_PS_AMTS		**Identifizierendes Merkmal für Zahnärzte im eMP** Die PS-Anforderung AMTS-A_2272 verlangt das Befüllen eines der Attribute "LANR", "IDF" oder "KH-IK". Primärsysteme der Zahnärzte sollen eine zahnarztspezifische Bildungsregel verwenden, um über eine Nummer zu verfügen, die sie in das Attribut "MP/A/@lanr" eintragen. Diese Nutzung von MP/A/@lanr wird in gemSpec_INFO_AMTS übernommen.	*Alt:* AMTS-A_2272 - Entweder nur LANR, LEAN, IDF oder KH-IK Das Primärsystem MUSS gewährleisten, dass entweder nur die lebenslange Arztnummer, die Apotheken-IDF oder das Krankenhaus–IK gesetzt wird.<= *Neu:* AMTS-A_2272 - Identifikationsmerkmale LANR, IDF oder KH-IK Das Primärsystem MUSS gewährleisten, dass zur Identifikation des den eMP aktualisierenden Akteurs entweder nur die lebenslange Arztnummer, die Apotheken-IDF oder das Krankenhaus–IK gesetzt wird. Zahnärztliche Primärsysteme verwenden zur Identifikation des zahnärztlichen Akteurs das Attribut lanr mit der folgenden Bildungsregel: „0“ + KZV-Nr + Abrechnungsnummer der Zahnarztpraxis (ggf. ergänzt um eine oder mehrere führende Nullen). Bei der Konkatenation der Nummern muss nach der „0“ die KZV-Nummer des Akteurs an die Stellen 2-3 gesetzt werden (ggf. mit ihrer führenden Null), und an die Stellen 4-9 die 1 - 6-stellige Abrechnungsnummer der Zahnarztpraxis, wobei sie ggf. mit einer oder mehreren führenden Nullen aufgefüllt werden muss. Beispielresultat: „004006789“<=	gemILF_PS_AMTS_V1.4.0 gemSpec_INFO_AMTS_V1.3.0
C_6844	gemSpec_Kon	TIP1-A_4621	**XML-Teilbaumverschlüsselung entfernen** Um Risiken für eine erfolgreiche Zertifizierung des Konnektors zu reduzieren wird die Funktionalität zur Ver- und Entschlüsselung von XML-Teilbäumen entfernen	TIP1-A_4621: Beschreibung: ...~~Bei XML-Dokumenten werden ein oder mehrere XML-Elemente des Dokumentes verschlüsselt.~~ Für alle ~~übrigen~~ Dokumenttypen wird immer das gesamte Dokument verschlüsselt. EncryptionType: ... XML-Documente werden nach Type=http://www.w3.org/2001/04/xmlenc#Element verschlüsselt. Element: Der Parameter wird nicht ausgewertet TIP1-A_4622: Beschreibung: ...Die Operation entschlüsselt alle hyprid verschlüsselten Dokumente, die Mit der Operation EncryptDocument erzeugt wurden....	gemSpec_Kon_V5.4.0
C_6890	gemSpec_Kon	Kap. 4.1.8 Signaturdienst	**Profilierung der nonQES-Signatur** In der Spezifikation soll eine Profilierung der nonQES-Schnittstelle des Konnektors vorgenommen werden. Dabei handelt es sich im Einzelnen um: - Verzicht auf PDF-Gegensignatur - PDF-Signatur wird als Incremental Update des PDF-Dokuments festgelegt - Verzicht auf eingebettete OCSP-Responses bei PDF-Signaturen - Reduzierung auf eine Gegensignatur-Variante bei XAdES/CAdES: dokument-exkludierend - Streichung von Teilbaum XML Signaturen (auch nonQES)	siehe C_6820_C_6890_Anlage	gemSpec_Kon_V5.4.0 gemKPT_Arch_TIP_V2.5.0 gemProdT_Kon_PTV3

ID	Dokument	Quelle Dokument und/oder Kapitel	Beschreibung der Änderung	**Anpassungen** an Afos; TUCs, Tabellen, Korrekturen	von Änderung betroffene Dokumente
C_6892	gemSpec_InfoNFDM gemSpec_Info_AMTS	gemSpec_InfoNFDM Kap. 3.3 Ebene 1, gemSpec_InfoNFDM Kap. 5.4 Ebene 1, gemSpec_Info_AMTS Kap. 2.3	**Einfügen der OIDs für die Datensätze NFD, DPE und AMTS** Für jeden Datensatz (NFD, DPE und AMTS) gibt es jeweils eine eindeutige Identifikationsnummer die von der DIMDI vergeben wird. Dadurch lässt sich ein Datensatz als ein eindeutig von der gematik Spezifiziert NFD/DPE/AMTS-Datensatz identifizieren. Jede dieser Identifikationsnummer wird durch ein Attribut in dem jeweiligen Datensatz repräsentiert und erhält eine fest definierte OID. Bislang wurde nur ein Platzhalter für die OID als Wert eingetragen und soll nun durch die richtige OID ersetzt werden. Da eine Anpassung an den Schemata erfolgt wird die Versionsnummer der Schemadateien korrigiert.	Für NFDM im Attribut ID_NFD: Alt: attribute name ID_NFD type OID use required Beschreibung Identifier in der gematik-Objektnomenklatur Kommentar Angleichung an eMP/AMTS Neu: attribute name ID_NFD type string fixed 1.2.276.0.76.7.8 use required Beschreibung Identifier in der gematik-Objektnomenklatur Kommentar Angleichung an eMP/AMTS Abbildung Abb_INFO_012 element Notfalldaten wird entsprechend angepasst. Das Attribut ID_NFD wird in NFD_Document_v1_3.xsd angepasst. Für NFDM im Attribut ID_DPE: Alt: attribute name ID_DPE type OID use required Beschreibung Identifier in der gematik-Objektnomenklatur Kommentar Angleichung an eMP/AMTS Neu: attribute name ID_DPE type string fixed 1.2.276.0.76.7.9 use required	gemSpec_InfoNFDM_V1.4.0 gemSpec_Info_AMTS_V1.3.0 NFD_Document_v1_3.xsd DPE_Document.xsd AMTS_Document_v1_4.xsd
C_6907	gemSpec_Kon_SigProxy		**Darstellung von Formaten bekannter Signaturrichtlinien** Um sicherzustellen, dass der Arzt die Möglichkeit hat, eine geprüfte Anzeige der Qualifiziert zu Signierenden XML-Daten zu erhalten muss der Signaturproxy eine Vollständige Anzeige für Notfalldaten implementieren, ohne dass dafür XSLT/CSS-Dateien übergeben werden müssen.	Alt: TIP1-A_5674 Aufrufparameter : sp:GenerateUnderSignaturePolicy : Der Parameter wird im Signaturproxy nicht ausgewertet. Neu: TIP1-A_5674 Aufrufparameter : sp:GenerateUnderSignaturePolicy : Über den Parameter wird für alle im Konnektor eingebauten Signaturrichtlinien eine Vollständige Anzeige und PDF-Aufbereitung ausgelöst. A_18225 Anzeige von Formaten aus Signaturrichtlinien Der Signaturproxy muss zu jeder durch den Konnektor umgesetzten Signaturrichtlinie eine vollständige Anzeige und PDF-Erzeugung umsetzen und bei der Signaturerstellung und Prüfung anwenden.	gemSpec_Kon_SigProxy_V1.4.0

Änderungen sind gelb markiert. Änderungen aus C_6820 sind grau markiert.

Änderungen in [gemSpec_Kon]

4.1.8 Signaturdienst

4.1.8.1 Funktionsmerkmalweite Aspekte

4.1.8.1.1 Dokumentensignatur

Der Signaturdienst umfasst die Funktionalität der nicht-qualifizierten elektronischen Signatur (nonQES) mit der SM-B, sowie die qualifizierte elektronische Signatur (QES) mit dem HBA und den HBA-Vorläuferkarten HBA-qSig und ZOD_2.0 (=HBAx).
In der Abbildung fachlicher Abläufe kann es nötig sein, ein Dokument mehrfach parallel zu signieren, oder existierende Signaturen gegenzusignieren. Der Konnektor unterstützt **parallele Signaturen** (QES und nonQES). Ebenso unterstützt er Gegensignaturen (QES und nonQES), die jeweils alle bestehenden Signaturen gegensignieren. Die angebotene Möglichkeit des Gegensignierens bezieht sich dabei auf das Signieren aller vorhandenen parallelen Signaturen, während ein Gegensignieren von Gegensignaturen nicht angeboten wird. ~~Zwei Arten der Gegensignatur werden unterstützt, eine **dokumentinkludierende Gegensignatur**, bei der das Dokument und alle Signaturen gegensigniert werden, sowie eine~~ Der Konnektor unterstützt ausschließlich eine **dokumentexkludierende Gegensignatur** ~~**unterstützt**~~, bei der alle Signaturen gegensigniert werden, aber nicht der fachliche Inhalt des Dokumentes selbst.

TIP1-A_5447 **-** Einsatzbereich der Signaturvarianten
Der Signaturdienst MUSS für die Erstellung und Prüfung von nicht-qualifizierten elektronischen Signaturen (nonQES) und qualifizierten elektronischen Signaturen (QES) die Vorgaben zum Einsatzbereich gemäß Tabelle TAB_KON_778 umsetzen.
[<=]

Tabelle 184: TAB_KON_778 – Einsatzbereich der Signaturvarianten für XAdES, CAdES und PAdES

Signaturvarianten				**Einsatzbereich**		
Signatur verfahren	Signatur-variante	WAS wird signiert?	WO wird die Signatur abgelegt?	nonQES	QES Außen-schnittstelle	QES Fachmodul-schnittstelle
XAdES	detached	beliebiges (Binär)-Dokument	außerhalb des Dokuments in der SignResponse	Nein	Nein	Nein
XAdES	detached	gesamtes Input XML-Dokument (= Root-Element mit Subelementen)	außerhalb des Dokuments in der SignResponse	Nein	Nein	Nein
XAdES	detached	ausgewähltes nicht Root-Element mit Subelementen im Input XML-	außerhalb des Dokuments in der SignResponse	Nein	Nein	Nein

		Dokument				
XAdES	detached	ausgewähltes nicht Root-Element mit Subelementen im Input XML-Dokument	Innerhalb des Dokuments, aber außerhalb des signierten Subbaums	NeinJa	Bedingt	Ja Bedingt
XAdES	enveloped	gesamtes Input XML-Dokument (= Root-Element mit Subelementen)	Als direktes Child des Root-Elements	Ja	Ja Bedingt	Ja Bedingt
XAdES	enveloped	ausgewähltes nicht Root-Element mit Subelementen im Input XML-Dokument	Als direktes Child des ausgewählten Elements	NeinJa	Nein	Ja Bedingt
XAdES	enveloping	gesamtes Input XML-Dokument (= Root-Element mit Subelementen)	Im Dokument, das Root-Element umschließend	Ja	Ja Bedingt	Ja Bedingt
XAdES	enveloping	ausgewähltes nicht Root-Element mit Subelementen im Input XML-Dokument	Im Dokument, das ausgewählte Element umschließend	Nein	Nein	Nein
CAdES	detached	gesamtes Binärdokument	außerhalb des Dokuments in der SignResponse	Ja	Ja	Ja
CAdES	enveloping	gesamtes Binär-Dokument	innerhalb des CMS-Dokuments	Ja	Ja	Ja
PAdES	-	gesamtes PDF-Dokument	Im PDF-Dokument	Ja	Ja	Ja

Legende:
Ja: Die Signaturvariante ist für den Einsatzbereich erlaubt.
Nein: Die Signaturvariante ist für den Einsatzbereich nicht erlaubt.
Bedingt: Die Signaturvariante ist für den Einsatzbereich nicht erlaubt, es sei denn es wird durch eine im Konnektor integrierte Signaturrichtlinie explizit gefordert.
Die Spalten mit gelber Kopfzeile definieren die Signaturvarianten, die mit grauer, den Einsatzbereich. Beim Einsatzbereich wird zwischen nonQES und QES unterschieden und im Fall QES nach der Bereitstellung an der Außenschnittstelle oder intern für Fachmodule.
Die benötigten Signaturvarianten werden für XAdES über die Aufrufparameter IncludeObject und SignaturePlacement gemäß [OASIS-DSS] gesteuert.
Für CAdES erfolgt die Steuerung welche Signaturvariante gewählt wird, über den Aufrufparameter IncludeEContent.

