Es wird die Tabelle TAB_KON_862 angepasst, und damit vier Anforderungen zur QES Dokumentensignatur:

- *TIP1-A_4651: TUC_KON_154 „QES Signaturen erstellen"*
- *TIP1-A_4652: TUC_KON_168 „Einzelsignatur QES erstellen*
- *TIP1-A_4655: TUC_KON_150 „Dokument QES signieren„*
- *TIP1-A_5010-02: Operation SignDocument (nonQES und QES)*

Änderung in gemSpec_Kon_V5.9.0

Es wird in Kapitel 4.1.8.1.1 angepasst: (4.1.8 Signaturdienst)

1.1.1.1.1 Dokumentensignatur

Tabelle 1: TAB_KON_862-01 Werteliste und Defaultwert des Parameters crypt bei QES-Erzeugung

Typname	Werteliste	Defaultwert	Bedeutung
SIG_CRYPT_QES	RSA ECC RSA_ECC	RSA ~~RSA_ECC~~	Werteliste des Parameters crypt bei der bei der Erzeugung einer QES-Signatur RSA: Es wird eine RSA-2048 Signatur erzeugt. ECC: Es wird eine ECC-256 Signatur erzeugt. RSA_ECC: In Abhängigkeit von der Kartengeneration wird eine RSA-2048 bzw. eine ECC-256 Signatur erzeugt (siehe TAB_KON_900).

TIP1-A_4651-02 - TUC_KON_154 „QES Signaturen erstellen"
Der Konnektor MUSS den technischen Use Case TUC_KON_154 „QES Signaturen erstellen" umsetzen.

Tabelle 2: TAB_KON_752 – TUC_KON_154 „QES Signaturen erstellen"

Element	Beschreibung
Name	TUC_KON_154 „QES Signaturen erstellen"
Beschreibung	Die Data To Be Signed (DTBS) werden erzeugt, an die Signaturkarte gesendet und dort signiert.
Auslöser	TUC_KON_150 „Dokumente QES signieren"

Vorbedingungen	Die Ressourcen Signaturkarte und Kartenterminal sind für den Vorgang reserviert. DF.QES ist selektiert. PIN.QES ist initial verifiziert
Eingangsdaten	• Zu signierendes Dokument bzw. zu signierende Dokumente • cardSession (nur HBA erlaubt) • zu verwendende Identität (Zertifikatsreferenz) • crypt [SIG_CRYPT_QES] - *optional*; *default und* Werte*bereich*: siehe TAB_KON_862-01 (Dieser Parameter steuert, ob RSA-basierte oder ECC-basierte Signaturen erzeugt werden.) • WorkplaceId
Komponenten	Konnektor, Kartenterminal, Signaturkarte (HBA)
Ausgangsdaten	• Signierte Dokumente

Standardablauf	Basierend auf SAK.AUTD_CVC und HPC.AUTD_SUK_CVC und den zugehörigen privaten Schlüsseln wird ein sicherer Kanal zwischen der gSMC-K des Konnektors und dem HBA aufgebaut mittels Aufruf TUC_KON_005 „Card-to-Card authentisieren“ {
	sourceCardSession = gSMC-K;
	targetCardSession = CardSession;
	authMode = „gegenseitig+TC“}
	
	Die folgenden Schritte werden für jedes Dokument des Stapels durchgeführt.
	1. Das zu signierende Dokument wird, soweit noch nicht erfolgt, für die Signatur gemäß des entsprechenden Formats vorbereitet. Ein Ergebnis dieser Vorbereitung sind die Data To Be Signed (DTBS): der Hash-