4.1.8.3.1 TUC_KON_155 „Dokumente zur Signatur vorbereiten”

TIP1-A_4646 - TUC_KON_155 „Dokumente zur Signatur vorbereiten“
Der Konnektor MUSS den technischen Use Case TUC_KON_155 „Dokumente zur Signatur vorbereiten” umsetzen.

Tabelle 187: TAB_KON_748 - TUC_KON_155 „Dokumente zur Signatur vorbereiten“

Element	Beschreibung
Name	TUC_KON_155 ”Dokumente zur Signatur vorbereiten”
Beschreibung	Die zu signierenden Dokumente werden entsprechend den Erfordernissen der Signaturverfahren für die QES oder nonQES vorbereitet.
Anwendungsumfeld	Erstellung von qualifizierten elektronischen Signaturen (QES) und nicht-qualifizierten elektronischen Signaturen (nonQES)
Auslöser	Aufruf durch TUC_KON_150 „Dokumente QES signieren“ oder TUC_KON_160 „Dokumente nonQES signieren“
Vorbedingungen	keine
Eingangsdaten	- signatureMode (Signaturart: QES | nonQES) - documentsToBeSigned (Zu signierendes Dokument bzw. zu signierende Dokumente) und pro Dokument: - documentFormat (Formatangabe für das zu signierende Dokument) - optionalInputs (weitere optionale Eingabeparameter zur Steuerung der Details bei der zu erstellenden Signatur (siehe Operation SignDocument, Parameter dss:OptionalInputs), darin u.a. -signatureType (URI für den Signaturtyp XML-, CMS-, S/MIME-o PDF-Signatur) - certificate (Signaturzertifikat) - ocspResponses – *optional* (OCSP-Response des EE-Zertifikats, das bei der Signaturerstellung in die Signatur eingebettet wird.)
Komponenten	Konnektor
Ausgangsdaten	• preProcessedDocuments (Aufbereitetes zu signierendes Dokument bzw. aufbereitete zu signierende Dokumente)
Standardablauf	signatureType = XMLDSig (XAdES) Entsprechend den Regeln für die QES und die nonQES werden zunächst weitere Signatureigenschaften zum jeweiligen Dokument in Form von `QualifyingProperties` (siehe [XAdES]) hinzugefügt. Die Systemzeit des Anwendungskonnektors muss in das XML-Element `SigningTime` (siehe [XAdES]) eingetragen werden. Die Signatur wird anschließend entsprechend [XMLDSig] vorbereitet. D. h., es wird je Dokument nach Erzeugung der Reference Elemente das SignedInfo Element aufgebaut. Dessen Inhalt ergibt dann nach erfolgter XML-Kanonisierung und Hashing die DTBS (Data To Be Signed), die später zur Karte gesendet werden. certificate wird im Element `ds:KeyInfo/ds:X509Data` gespeichert. Im Fall signatureMode = QES können neben den reinen Nutzdaten auch alle weiteren Elemente in die Signatur einbezogen werden, die für die Rekonstruktion der ursprünglich dargestellten Daten in der sicheren

	Anzeige erforderlich sind. Für XML-Dokumente sind das, falls vorhanden, das/die XML-Schema(ta). Für diese werden Referenzen (Hash + URI) in die Signatur eingebettet. Die URI ist im Fall übergebener XML-Schemata der übergebene *signatureType* - Parameter. Die URI ist im Fall der im Konnektor im Rahmen einer Signaturrichtlinie hinterlegten XML-Schemata/XSL-Stylesheets die URI der Signaturrichtlinie, ergänzt um den Dateinamen mit Pfad, wie in der Signaturrichtlinie festgelegt. (Beispiel: URI für Schemadatei ~~CDA.xsd der Signaturrichtlinie SR_DF_SV_VHITG_ARZTBRIEF lautet: urn:gematik:fa:sak:cda:r1:v1:fa:sak:cda:schemas:CDA.xsd~~ NFD_Document.xsd der Signaturrichtlinie SR_DF_NFDM_NOTFALLDATEN lautet: urn:gematik:fa:sak:nfdm:r1:v1) Das Einbetten der Referenzen erfolgt über das XML-Element ds:object/ds:manifest (XMLDSig) mit eingebetteten XML-Elementen ds:Reference, die eine URI (RefURI) als Identifier für die jeweilige Datei und einen Hash über die jeweilige Resource enthalten. Der ShortTextClientsystem muss in die Signatur in das `DataObjectFormat/Description`-Element gemäß [XAdES] (Abschnitt 7.2.5) eingebettet werden. Falls durch den Aufrufparameter `SIG:IncludeRevocationInfo` angefordert, wird die für die Offline-Prüfung notwendige OCSP-Antwort im Sinne vom ES-X-L vom Konnektor in die Signatur eingebettet: Die base-64 kodierte OCSP-Response wird im Feld `QualifyingProperties/UnsignedProperties /UnsignedSignatureProperties/RevocationValues /OCSPValues/EncapsulatedOCSPValue` (selbst DER-kodiert) gespeichert. signatureType = CMS (CAdES) Etwaig einzubettende XML-Schemata werden zunächst wie für XAdES definiert in ein ds:manifest-Element eingebettet. Die so erzeugte Zeichenkette wird als genau ein ASN.1 Character String vom Typ UTF8String verpackt. Dieser wird als contentDescription in einen Content-Hints Attributwert vom Typ ContentHints verpackt, wobei der contentType=id-data gemäß [CAdES]. Der ShortTextClientsystem muss in die Signatur in das `content-hints.ContentDescription`-Attribut gemäß [CAdES] (Abschnitt 5.10.3) eingebettet werden. Ist die Einbettung von OCSP-Responses gefordert, wird die für die Offline-Prüfung notwendige OCSP-Antwort des EE-Zertifikats im Attribut SingedData.crls.other abgelegt. signatureType = PDF/A (PAdES) Der ShortTextClientsystem muss bei einer PDF-Signatur in das `Reason`-Feld eingebettet werden. OCSP-Responses werden bei PAdES nicht eingebettet. ~~OCSP-Response des EE-Zertifikats ist im Document Security Store gemäß [PAdES-4] einzubetten.~~ Es sind die Vorgaben zum Signaturprofil gemäß Tabelle TAB_KON_779 „Profilierung der Signaturformate“ zu erfüllen.

	Die aufbereiteten zu signierenden Dokumente werden an den Aufrufer zurückgegeben.
Varianten/ Alternativen	keine
Fehlerfälle	Das Verhalten des TUCs bei einem Fehlerfall ist in TAB_KON_586 Fehlercodes TUC_KON_155 „Dokumente zur Signatur vorbereiten“ „PDF/A (PAdES)“ Es ist nicht genügend Speicherplatz im PDF-Dokument verfügbar: 4205
Nichtfunktionale Anforderungen	keine
Zugehörige Diagramme	keine

Tabelle 188: TAB_KON_586 Fehlercodes TUC_KON_155 „Dokumente zur Signatur vorbereiten“

Fehlercode	ErrorType	Severity	Fehlertext
4205	Technical	Error	Es ist nicht genügend Speicherplatz im PDF-Dokument verfügbar.

[<=]

4.1.8.4.2 TUC_KON_161 „nonQES Dokumentsignatur prüfen ”

TIP1-A_4654 - TUC_KON_161 „nonQES Dokumentsignatur prüfen”
Der Konnektor MUSS den technischen Use Case TUC_KON_161 „nonQES Dokumentsignatur prüfen” umsetzen.

Tabelle 201: TAB_KON_121 - TUC_KON_161 „nonQES Dokumentsignatur prüfen“

Element	Beschreibung
Name	TUC_KON_161 „nonQES Dokumentsignatur prüfen”
Beschreibung	Es wird die nicht-qualifizierte elektronische Signatur (nonQES) eines Dokuments geprüft. Dabei werden die Signaturverfahren laut Tabelle 183: TAB_KON_582 – Signaturverfahren unterstützt. Sind mehrere Signaturen vorhanden, so werden alle geprüft. Auch Parallel- und Gegensignaturen MÜSSEN unterstützt werden.
Auslöser	Aufruf durch ein Clientsystem (Operation VerifyDocument) oder ein Fachmodul
Vorbedingungen	keine
Eingangsdaten	• signedDocument (Signiertes Document vom Typ `nonQES_DocFormate`) • signature – *optional/falls detached Signatur* (Signatur. Es werden Parallel- und Gegensignaturen unterstützt.) • optionalInputs (optionale Eingabeparameter, siehe Operation VerifyDocument, Parameter SIG:OptionalInputs) • certificate – *optional/verpflichtend, wenn das Zertifikat nicht im signierten Dokument enthalten ist* (X.509-Zertifikat, gegen das die Signatur geprüft werden soll)

	ocspGracePeriod (OCSP-Grace Period: maximal zulässiger Zeitraum, den die letzte OCSP-Antwort aus dem Cache bezüglich des Referenzzeitpunkts zurückliegen darf) • xmlSchemas – *optional/nur für XML-Dokumente* (XMLSchema und ggf. weitere vom Hauptschema benutzte Schemata) • includeRevocationInfo: – *optional; Default = false* (Dieser optionale Parameter steuert die Einbettung von OCSP Antworten in die Signatur)
Komponenten	Konnektor
Ausgangsdaten	• verificationResult [VerificationResult] (Ergebnis der Signaturprüfung) • optionalOutput – *optional* (weitere Ausgabedaten gemäß SIG:OptionalOutput)
Standardablauf	1. **„DocumentValidation“**: Falls die Signatur im Dokument eingebettet ist, wird das signierte Dokument validiert durch Aufruf TUC_KON_080 „Dokument validieren“ { CheckDisplayability=false; … } Treten dabei Fehler bei Validierung der Typkonformität auf, wird die Prüfung mit einem Fehler abgebrochen. 2. **„CoreValidation“**: Es erfolgt die mathematische Prüfung der Signatur bestehend aus der Prüfung der Hash-Kette bis zum signierten Hashwert und der Prüfung der kryptographischen Signatur unter Verwendung des öffentlichen Schlüssels, des Signaturwertes und des signierten Hashwertes. XML-Signatur: Die Core Validation erfolgt entsprechend [XMLDSig] Kapitel 3.2 Core Validation. CMS-Signatur: Die Core Validation erfolgt entsprechend Cryptographic Message Syntax (CMS) Kapitel 5.6 Signature Verification Process [RFC5652]. PDF-Signatur: Die Core Validation erfolgt entsprechend [PAdES-3] Kapitel 4.6 Signature Validation aus PAdES-BES Part 3. Auch wenn die Validierung fehlschlägt, werden die folgenden Prüfschritte durchgeführt, so dass ein vollständiges Prüfprotokoll erstellt werden kann. 3. „CheckSignatureCertificate“: **Teil 1: Signaturzertifikat ermitteln** XML-Signatur: Das Signaturzertifikat ist im XMLDSig Element `ds:KeyInfo/ds:X509Data` gespeichert [XMLDSig] oder wird als Eingangsparameter übergeben. CMS-Signatur: Das Signaturzertifikat für CAdES ist im Feld `certificates` im `SignedData` Container gespeichert [CAdES] oder wird als Eingangsparameter übergeben. PDF-Signatur: Das PDF Signaturzertifikat für PAdES ist im Feld SignedData.certificates entsprechend Kapitel 6.1.1 „Placements of the signing certificate“

	[PAdES Baseline Profile] gespeichert oder wird als Eingangsparameter übergeben.
	Teil 2: Signaturzeitpunkt bestimmen
	Der Signaturzeitpunkt Ermittelter_Signaturzeitpunkt_Eingebettet wird wie folgt selektiert:
	XML-Signatur:
	Das XML element `SigningTime` spezifiziert den Signaturzeitpunkt entsprechend Kapitel 7.2.1 XAdES [XAdES].
	CMS-Signatur:
	Das Attribut `SigningTime` spezifiziert den Signaturzeitpunkt entsprechend Kapitel 11.3 CMS [CMS].
	PDF-Signatur:
	Der Signaturzeitpunkt kann dem M Eintrag des Signature Dictionary entnommen werden [PAdES Baseline Profile] Kapitel 6.2.1 Signing time.
	Der Signaturzeitpunkt `Benutzerdefinierter_Zeitpunkt` liegt gegebenenfalls als Aufrufparameter vor. Der Signaturzeitpunkt `Ermittelter_Signaturzeitpunkt_System` wird ermittelt.
	Teil 3: Signaturzertifikatsprüfung:
	Bei der folgenden Signaturzertifikatsprüfung sind die Signaturzeitpunkte gemäß [TIP1-A_5545] zu berücksichtigen. Die Signaturzertifikatsprüfung erfolgt durch Aufruf von TUC_KON_037 „Zertifikat prüfen“, und zwar:
	Wenn es sich um das X.509-Zertifikat einer eGK handelt (PolicyList = oid_egk_aut bzw. oid_egk_autn), dann:
	TUC_KON_037 „Zertifikat prüfen“ {
	certificate;
	qualifiedCheck = not_required;
	baseTime = Signaturzeitpunkt;
	offlineAllowNoCheck = true;
	policyList = [oid_egk_aut | oid_egk_autn];
	intendedKeyUsage = digitalSignature&keyEncipherment;
	intendedExtendedKeyUsage = id-kp-clientAuth;
	ocspResponses = OCSP-Response;
	gracePeriod = ocspGracePeriod;
	validationMode = OCSP;
	getOCSPResponses = includeRevocationInfo }
	Wenn es ein X.509-Zertifikat der SM-B ist (PolicyList = oid_smc_b_osig), dann:
	TUC_KON_037 „Zertifikat prüfen“ {
	certificate;
	qualifiedCheck = not_required;
	baseTime = Signaturzeitpunkt;
	offlineAllowNoCheck = true;
	policyList = oid_smc_b_osig;
	intendedKeyUsage = nonRepudiation;
	ocspResponses = OCSP-Response;
	gracePeriod = ocspGracePeriod;
	validationMode = OCSP ;
	getOCSPResponses = includeRevocationInfo }
	Sind OCSP-Responses in der Signatur eingebettet, ist die jüngste OCSP-Response, die für die Zertifikatsprüfung notwendig ist, beim Aufruf von TUC_KON_037 zu übergeben. Sofern der Aufruf von TUC_KON_037 ocspResponsesRenewed zurückgibt, wird die Liste der OCSP-Responses in die Signatur eingebettet.

	Auch wenn die Zertifikatsprüfung fehlschlägt, werden die folgenden Prüfungen durchgeführt. 4. “CheckPolicyConstraints“ In diesem Schritt wird das signierte Dokument entsprechend der Profilierung der Signaturformate (siehe Anhang B.2) geprüft. Es sind die Vorgaben für die Prüfung von Signaturen aus den Standards für AdES [XAdES], [XAdES Baseline], [CAdES], [CAdES Baseline], [PAdES-3] und [PAdES Baseline] umzusetzen. Dabei sind die Vorgaben aus Tabelle TAB_KON_779 „Profilierung der Signaturformate“ und Tabelle TAP_KON_778 „Einsatzbereich der Signaturvarianten“ zu erfüllen. Auch wenn nicht alle Anforderungen an das Format des signierten Dokuments erfüllt werden, wird die Prüfung mit den folgenden Schritten fortgesetzt, um ein vollständiges Prüfungsprotokoll zu erhalten. 5. Das Prüfergebnis (verificationResult, optionalOutput wird an den Aufrufer zurückgegeben (siehe TAB_KON_754 Übersicht Status für Prüfung einer Dokumentensignatur).
Varianten/ Alternativen	Im Fall, dass die Online-Prüfung des Sperrzustands des Signaturzertifikats nicht möglich ist und eine möglicherweise gecachte OCSP-Response nicht vorhanden ist oder nicht mehr verwendet werden darf, wird das Prüfergebnis mit der entsprechenden Warnung zurückgegeben. Im Fall einer PKCS#1-Signatur ist das verwendete Signaturverfahren, RSASSA-PSS bzw. RSASSA-PKCS1-v1_5, aus der Signatur zu bestimmen.
Fehlerfälle	Das Verhalten des TUCs bei einem Fehlerfall ist in TAB_KON_124 Fehlercodes TUC_KON_161 „nonQES Dokumentensignatur prüfen“ beschrieben. (->1) keine Signatur in signedDocument und signature vorhanden: 4253 (**à** 2 **„CoreValidation“**) Interner Fehler: 4001, Signatur des Dokument ungültig: 4115. Signatur umfasst nicht das gesamte Dokument: 4262 (**à** 3 „**CheckSignatureCertificate“**) Interner Fehler: 4001, Signaturzertifikat ermitteln fehlgeschlagen: 4206. (**à** 4 **„CheckPolicyConstraints“**) Interner Fehler: 4001, Dokument nicht konform zu Regeln für nonQES: 4112.
Nichtfunktionale Anforderungen	keine
Zugehörige Diagramme	keine

Tabelle 202: TAB_KON_124 Fehlercodes TUC_KON_161 „nonQES Dokumentensignatur prüfen“

Fehlercode	ErrorType	Severity	Fehlertext
Neben den Fehlercodes der aufgerufenen technischen Use Cases können folgende weiteren Fehlercodes auftreten.			

4001	Technical	Error	Interner Fehler
4206	Technical	Error	Signaturzertifikat ermitteln ist fehlgeschlagen
4112	Technical	Error	Dokument nicht konform zu Regeln für nonQES
4115	Security	Error	Signatur des Dokuments ungültig. Der SignatureValue des Dokuments ist falsch oder für mindestens eine Reference ist der DigestValue falsch.
4253	Technical	Error	Keine Signatur im Aufruf
4262	Technical	Error	Signatur umfasst nicht das gesamte Dokument

Das Gesamtergebnis (VerificationResult) für die Prüfung einer Dokumentensignatur fasst die Ergebnisse aller Prüfungsschritte in einem einzelnen Statuswert zusammen.

Tabelle 203: TAB_KON_754 Übersicht Status für Prüfung einer Dokumentensignatur

VerificationResult für gesamtes Dokument (VerificationResult/HighLevelResult)	
Wert	**Bedeutung**
VALID	Wenn VerificationResult für alle Signaturen zum Dokument VALID
INVALID	Wenn VerificationResult für eine Signatur zum Dokument INVALID
INCONCLUSIVE	in allen anderen Fällen
VerificationResult pro Signatur (VerificationReport/IndividualReport/Result)	
Wert	**Bedeutung** **mögliche Ausprägungen im VerificationReport**
VALID	Die Signatur wurde gemäß den Regeln für die nonQES geprüft und für gültig befunden.
	ResultMajor = urn:oasis:names:tc:dss:1.0:resultmajor:Success ResultMinor = urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:OnAllDocuments
	ResultMajor = urn:oasis:names:tc:dss:1.0:resultmajor:Success ResultMinor = urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:HasManifestResults
INVALID	Die Signatur ist ungültig oder aufgrund eines Fehlers konnte die Signaturprüfung nicht durchgeführt werden.
	ResultMajor = urn:oasis:names:tc:dss:1.0:resultmajor:Success ResultMinor = urn:oasis:names:tc:dss:1.0:resultminor:invalid:IncorrectSignature
	ResultMajor = urn:oasis:names:tc:dss:1.0:resultmajor:Success ResultMinor = urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:InvalidSignatureTimestamp
	ResultMajor = urn:oasis:names:tc:dss:1.0:resultmajor:RequesterError

	ResultMajor = urn:oasis:names:tc:dss:1.0:resultmajor:ResponderError
	ResultMajor = urn:oasis:names:tc:dss:1.0:resultmajor:InsufficientInformation ResultMinor = urn:oasis:names:tc:dss:1.0:resultminor:invalid:IncorrectSignature
	ResultMajor = urn:oasis:names:tc:dss:1.0:resultmajor:InsufficientInformation ResultMinor = urn:oasis:names:tc:dss:1.0:resultminor:CertificateChainNotComplete
INCONCLUSIVE	Die Signatur wurde gemäß den Regeln für die nonQES geprüft. Allerdings konnten eine oder mehrere Prüfungen nicht vollständig durchgeführt werden. Einzelheiten finden sich in Result-Detail. Die Prüfungen, die durchgeführt werden konnten, waren erfolgreich.
	ResultMajor = urn:oasis:names:tc:dss:1.0:resultmajor:InsufficientInformation ResultMinor = urn:oasis:names:tc:dss:1.0:resultminor:OcspNotAvailiable Hinweis: Das Erreichen dieses Zustandes hängt davon ab, ob eine OCSP-Abfrage nicht durchgeführt werden konnte, unabhängig davon, ob die Ursache dafür die Offlineschaltung des Konnektors (MGM_LU_ONLINE = Disabled) oder die Nichterreichbarkeit des OCSP-Responders im Online-Betrieb (MGM_LU_ONLINE = Enabled) ist.

4.1.8.4.6 TUC_KON_151 „QES Dokumentensignatur prüfen“

TIP1-A_4672 - TUC_KON_151 „QES-Dokumentensignatur prüfen“
Der Konnektor MUSS den technischen Use Case TUC_KON_151 „QES-Dokumentensignatur prüfen” umsetzen.

Tabelle 209: TAB_KON_591 - TUC_KON_151 „QES-Dokumentensignatur prüfen“

Element	Beschreibung
Name	TUC_KON_151 „QES-Dokumentensignatur prüfen”
Beschreibung	Es wird die QES eines Dokuments geprüft. Dabei werden die Signaturverfahren laut Tabelle 183: TAB_KON_582 – Signaturverfahren unterstützt. Sind mehrere Signaturen vorhanden, so werden alle geprüft. Auch Parallel- und Gegensignaturen MÜSSEN unterstützt werden.
Eingangsanforderung	keine
Auslöser	Aufruf durch ein Clientsystem (Operation VerifyDocument) oder durch ein Fachmodul im Konnektor
Vorbedingungen	keine
Eingangsdaten	• signedDocument – *optional* (QES-signiertes Dokument vom Typ `QES_DocFormate` -> siehe Definition in Operation `VerifyDocument` mit `SIG:Document`) • signatureObject – *optional* (-> siehe Definition in Operation VerifyDocument mit dss:SignatureObject. Es werden Parallel- und Gegensignaturen unterstützt.) • optionalInputParams (optionale Eingabeparameter, siehe Operation VerifyDocument, Parameter SIG:OptionalInputs) • certificates – *optional/falls diese nicht im signierten Dokument enthalten sind, sondern nur referenziert werden*

	(X.509-Zertifikate). • xmlSchemas – *optional/nur für XML-Dokumente* (XMLSchema und ggf. weitere vom Hauptschema benutzte Schemata) • includeRevocationInfo [Boolean]: – *optional; Default: false* (Dieser optionale Parameter steuert die Einbettung von OCSP Antworten in die Signatur)
Komponenten	Konnektor
Ausgangsdaten	• verificationResult [VerificationResult] (Ergebnis der Signaturprüfung) • optionalOutput – *optional* (weitere Ausgabedaten gemäß SIG:OptionalOutput)
Standardablauf	1. **„DocumentValidation“**: Das signierte Dokument wird validiert mit Aufruf TUC_KON_080 „Dokument validieren“{ … }. Treten Fehler bei der Validierung der Typkonformität auf, wenn die Signatur im Dokument eingebettet ist, wird die Prüfung mit einem Fehler abgebrochen. Treten bei der Typkonformität, wenn die Signatur nicht im Dokument eingebettet ist, Fehler auf, so bricht der TUC nicht ab, sondern führt die folgenden Schritte soweit sinnvoll möglich durch. (Die Entscheidung über das sinnvoll Durchführbare liegt beim Hersteller des Konnektors.) 2. **„CoreValidation“**: Es erfolgt die mathematische Prüfung der Signatur, bestehend aus der Prüfung der Hash-Kette bis zum signierten Hashwert und der Prüfung der Signatur unter Verwendung des öffentlichen Schlüssels, des Signaturwertes und des signierten Hashwertes. XML-Signatur: Die Core Validation erfolgt entsprechend [XMLDSig] Kapitel 3.2 Core Validation. CMS-Signatur: Die Core Validation erfolgt entsprechend Cryptographic Message Syntax (CMS) Kapitel 5.6 Signature Verification Process [RFC5652]. PDF-Signatur: Die Core Validation erfolgt entsprechend [PAdES-3] Kapitel 4.6 Signature Validation aus PAdES-BES Part 3. Auch wenn die Validierung fehlschlägt, werden die folgenden Prüfschritte durchgeführt, so dass ein vollständiges Prüfprotokoll erstellt werden kann. 3. „CheckSignatureCertificate“: **Teil 1: Signaturzertifikat ermitteln** XML-Signatur: Das Signaturzertifikat ist im XMLDSig Element `ds:KeyInfo/ds:X509Data` gespeichert [XMLDSig] oder wird als Eingangsparameter übergeben. CMS-Signatur: Das Signaturzertifikat für CAdES ist im Feld `certificates` im `SignedData` Container gespeichert [CAdES] oder wird als Eingangsparameter übergeben. PDF-Signatur: Das PDF Signaturzertifikat für PAdES ist im Feld SignedData.certificates entsprechend Kapitel 6.1.1 „Placements of the signing certificate“ [PAdES Baseline Profile] gespeichert oder wird als Eingangsparameter übergeben.

	Teil 2: Signaturzeitpunkt bestimmen
	Der Signaturzeitpunkt `Ermittelter_Signaturzeitpunkt_Eingebettet` wird wie folgt selektiert:
	XML-Signatur:
	Das XML element `SigningTime` spezifiziert den Signaturzeitpunkt entsprechend Kapitel 7.2.1 XAdES [XAdES].
	CMS-Signatur:
	Das Attribut `SigningTime` spezifiziert den Signaturzeitpunkt entsprechend Kapitel 11.3 CMS [CMS].
	PDF-Signatur:
	Der Signaturzeitpunkt kann dem M Eintrag des Signature Dictionary entnommen werden [PAdES Baseline Profile] Kapitel 6.2.1 Signing time.
	Der Signaturzeitpunkt `Ermittelter_Signaturzeitpunkt_Qualifiziert` wird wie folgt selektiert:
	XML-Signatur:
	Das XML element `SignatureTimeStamp` spezifiziert den Signaturzeitpunkt entsprechend Kapitel 4.4.3.1 XAdES [XAdES].
	CMS-Signatur und PDF-Signatur:
	Das Attribut `signature-time-stamp` spezifiziert den Signaturzeitpunkt entsprechend Kapitel 6.1 CAdES [CAdES].
	Der Signaturzeitpunkt `Benutzerdefinierter_Zeitpunkt` liegt gegebenenfalls als Aufrufparameter vor. Der Signaturzeitpunkt `Ermittelter_Signaturzeitpunkt_System` wird ermittelt.
	
	Teil 3: Signaturzertifikatsprüfung:
	Bei der folgenden Signaturzertifikatsprüfung sind die Signaturzeitpunkte gemäß [TIP1-A_5540] zu berücksichtigen.
	Die Signaturzertifikatsprüfung erfolgt durch Aufruf von TUC_KON_037 „Zertifikat prüfen“ {
	certificate = C.HP.QES;
	qualifiedCheck = required;
	baseTime = Signaturzeitpunkt;
	offlineAllowNoCheck = true;
	validationMode = OCSP;
	ocspResponses = OCSP-Response;
	getOCSPResponses = includeRevocationInfo
	}.
	Sind OCSP-Responses in der Signatur eingebettet, ist die jüngsten OCSP-Response des EE-Zertifikats, die für die Zertifikatsprüfung notwendig ist, beim Aufruf von TUC_KON_037 zu übergeben.
	Sofern der Aufruf von TUC_KON_037 ocspResponses zurückgibt, wird die OCSP-Response des EE-Zertifikats in die Signatur eingebettet.
	Auch wenn die Zertifikatsprüfung fehlschlägt, werden die folgenden Prüfungen durchgeführt.
	4. **„CheckPolicyConstraints“**:
	In diesem Schritt wird das signierte Dokument entsprechend der Profilierung der Signaturformate (siehe Anhang B.2) geprüft. Es sind die Vorgaben für die Prüfung von Signaturen aus den Standards für AdES [XAdES], [XAdES Baseline], [CAdES], [CAdES Baseline], [PAdES-3] und [PAdES Baseline] umzusetzen. Dabei sind die Vorgaben aus Tabelle TAB_KON_779 „Profilierung der Signaturformate“ und Tabelle TAP_KON_778 „Einsatzbereich der Signaturvarianten“ zu erfüllen.
	Auch wenn nicht alle Anforderungen an das Format des signierten

	Dokuments erfüllt werden, wird die Prüfung mit den folgenden Schritten fortgesetzt, um ein vollständiges Prüfungsprotokoll zu erhalten. 5. Das **Prüfergebnis** (VerificationResult, OptionalOutput) wird an den Aufrufer zurückgegeben (siehe TAB_KON_593 Übersicht Status für Prüfung einer Dokumentensignatur).
Varianten/Alternativen	Keine
Fehlerfälle	Das Verhalten des TUCs bei einem Fehlerfall ist in TAB_KON_592 Fehlercodes TUC_KON_151 „QES Dokumentensignatur prüfen“ beschrieben. (->1) keine Signatur in signedDocument und signatureObject vorhanden: 4253. (**à** 2 **„CoreValidation“**) Interner Fehler: 4001, Signatur des Dokuments ungültig: 4115, Signatur umfasst nicht das gesamte Dokument: 4262 (**à** 3 **„CheckSignatureCertificate“**) Interner Fehler: 4001, Signaturzertifikat ermitteln ist fehlgeschlagen: 4206. (**à** 4 **„CheckPolicyConstraints“**) Interner Fehler: 4001, Dokument nicht konform zu Regeln für QES: 4124, Dokument nicht konform zu Profilierung der Signaturformate: 4208.
Nichtfunktionale Anforderungen	Keine
Zugehörige Diagramme	keine

Tabelle 210: TAB_KON_592 Fehlercodes TUC_KON_151 „QES Dokumentensignatur prüfen“

Fehlercode	ErrorType	Severity	Fehlertext
Neben den Fehlercodes der aufgerufenen technischen Use Cases können folgende weitere Fehlercodes auftreten:			
4001	Technical	Error	interner Fehler
4115	Security	Error	Signatur des Dokuments ungültig. Prüfung der Hashwertkette bzw. Prüfung der kryptographischen Signatur fehlgeschlagen.
4124	Technical	Error	Dokument nicht konform zu Regeln für QES
4206	Technical	Error	Signaturzertifikat ermitteln ist fehlgeschlagen
4208	Technical	Error	Dokument nicht konform zu Profilierung der Signaturformate
4253	Technical	Error	Keine Signatur im Aufruf
4262	Technical	Error	Signatur umfasst nicht das gesamte Dokument

Das Gesamtergebnis (VerificationResult) für die Prüfung einer Dokumentensignatur fasst die Ergebnisse aller Prüfungsschritte in einem einzelnen Statuswert zusammen.

Tabelle 211: TAB_KON_593 Übersicht Status für Prüfung einer Dokumentensignatur

VerificationResult für gesamtes Dokument (VerificationResult/HighLevelResult)	
Wert	Bedeutung
VALID	Wenn VerificationResult für alle Signaturen zum Dokument VALID
INVALID	Wenn VerificationResult für eine Signatur zum Dokument INVALID
INCONCLUSIVE	in allen anderen Fällen
VerificationResult pro Signatur (VerificationReport/IndividualReport/Result)	
Wert	Bedeutung

	mögliche Ausprägungen im VerificationReport
VALID	Die Signatur wurde gemäß den Regeln für die QES geprüft und für gültig befunden.
	ResultMajor = urn:oasis:names:tc:dss:1.0:resultmajor:Success ResultMinor = urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:OnAllDocuments
	ResultMajor = urn:oasis:names:tc:dss:1.0:resultmajor:Success ResultMinor = urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:HasManifestResults
INVALID	Die Signatur ist ungültig oder aufgrund eines Fehlers konnte die Signaturprüfung nicht durchgeführt werden.
	ResultMajor = urn:oasis:names:tc:dss:1.0:resultmajor:Success ResultMinor = urn:oasis:names:tc:dss:1.0:resultminor:invalid:IncorrectSignature
	ResultMajor = urn:oasis:names:tc:dss:1.0:resultmajor:Success ResultMinor = urn:oasis:names:tc:dss:1.0:resultminor:valid:signature:InvalidSignatureTimestamp
	ResultMajor = urn:oasis:names:tc:dss:1.0:resultmajor:RequesterError
	ResultMajor = urn:oasis:names:tc:dss:1.0:resultmajor:ResponderError
	ResultMajor = urn:oasis:names:tc:dss:1.0:resultmajor:InsufficientInformation ResultMinor = urn:oasis:names:tc:dss:1.0:resultminor:invalid:IncorrectSignature
	ResultMajor = urn:oasis:names:tc:dss:1.0:resultmajor:InsufficientInformation ResultMinor = urn:oasis:names:tc:dss:1.0:resultminor:CertificateChainNotComplete
INCONCLUSIVE	Die Signatur wurde gemäß den Regeln für die QES geprüft. Allerdings konnten eine oder mehrere Prüfungen nicht vollständig durchgeführt werden. Einzelheiten finden sich in Result-Detail. Die Prüfungen, die durchgeführt werden konnten, waren erfolgreich.
	ResultMajor = urn:oasis:names:tc:dss:1.0:resultmajor:InsufficientInformation ResultMinor = urn:oasis:names:tc:dss:1.0:resultminor:OcspNotAvailiable Hinweis: Das Erreichen dieses Zustandes hängt davon ab, ob eine OCSP-Abfrage nicht durchgeführt werden konnte, unabhängig davon, ob die Ursache dafür die Offlineschaltung des Konnektors (MGM_LU_ONLINE = Disabled) oder die Nichterreichbarkeit des OCSP-Responders im Online-Betrieb (MGM_LU_ONLINE = Enabled) ist.

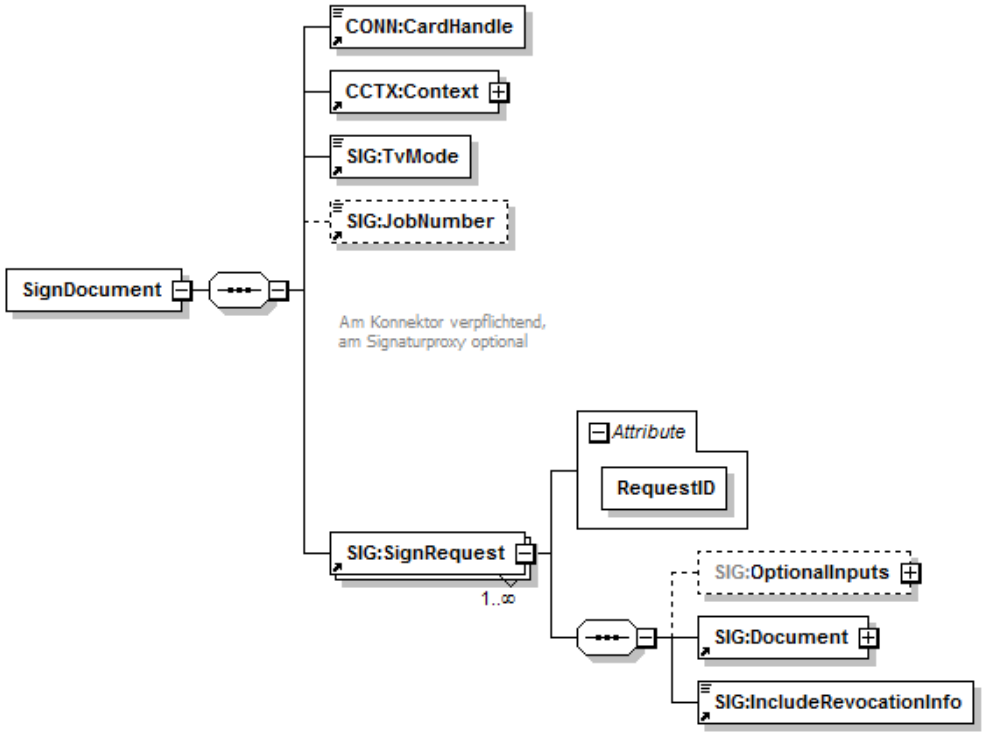
[<=]

4.1.8.5 Operationen an der Außenschnittstelle

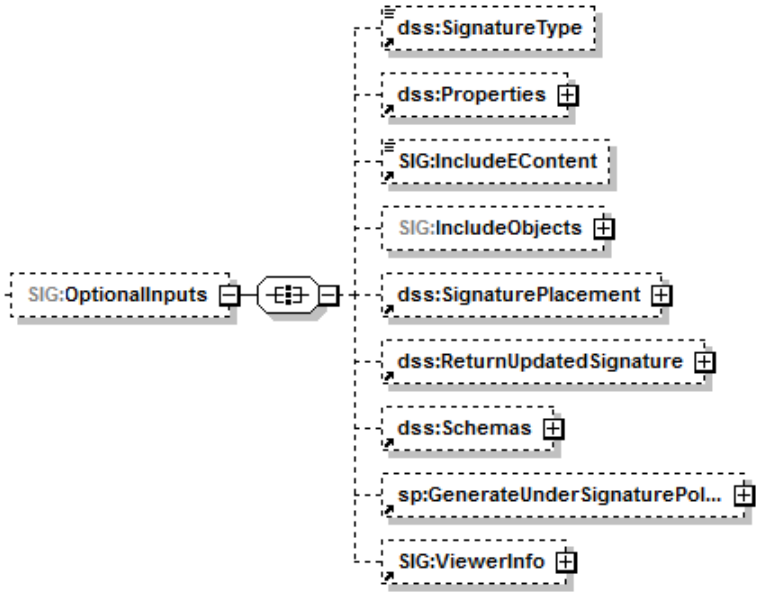
4.1.8.5.1 SignDocument (nonQES und QES)

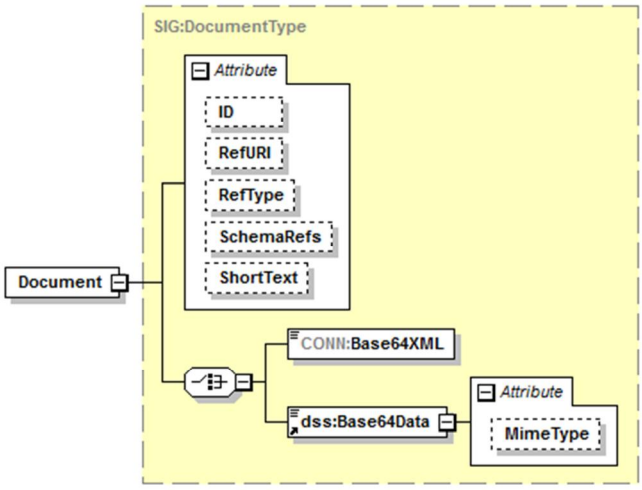
TIP1-A_5010 - Operation SignDocument (nonQES und QES)
Der Signaturdienst des Konnektors MUSS an der Clientschnittstelle eine an [OASIS-DSS] angelehnte Operation SignDocument anbieten.

Tabelle 213: TAB_KON_065 Operation SignDocument (nonQES und QES)

Name	`SignDocument`
Beschreibung	Diese Operation lehnt sich an [OASIS-DSS] an. Sie enthält voneinander unabhängige SignRequests. Jeder SignRequest erzeugt eine Signatur für ein Dokument. Für die qualifizierte elektronische Signatur (QES) werden die `QES_DocFormate` unterstützt. Für nicht-qualifizierte elektronische Signaturen (nonQES) werden die `nonQES_DocFormate` unterstützt. Zur Signaturerzeugung werden Schlüssel und Zertifikate einer Chipkarte benutzt. Unterstützte Karten sind für die QES der HBAx mit dem QES-Zertifikat. Für die nonQES wird für die Signaturtypen „XML-Signatur, CMS-Signatur, PDF-Signatur, S/MIME-Signatur“ die SM-B mit dem OSIG–Zertifikat unterstützt. Bei der Erstellung von XML-Signaturen MUSS Canonical XML 1.1 verwendet werden [CanonXML1.1]. Es soll der Common-PKI-Standard eingesetzt werden, siehe [Common-PKI]. In Summe für die Größe der Dokumente in allen SignRequests innerhalb einer SignDocument-Anfrage MUSS der Konnektor eine Gesamtgröße von <= 250 MB unterstützen.
Aufrufparameter	

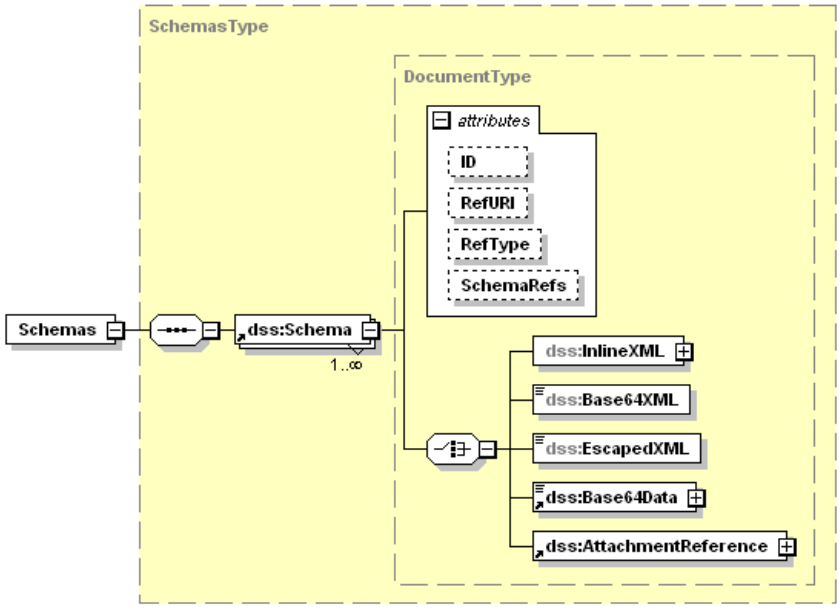
Name	Beschreibung

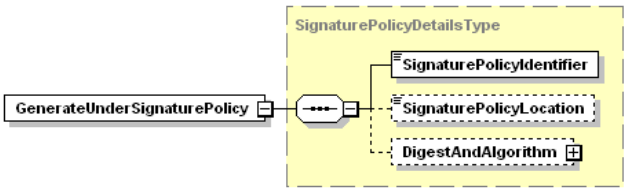
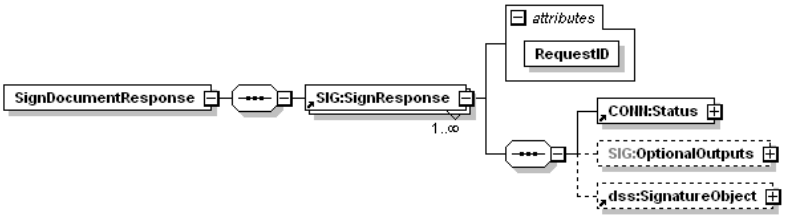
	CONN: Card Handle	Identifiziert die zu verwendende Signaturkarte. Die Operation DARF die Signatur mit der eGK NICHT unterstützen. Wird die Operation mit einem nicht unterstützten Kartentypen aufgerufen, so MUSS der Konnektor die Bearbeitung mit dem Fehler 4126 abbrechen
	CCTX: Context	Aufrufkontext QES mit HBAx: MandantId, ClientSystemId, WorkplaceId, UserId verpflichtend Aufrufkontext nonQES mit SM-B: MandantId, ClientSystemId, WorkplaceId verpflichtend; UserId nicht ausgewertet
	TvMode	Der Parameter wird im Konnektor nicht ausgewertet.
	SIG: JobNumber	Die Nummer des Jobs, unter der der nächste Signaturvorgang gestartet wird. Parameter ist verpflichtend.
	SIG: Sign Request	Ein SignRequest kapselt den Signaturauftrag für ein Dokument. Das verpflichtende XML-Attribut RequestID identifiziert einen SignRequest innerhalb eines Stapels von SignRequests eindeutig. Es dient der Zuordnung der SignResponse zum jeweiligen SignRequest.
	SIG: Optional Inputs	Enthält optionale Eingangsparameter (angelehnt an dss:OptionalInputs gemäß [OASIS-DSS] Section 2.7): 

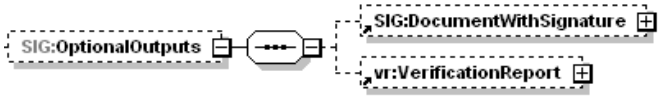
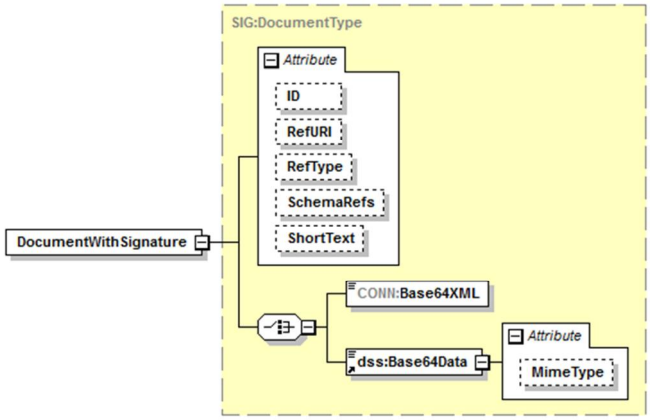
	SIG: Document	 Dieses an das `dss:Document` Element aus [OASIS-DSS] Section 2.4.2 angelehnte Element enthält das zu signierende Dokument, wobei die Kindelemente `CONN:Base64XML` und `dss:Base64Data` auftreten können. Bei den als `dss:Base64Data` übergebenen Dokumenten werden folgende (Klassen von) MIME-Types unterschieden: • "application/pdf-a" – für PDF/A-Dokumente, • "text/plain", "text/plain; charset=iso-8859-15" oder "text/plain; charset=utf-8" – für Text-Dokumente, • "image/tiff" – für TIFF-Dokumente und • ein beliebiger anderer MIME-Type für nicht näher unterschiedene Binärdaten des spezifizierten Typs. Der MIME-Type „text/plain" wird interpretiert als „text/plain; charset=iso-8859-15". Das Element enthält ein Attribut `ShortText` Es muss für QES-Signaturen bei jedem Aufruf vom Clientsystem übergeben werden, für nonQES-Signaturen ist es optional. Über das Attribut `RefURI` kann gemäß [OASIS-DSS] (Abschnitt 2.4.1) ein zu signierender Teilbaum eines XML-Dokuments ausgewählt werden. Wenn die Signatur eines Teilbaums für die Signaturvariante nicht unterstützt wird, muss der Signaturauftrag mit Fehler 4111 abgelehnt werden.
	SIG: IncludeRevocationInfo	Durch diesen verpflichtenden Schalter kann der Aufrufer die Einbettung von zum Zeitpunkt der Signaturerstellung vorliegenden Sperrinformationen anfordern. Es wird ausschließlich die zu erstellende Signatur betrachtet, d.h. es erfolgt keine Einbettung von Sperrinformationen für bereits enthaltene Signaturen. Für nicht-qualifizierte elektronische Signaturen (nonQES) wird diese Funktionalität nicht unterstützt. Für PDF-Signaturen werden keine Sperrinformationen eingebettet.

	`dss: SignatureType`	Durch dieses in [OASIS-DSS] (Abschnitt 3.5.1) beschriebene Element kann der generelle Typ der zu erzeugenden Signaturen spezifiziert werden. Hierbei MÜSSEN folgende Signaturtypen unterstützt werden: • XML-Signatur Durch Übergabe der URI urn:ietf:rfc:3275 wird die Erstellung von XML-Signaturen gemäß [RFC3275], [XMLDSig] angestoßen. Das zu verwendende Profil ist XAdES-BES ([XAdES]). Die Rückgabe einer solchen Signatur erfolgt als `ds:Signature`-Element. • CMS-Signatur Durch Übergabe der URI urn:ietf:rfc:5652 wird eine CMS-Signatur gemäß [RFC5652] angestoßen. Das zu verwendende Profil ist CAdES-BES ([CAdES]). Die Signatur wird als `dss:Base64Signature` mit der oben genannten URI als `Type` zurückgeliefert. • S/MIME-Signatur Durch Übergabe der URI „urn:ietf:rfc:5751" wird eine S/MIME-Signatur gemäß [RFC5751] angestoßen. Die CMS-Signatur der übergebenen MIME-Nachricht erfolgt konform der Vorgaben zur CMS-Signatur. Das Rückgabedokument ist eine MIME-Nachricht vom Typ „application/pkcs7-mime" mit einer CMS-Struktur vom Typ_SignedData. Ist das übergebene Dokument keine MIME-Nachricht, so wie der Fehler 4111 (Ungültiger Signaturtyp oder Signaturvariante) zurückgeliefert. • PDF-Signatur Durch Übergabe der URI http://uri.etsi.org/02778/3 wird die Erzeugung einer PAdES-Basic Signatur gemäß [PAdES-3]angestoßen, wobei das Dokument mit der integrierten Signatur als `dss:Base64Signature` mit der oben genannten URI als `Type` zurückgeliefert wird. Handelt es sich beim übergebenen Dokument nicht um ein `Base64Data`-Element mit MIME-Type „application/pdf-a", so wird ein Fehler 4111 (Ungültiger Signaturtyp oder Signaturvariante) zurückgeliefert. Andere `SignatureType`-Angaben führen zu einer Fehlermeldung 4111 (Ungültiger Signaturtyp oder Signaturvariante). Die Signaturtypen „XML-Signatur, CMS-Signatur, PDF-Signatur, S/MIME-Signatur" DÜRFEN für QES der HBAx nur mit dem QES-Zertifikat erfolgen, für nonQES nur mit dem OSIG-Zertifikat der SM-B. In jedem diese Anforderung verletzenden Fall MUSS der Fehler 4058 (Aufruf nicht zulässig) zurückgeliefert werden. Fehlt dieses Element, so wird der Signaturtyp gemäß TAB_KON_583 – Default-Signaturverfahren aus dem Dokumententyp abgeleitet.

	dss: Properties	Durch dieses in [OASIS-DSS] (Abschnitt 3.5.5) definierte Element können zusätzliche signierte und unsignierte Eigenschaften (Properties) bzw. Attribute in die Signatur eingefügt werden. Unterstützt werden genau folgende Attribute: Im CMS-Fall (SignatureType = urn:ietf:rfc:5652) kann es XML-Elemente ./SignedProperties/Property/Value/CMSAttribute und ./UnsignedProperties/Property/Value/CMSAttribute enthalten. Ein solches XML-Element CMSAttribute muss ein vollständiges, base64/DER-kodiertes ASN.1-Attribute enthalten, definiert in [CMS#5.3.SignerInfo Type]. Es muss bei der Erstellung des CMS-Containers unverändert unter SignedAttributes bzw. UnsignedAttributes aufgenommen werden.
	SIG: Include EContent	Durch dieses in [OASIS-DSS] (Abschnitt 3.5.7), definierte Element kann bei einer CMS-basierten Signatur das Einfügen des signierten Dokumentes in die Signatur angefordert werden. Die Verwendung dieses Parameters bei anderen Signaturtypen führt zu einem Fehler 4111 (Ungültiger Signaturtyp oder Signaturvariante).
	SIG: Include Object	Dieses Element enthält zum Anfordern einer Enveloping XML Signatur ein dss:IncludeObject-Element gemäß [OASIS-DSS] (Abschnitt 3.5.6). Ist das Element vorhanden und ein anderer Signaturtyp als eine XML-Signatur angefordert, so wird der Fehler 4111 (Ungültiger Signaturtyp oder Signaturvariante) zurückgeliefert.
	dss: Signature Placement	Durch dieses in [OASIS-DSS] (Abschnitt 3.5.8) definierte Element kann bei XML-basierten Signaturen gemäß [RFC3275] die Platzierung der Signatur im Dokument angegeben werden. Bei anderen Signaturtypen wird das Element ignoriert und eine Warnung (Fehlercode 4197, Parameter SignaturePlacement wurde ignoriert) zurückgeliefert.

	`dss:ReturnUpdatedSignature`	Durch dieses in [OASIS-DSS] (Abschnitt 4.5.8) definierte Element kann eine übergegebene XML- oder CMS-Signatur mit zusätzlichen Informationen und Signaturen (Parallel- und Gegensignaturen) versehen werden. Hierbei sind folgende Ausprägungen für das Type-Attribut vorgesehen: • http://ws.gematik.de/conn/sig/sigupdate/parallel Hierdurch wird eine Parallelsignatur zu einer bereits existierenden Signatur erzeugt und entsprechend zurückgeliefert. • ~~http://ws.gematik.de/conn/sig/sigupdate/counter/documentincluding~~ ~~Hierdurch wird eine dokumentinkludierende Gegensignatur für das Dokument und alle vorhandenen parallelen Signaturen erzeugt.~~ • http://ws.gematik.de/conn/sig/sigupdate/counter/documentexcluding Hierdurch wird eine dokumentenexkludierende Gegensignatur für alle vorhandenen parallelen Signaturen erzeugt. Bei anderen Type-Attributen wird der Fehler 4111 (Ungültiger Signaturtyp oder Signaturvariante) zurückgeliefert.
	`dss:Schemas`	Durch das in [OASIS-DSS] (Abschnitt 2.8.5) definierte Element können eine Menge von XML-Schemata übergeben werden, die zur Validierung der übergebenen XML-Dokumente verwendet werden können.
		

	dss:Schema	Dieses Element enthält ein XML-Schema zur Validierung des übergebenen XML-Dokuments. Das Attribut RefURI ist verpflichtend. Es kennzeichnet dabei den Namensraum des XML-Schemas entsprechend [OASIS-DSS] (Abschnitt 2.8.5)
	sp: GenerateUnderSignaturePolicy	 Über dieses in [OASIS-SP], Kapitel 2.2.1.1.1.1 Optional Input <GenerateUnderSignaturePolicy>, definierte Element wird die erforderliche Singnaturrichlinie ausgewählt. Die im Element sp:SignaturePolicyIdentifier übergebene URI identifiziert die Signaturrichtlinie. Wenn eine nach TAB_KON_778 notwendige Signaturrichtlinie fehlt oder die übergebene Signaturrichtlinie unbekannt ist, wird Fehler 4111 zurückgeliefert
	SIG: Viewer Info	Enthält optional die vom Konnektor in die Signatur einzubeziehende Referenzen für die Stylesheets zur Anzeige.
Rückgabe		
	SIG: SignResponse	Eine SignResponse kapselt den ausgeführten Signaturauftrag pro Dokument. Die Zuordnung zwischen SignRequest und SignResponse erfolgt über die RequestID.
	CONN: Status	Enthält den Status der ausgeführten Operation pro SignRequest.

	SIG: OptionalOutputs	Enthält (angelehnt an dss:OptionalOutputs) optionale Ausgangsparameter: 
	SIG: DocumentWithSignature	 Pro SignResponse wird ein Element SIG:DocumentWithSignature gemäß [OASIS-DSS] (Abschnitt 3.5.8) zurückgeliefert, in dem das Dokument mit Signatur enthalten ist. Dabei werden die XML-Attribute des Elements SIG:Document auf dem zugehörigen SignRequest übernommen. Ist die Signatur nicht im Dokument enthalten, wird ein leeres Element Base64XML oder Base64Data zurückgegeben. Die Signatur wird dann im Element dss:SignatureObject abgelegt. Wenn die Signatur im Dokument enthalten ist, wird das signierte Dokument im Feld Base64XML bzw. Base64Data zurückgeliefert. In diesem Fall MUSS die dss:SignaturePtr-Alternative in dss:SignatureObject (vgl. [OASIS-DSS] Abschnitt 2.5) dazu genutzt werden, auf die in den Dokumenten enthaltenen Signaturen zu verweisen.
	vr: VerificationReport	Vom Konnektor nicht befüllt.
	dss: SignatureObject	Enthält im Erfolgsfall die erzeugte Signatur pro SignRequest in Form eines dss:SignatureObject-Elementes gemäß [OASIS-DSS] (Abschnitt 3.2).
Vorbedingungen	Keine	

Nachbe-dingungen	Keine

Der Ablauf der Operation SignDocument ist in Tabelle TAB_KON_756 Ablauf Operation SignDocument (nonQES und QES) beschrieben:

Tabelle 214: TAB_KON_756 Ablauf Operation SignDocument (nonQES und QES)

Nr.	Aufruf Technischer Use Case oder Interne Operation	Beschreibung
1.	checkArguments	Anhand des Kartentyps wird ermittelt, ob eine QES oder eine nonQES erzeugt werden soll. Alle übergebenen Parameterwerte werden auf Konsistenz und Gültigkeit überprüft. Treten hierbei Fehler auf, so bricht die Operation mit Fehler 4000 ab.

2.	TUC_KON_000 „Prüfe Zugriffs-berechtigung“	Die Prüfung erfolgt durch den Aufruf
		TUC_KON_000 {
		mandantId = $context.mandantId;
		clientsystemId = $context.clientsystemId;
		workplaceId = $context.workplaceId;
		userId = $context.userId;
		cardHandle = $cardHandle }
		Tritt bei der Prüfung ein Fehler auf, bricht die Operation mit Fehlercode aus TUC_KON_000 ab.

3.	TUC_KON_026 „Liefere CardSession“	Ermittle CardSession über TUC_KON_026 {
		mandatId =$context.mandantId;
		clientsystemId = $context.clientsystemId;
		cardHandle = $context.cardHandle;
		userId = $context.userId }

Im Fall QES wird Schritt 4 ausgeführt. Im Fall nonQES wird Schritt 5 ausgeführt.		
4a)	Prüfe Signaturdienst-Modul	Prüfe, ob MGM_LU_SAK=Enabled. Ist dies nicht der Fall, so bricht die Operation mit Fehler 4125 ab.
4b)	TUC_KON_150 „Dokumente QES signieren“	Die QES wird erzeugt. Tritt hierbei ein Fehler auf, bricht die Operation ab.
5)	TUC_KON_160 „Dokumente nonQES signieren“	Die nonQES wird erzeugt. Tritt hierbei ein Fehler auf, bricht die Operation ab.

Tabelle 215: TAB_KON_757 Fehlercodes „SignDocument (nonQES und QES)“

Fehlercode	ErrorType	Severity	Fehlertext
Neben den Fehlercodes der aufgerufenen TUCs können folgende weiteren Fehlercodes auftreten:			
4000	Technical	Error	Syntaxfehler
4111	Technical	Error	ungültiger Signaturtyp oder Signaturvariante
4126	Security	Error	Kartentyp nicht zulässig für Signatur
4125	Technical	Error	LU_SAK nicht aktiviert

4197	Technical	Warning	Parameter SignaturePlacement wurde ignoriert
4252	Technical	Error	Jobnummer wurde in den letzten 1.000 Aufrufen bereits verwendet und ist nicht zulässig

Die folgende Tabelle führt die zulässigen Zertifikate und Schlüssel für die nonQES auf:

Tabelle 216: TAB_KON_758 Zertifikat und privater Schlüssel je Karte für Sign/VerifyDocument (nonQES)

Karte	Zertifikat (Verify)	Schlüssel (Sign)
SM-B	EF.C.HCI.OSIG.R2048 in DF.ESIGN	PrK.HCI.OSIG.R2048 in DF.ESIGN

Die folgende Tabelle führt die zulässigen Zertifikate und Schlüssel für die QES auf:

Tabelle 217: TAB_KON_759 Zertifikat und privater Schlüssel je Karte für Sign/VerifyDocument (QES)

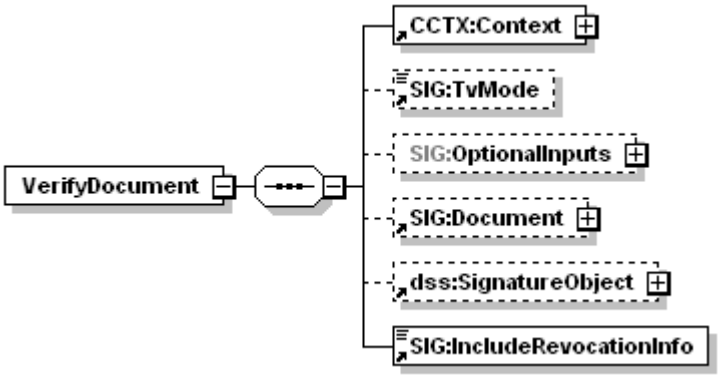
Karte	Zertifikat (Verify)	Schlüssel (Sign)
HBA-VK	EF.C.HP.QES in DF.QES	PrK.HP.QES in DF.QES
HBA	DF.C.HP.QES.R2048 in DF.QES	PrK.HP.QES.R2048 in DF.QES

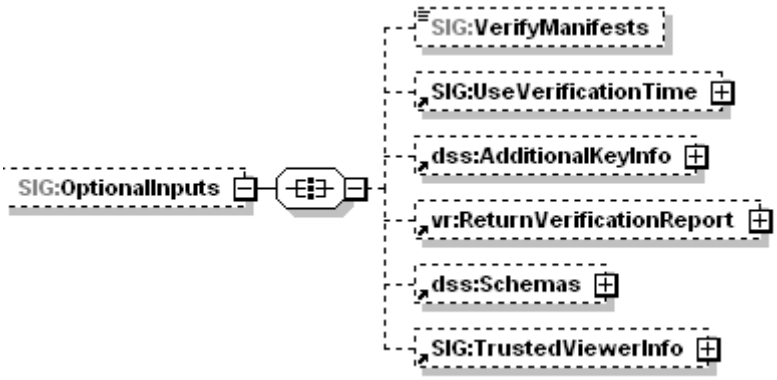
[<=]

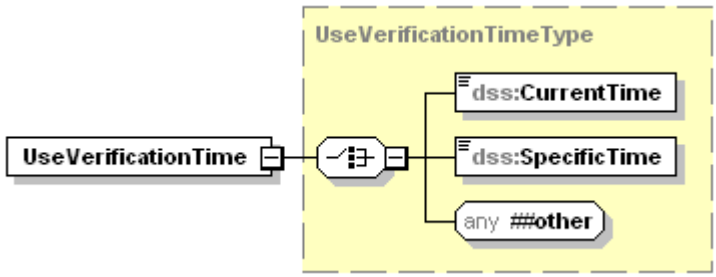
4.1.8.5.2 VerifyDocument (nonQES und QES)

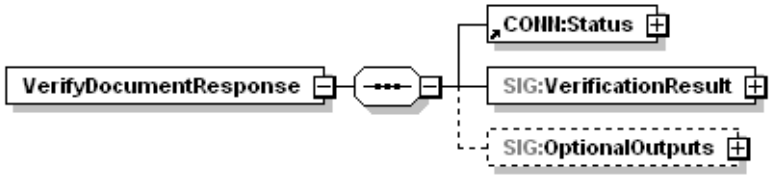
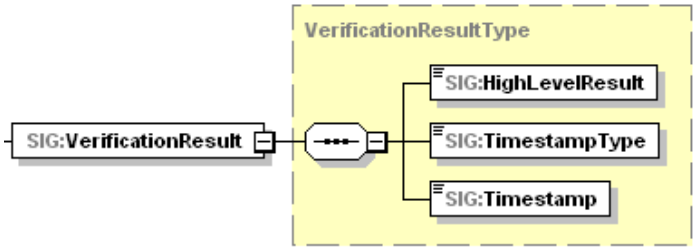
TIP1-A_5034 - Operation VerifyDocument (nonQES und QES)
Der Signaturdienst des Konnektors MUSS an der Clientschnittstelle eine an [OASIS-DSS] angelehnte Operation VerifyDocument (nonQES und QES) anbieten.

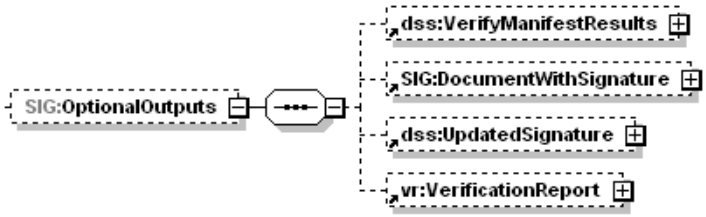
Tabelle 218: TAB_KON_066 Operation VerifyDocument (nonQES und QES)

Name	`VerifyDocument`
Beschreibung	Diese Operation verifiziert die Signatur eines Dokumentes. Der Konnektor MUSS jede konform zur Außenschnittstelle SignDocument erzeugte Signatur durch VerifyDocument prüfen können. Darüber hinaus müssen im Fall QES, falls vorhanden, auch qualifizierte Zeitstempel geprüft werden. Außerdem MÜSSEN die zusätzlich geforderten Signaturverfahren zur Dokumentensignaturprüfung unterstützt werden Das Ergebnis der Prüfung wird, wenn gefordert, in Form eines standardisierten Prüfberichts in einer `VerificationReport`-Struktur gemäß [OASIS-VR] zurückgeliefert.
Aufruf-parameter	

	Name	Beschreibung
	`CCTX: Context`	MandantId, ClientSystemId, WorkplaceId verpflichtend; UserId nicht ausgewertet
	`TvMode`	Der Parameter wird im Konnektor nicht ausgewertet.
	`SIG: Optional Inputs`	Enthält optionale Eingabeparameter (angelehnt an dss:OptionalInputs gemäß [OASIS-DSS] Section 2.7): Die zulässigen optionalen Eingabeparameter sind unten erläutert.
	`SIG: Document`	Enthält im Fall der Prüfung von detached oder enveloped Signaturen das zur Signatur gehörende bzw. das diese umschließende Dokument (siehe [OASIS-DSS] Section 2.4.2 und oben).
	`dss: Signature Object`	Enthält die zu prüfende Signatur, wenn sie nicht im Dokument selbst eingebettet ist ([OASIS-DSS] Kapitel 4.1). Hierbei werden **XML-Signaturen** als `ds:Signature` Element und alle anderen Signaturen als `dss:Base64Signature` mit entsprechend gesetztem `Type`-Attribut (siehe `SignatureType`, Operationen SignDocument und ExternalAuthenticate) übergeben, wobei die nachfolgenden Werte unterstützt werden müssen: • CMS-Signatur urn:ietf:rfc:5652 • S/MIME-Signatur urn:ietf:rfc:5751 • PDF-Signatur http://uri.etsi.org/02778/3 • PKCS#1-Signatur urn:ietf:rfc:3447
	`SIG: Include RevocationInfo`	Durch diesen verpflichtenden Schalter kann der Aufrufer die Einbettung von zum Zeitpunkt der Signaturprüfung vorliegenden Sperrinformationen anfordern. Ist bereits eine Sperrinformation eingebettet, so wird die neue Sperrinformation zusätzlich eingebettet. Für in einer Gegensignatur enthaltene Signaturen erfolgt keine Einbettung von Sperrinformationen. Für PDF-Signaturen erfolgt keine Einbettung von Sperrinformationen. Der Konnektor nimmt die Warnung 4261 in die Antwort auf.
		
	`SIG: Verify Manifests`	Durch das in [OASIS-DSS] (Abschnitt 4.5.1) definierte Element kann die Prüfung eines ggf. vorhandenen Manifests angefordert werden.



	SIG: UseVerificationTime	Durch das in [OASIS-DSS] (Abschnitt 4.5.2) spezifizierte Element kann die Prüfung der Signatur bezüglich eines durch den Aufrufer bestimmten Zeitpunktes (Benutzerdefinierter_Zeitpunkt) erfolgen.
	dss: AdditionalKeyInfo	Durch das in [OASIS-DSS] (Abschnitt 4.5.4) spezifizierte Element kann zusätzliches, für die Prüfung benötigtes, Schlüsselmaterial übergeben werden.
	vr: ReturnVerificationReport	Durch dieses in [OASIS-VR] spezifizierte Element kann die Erstellung eines ausführlichen Prüfberichtes angefordert werden. Der Konnektor MUSS die Anforderungen der Konformitätsstufe 2 („Comprehensive“) erfüllen und die Profilierung aus Anhang B3 beachten.
	dss: Schemas	Durch das in [OASIS-DSS] (Abschnitt 2.8.5) definierte Element können eine Menge von XML-Schematas übergeben werden, die zur Validierung des übergebenen XML-Dokumentes verwendet werden können. Zur Struktur dieses Elements siehe Beschreibung des Parameters dss:Schemas der Operation SignDocument.
	SIG: ViewerInfo	Der Parameter wird im Konnektor nicht ausgewertet.
Rückgabe		
	Status	Enthält den Ausführungsstatus der Operation.
	SIG: VerificationResult	 Das Element Sig:VerificationResult enthält das Ergebnis der Prüfung als Ampel, den Typ des zugehörigen angenommenen Signaturzeitpunkts und der angenommene Signaturzeitpunkt selbst.

	SIG: HighLevelResult	Das Ergebnis der Prüfung (Ampelschaltung) mit folgenden Werten: • VALID: alle Signaturen sind gültig • INVALID: mindestens eine der Signaturen ist ungültig • INCONCLUSIVE: in allen anderen Fällen
	SIG: TimestampType	Der Typ des angenommenen Signaturzeitpunkts mit folgenden Werten: • SIGNATURE_EMBEDDED_TIMESTAMP: in der Signatur eingebetter Zeitpunkt Ermittelter_Signaturzeitpunkt_Eingebettet • QUALIFIED_TIMESTAMP: qualifizierter Zeitstempel über die Signatur Ermittelter_Signaturzeitpunkt_Qualifiziert • SYSTEM_TIMESTAMP: Systemzeit des Konnektors bei Signaturprüfung Ermittelter_Signaturzeitpunkt_System • USER_DEFINED_TIMESTAMP: benutzerdefinierter Zeitpunkt Benutzerdefinierter_Zeitpunkt Als Format darf jedes zum XML-Typ "dateTime" konforme Format verwendet werden (<element name="Timestamp" type="dateTime"/>). Wenn mehrere Signaturen im Dokument vorhanden sind, wird hier der angenommene Signaturzeitpunkt der jüngsten Signatur angegeben.
	SIG: Timestamp	Im Element SIG:Timestamp wird der zu SIG:TimestampType gehörende Zeitstempel zurückgegeben.
	SIG: OptionalOutputs	Enthält (angelehnt an dss:OptionalOutputs, wie in Abschnitt 2.7 von [OASIS-DSS] beschrieben) optionale Ausgangselemente: 
	dss: VerifyManifestResults	Dieses in Abschnitt 4.5.1 von [OASIS-DSS] definierte Element enthält Informationen zur Prüfung eines ggf. vorhandenen Signaturmanifests und wird zurückgeliefert, sofern beim Aufruf das dss:VerifyManifest-Element, aber nicht das RequestVerificationReport als optionales Eingabeelement übergeben wurde.
	SIG: DocumentWithSignature	Dieses in Abschnitt 4.5.8 von [OASIS-DSS] spezifizierte Element wird zurückgeliefert, falls eine in dem Dokument enthaltene Signatur (Enveloped Signature) in Verbindung mit dem SIG:IncludeRevocationInfo-Element geprüft wurde.

	`dss:UpdatedSignature`	Dieses in Abschnitt 4.5.8 von [OASIS-DSS] spezifizierte Element wird zurückgeliefert, falls eine abgesetzte (Detached Signature) oder umschließende (Enveloping Signature) in Verbindung mit dem `SIG:IncludeRevocationInfo`-Element geprüft wurde.
	`vr:VerificationReport`	Dieses in [OASIS-VR] spezifizierte Element wird zurückgeliefert, falls das `ReturnVerificationReport`-Element als Eingabeparameter verwendet wurde. Die Profilierung von Anhang B3 MUSS beachtet werden.
Vorbedingungen	Keine	
Nachbedingungen	Keine	

Tabelle 219: TAB_KON_760 Ablauf Operation VerifyDocument (nonQES und QES)

Nr.	Aufruf Technischer Use Case oder Interne Operation	Beschreibung
1.	checkArguments	Die übergebenen Werte werden auf Konsistenz und Gültigkeit überprüft. Treten hierbei Fehler auf, so bricht die Operation mit Fehler 4000 ab.

2.	TUC_KON_000 „Prüfe Zugriffs-berechtigung“	Die Prüfung erfolgt durch den Aufruf
		TUC_KON_000 {
		mandantId = $context.mandantId;
		clientsystemId = $context.clientsystemId;
		workplaceId = $context.workplaceId;
		needCardSession= false;
		}
		Tritt bei der Prüfung ein Fehler auf, bricht die Operation mit Fehlercode aus TUC_KON_000 ab.

3.	prüfe, ob QES oder nonQES	Ist im jeweiligen Signaturzertifikat mindestens ein QCStatement mit dem OID id-etsi-qcs-QcCompliance (0.4.0.1862.1.1) enthalten, handelt es sich um eine QES-Signatur, andernfalls liegt eine nonQES-Signatur vor.
Für QES-Signaturen wird Schritt 4 ausgeführt. Für nonQES-Signaturen wird Schritt 5 ausgeführt.		
4.a	Prüfe Signaturdienst-Modul	Prüfe, ob MGM_LU_SAK=Enabled. Ist dies nicht der Fall, so bricht die Operation mit Fehler 4125 ab.
4.b	TUC_KON_151 „QES Dokumentensignatur prüfen“	Die QES wird geprüft. Tritt hierbei ein Fehler auf, bricht die Operation ab.
5.	TUC_KON_161 „nonQES Dokumentensignatur prüfen“	Die nonQES wird geprüft. Tritt hierbei ein Fehler auf, bricht die Operation ab.

Tabelle 220: TAB_KON_761 Fehlercodes „VerifyDocument (nonQES und QES)“

Fehlercode	ErrorType	Severity	Fehlertext

Neben den Fehlercodes der aufgerufenen TUCs (siehe Tabelle 220: TAB_KON_760 Ablauf Operation VerifyDocument) können folgende weiteren Fehlercodes auftreten:			
4000	Technical	Error	Syntaxfehler
4261	Technical	Warning	Einbettung von Revocation-Informationen nicht unterstützt
4125	Technical	Error	LU_SAK nicht aktiviert

[<=]

6 Anhang B – Profilierung der Signatur- und Verschlüsselungsformate (normativ)

6.2 Profilierung der Signaturformate

Tabelle 355: TAB_KON_779 „Profilierung der Signaturformate“

Aspekt (QES/nonQES)	**Festlegung** (XML-Signatur/CMS-Signatur/PDF-Signatur)
Zertifikatsreferenz (QES und nonQES)	XML-Signatur Bei der Signaturerstellung ist das XML-Element `SigningCertificate` gemäß den Vorgaben aus XAdES Kapitel 7.2.2 „The SigningCertificate element“ anzulegen. Bei der Signaturprüfung ist es gemäß XAdES Kapitel G.2.2.5 „Verification technical rules” [XAdES] zu prüfen. Grundsätzlich sind auch Signaturen zu prüfen, die keine Zertifikatsreferenz enthalten. Das Prüfergebnis muss dann widerspiegeln, dass diese Sicherheitsfunktion nicht enthalten war. CMS-Signatur Bei der Signaturerstellung ist das Attribut `signing certificate reference` gemäß CAdES Kapitel 5.7.3 „Signing Certificate Reference Attributes“ [CAdES] anzulegen. Bei der Signaturprüfung ist es gemäß CAdES Kapitel 5.6.3 „Message signature verification process“ [CAdES] zu prüfen. Grundsätzlich sind auch Signaturen zu prüfen, die keine Zertifikatsreferenz enthalten. Das Prüfergebnis muss dann widerspiegeln, dass diese Sicherheitsfunktion nicht enthalten war. PDF-Signatur Bei der Signaturerstellung ist das Attribut `signing certificate reference` gemäß den Vorgaben aus [PAdES-3] Kapitel 4.4.3 „Signing Certificate Reference Attribute“ anzulegen. Bei der Signaturprüfung ist es gemäß [PAdES-3]Kapitel 4.6.1 „Signing Certificate Reference Validation“ zu prüfen. Grundsätzlich sind auch Signaturen zu prüfen, die keine Zertifikatsreferenz enthalten. Das Prüfergebnis muss dann widerspiegeln, dass diese Sicherheitsfunktion nicht enthalten war.
Signaturablage	PDF-Signatur Die Signatur wird als Incremental Update gemäß [PDF/A-2] Kapitel 7.5.6 an das Dokument angefügt.
Parallelsignatur (QES und nonQES)	XML-Signatur Parallele Signaturen werden durch je ein `ds:signature`-Element pro Signatur abgebildet. Für die Signaturvariante „eveloping“ werden parallele Signaturen nicht angeboten. CMS-Signatur: Parallele Signaturen werden durch je einen SignerInfo-Container pro Signatur realisiert. PDF-Signatur: Parallele Signaturen werden nicht angeboten.
Dokumentexkludierende Gegensignatur (QES und nonQES)	XML-Signatur Die Implementierung erfolgt mittels Countersignature gemäß [XAdES], Kapitel 7.2.4. Jede vorhandene Parallel-Signatur wird gegensigniert. CMS-Signatur:

	Die Implementierung erfolgt mittels der Countersignature gemäß CMS-Spezifikation [RFC5652]. Jede vorhandene Parallel-Signatur wird gegensigniert. PDF-Signatur: Dokumentexkludierende Gegensignaturen werden nicht angeboten.
~~**Dokumentinkludierende Gegensignatur** (QES und nonQES)~~	~~XML-Signatur~~ ~~Wird als Enveloping XML-Signatur auf dem Gesamtdokument ausgeführt.~~ ~~CMS-Signatur:~~ ~~Dokumentinkludierende Gegensignaturen ist durch Signatur des gesamten SignedData Container zu realisieren.~~ ~~PDF-Signatur:~~ ~~Dokumentinkludierende Gegensignaturen sind gemäß [PAdES-1], Kapitel 4.4 PDF serial signatures, zu realisieren.~~

Änderungsbedarf in gemKPT_Arch_TIP

5.5.1.3 Erstellung_Prüfung_QES

TIP1-A_2254 - Logische Operation I_SAK_Operations::sign_Document_QES
Die Schnittstelle I_SAK_Operations MUSS die logische Operation sign_Document_QES implementieren.

Tabelle : Operation sign_Document_QES

I_SAK_Operations					**Berechtigung: CS, FM**
sign_Document_QES	Parameter				V, I, A
	In	DataToBeSigned	List of DocumentType	IM101	SH/SH/SH (Dokument) SH/SH/SH (Liste)
	In	CuRef	CardUsageReference	IM308	SH/SH/SH
	In	Policies	List of Text	IM302	KS/H/H
	Out	SignedData	List of SignedDocumentType	IM103	SH/M/M (Dokument) SH/SH/SH (Liste)
Mit dieser Operation wird eine qualifizierte elektronische Signatur (QES) gemäß [eIDAS] für jedes der übergebenen Dokumente (*DataToBeSigned*) erzeugt. Die QES wird mit dem HBA unter Verwendung der kryptographischen Identität ID.HP.QES des HBA-Inhabers erstellt. Zu nutzende Karten sind zeitlich begrenzt auch die HBA-Vorläuferkarten HBA-qSig und ZOD-2.0. Zugriffsrechte auf die zu verwendende Karte werden anhand der übergebenen CardUsageReference (*CuRef*) geprüft. Es wird die Übereinstimmung der Eigenschaften der Dokumente mit den Vorgaben der übergebenen Signaturrichtlinien ~~Policies~~ (*Policies)* überprüft. Die Policies beinhalten spezifische Signaturformatfestlegungen und Darstellungsvorgaben für die jeweils verwendeten Datenformate. Es werden nur vorab bekannte Policies unterstützt. Für XML-Dokumente beinhalten die Policies ein XML-Schema, gegen welches das XML-Dokument geprüft wird. Sollte die Schemaprüfung fehlschlagen, wird die Signaturerstellung abgebrochen. Des Weiteren muss vor Erstellung der Signatur geprüft werden, ob die Gültigkeitsdauer des Signaturzertifikats überschritten ist. Die erzeugte Signatur wird jeweils entsprechend des angewendeten Signaturformats in das Ergebnisdokument (*SignedData*) eingebettet. Als Signaturzeitpunkt wird die Systemzeit zum Zeitpunkt der Erstellung verwendet. Sofern verfügbar wird die aktuelle Sperrinformation (OCSP-Response) des Signaturzertifikats in das Ergebnisdokument (*SignedData*) eingebettet. Am Ende der Operation wird das/werden die signierten Dokumente an den Aufrufer übergeben (*SignedData*).					

Es werden die nachfolgenden Dokumententypen mit dem genannten Signaturformat unterstützt: · Text und TIFF mit CMS · XML mit XAdES · PDF/A mit PDF-Signatur Es werden die folgenden Formen der Signatur unterstützt: · Einzelsignatur für alle angegebenen Formate · Stapelsignatur für alle angegebenen Formate · Parallelsignatur für die Formate Text, TIFF und XML · Gegensignatur für alle angegebenen Formate außer PDF-Signatur Bei Nichtvorhandensein der Konfiguration LU_SAK muss die Operation unmittelbar mit einer Fehlermeldung abgebrochen werden bzw. darf nicht angeboten werden. Eigenschaften der Stapelsignatur Im Falle der Stapelsignatur enthält der Parameter *DataToBeSigned* eine Liste von zu signierenden Dokumenten. · Jedes Dokument des Stapels wird einzeln qualifiziert signiert. · Stapelsignatur ist für alle für die Einzelsignatur unterstützten Formate möglich. · Gemischte Formate innerhalb eines Stapels sind möglich. · Innerhalb eines Stapels werden Erst-, Gegen- und Parallelsignatur auch in gemischter Form unterstützt. · Die Stapelgröße muss unabhängig von Limitierungen auf dem HBA festgelegt werden. · Die Stapelsignatur fordert für jeden Stapel vor dem Signieren der Dokumente einmal eine PIN-Eingabe des Benutzers und signiert die Dokumente eines Stapels in unmittelbarer Folge ohne wiederholte PIN-Eingabe des Benutzers. Wenn die festgelegte Stapelgröße die Limitierung auf dem HBA übersteigt, werden Teilstapel gebildet, für die jeweils eine separate PIN-Eingabe erforderlich ist. · Dokumente verschiedener Versicherter können innerhalb eines Stapels signiert werden. · Die Stapelsignatur kann bis zum Auslösen der qualifizierten elektronischen Signaturen (PIN-Eingabe) und während der Stapelabarbeitung kontrolliert abgebrochen werden. Die Operation unterstützt mindestens Dokumente bis zu einer Größe von 25 MB. Die Performancevorgaben gelten für Einzelsignaturen.
Verfügbarkeit: NA, Nichtabstreitbarkeit: SH

TIP1-A_2255 - Logische Operation I_SAK_Operations::verify_Document_QES

Die Schnittstelle I_SAK_Operations MUSS die logische Operation verify_Document_QES implementieren.

Tabelle : Operation verify_Document_QES

I_SAK_Operations					**Berechtigung: CS, FM**
verify_Document_QES	Parameter				V, I, A
	In	SignedData	SignedDocumentType	IM103	SH/M/M
	In	Certificate	CertificateX.509	IM404	M/M/M
	~~In~~	~~Policies~~	~~Text~~	~~IM302~~	~~KS/H/H~~
	Out	VerificationResult	VerificationResultType	IM420	M/H/H
Diese Operation überprüft die qualifizierte elektronische Signatur (QES) des übergebenen Dokuments (*SignedData*) gemäß [eIDAS] unter Verwendung des mit dem Dokument übergebenen Signaturzertifikats. Das Signaturzertifikat muss entweder bereits im signierten Dokument enthalten sein oder über den optionalen Parameter *Certificate* separat übergeben werden. Es wird zuerst die Gültigkeit des Signaturzertifikats durch Nachnutzung des Dienstes „Prüfung_Zertifikat“ geprüft. Dies					

umfasst die Prüfung im Online- wie auch im Offline-Fall. Sollte das übergebene Dokument (*SignedData*) eine Sperrinformation (OCSP-Response) für das Signaturzertifikat enthalten, so wird diese bei der Prüfung des Zertifikates verwendet. War das genutzte Zertifikat bei Erstellung der Signatur nicht gültig, dann ist auch die Signatur im rechtlichen Sinn nicht gültig. Im Ergebnis der Operation (*VerificationResult*) wird dokumentiert, ob die Prüfung erfolgreich war oder ob sie fehlgeschlagen ist. Falls die Prüfung nicht vollständig erfolgen konnte, da z.B. die Online-Statusprüfung des Zertifikats nicht möglich war (Offline-Fall), muss dies dem Nutzer mitgeteilt werden. Dazu werden die durchgeführten Prüfschritte im Ergebnis der Operation (*VerificationResult*) aufgeführt. Falls ein Algorithmus oder Parameter, der zur Signatur genutzt wurde, nicht mehr als geeignet betrachtet wird, muss die Signaturprüfung trotzdem durchgeführt werden. Das Ergebnis der Signaturprüfung muss im Parameter *VerificationResult* enthalten sein. In den *SignedData* enthaltene qualifizierte Zeitstempel werden ausgewertet. Vor der Prüfung der Signatur muss der Status der verwendeten Algorithmen gegen den aktuell gültigen Algorithmenkatalog der zuständigen Behörde (BNetzA) geprüft werden. Es werden die nachfolgenden Dokumententypen mit dem genannten Signaturformat unterstützt: · Text und TIFF mit CMS · XML mit XAdES · PDF/A mit PDF-Signatur Bei der Prüfung werden die folgenden Formen der Signatur unterstützt: · Einzelsignatur für alle angegebenen Formate · Parallelsignatur für die Formate Text, TIFF und XML · Gegensignatur für alle angegebenen Formate außer PDF-Signatur Bei Nichtvorhandensein der Konfiguration LU_SAK muss die Operation unmittelbar mit einer Fehlermeldung abgebrochen werden bzw. darf nicht angeboten werden. Es werden nur Signaturen der kryptographischen Identitäten von Leistungserbringern (zulässige Karten: HBA wie auch zeitlich begrenzt die HBA-Vorläuferkarten HBA-qSig und ZOD-2.0) geprüft. Die Operation unterstützt mindestens Dokumente bis zu einer Größe von 25 MB.
Verfügbarkeit: NA, Nichtabstreitbarkeit: NA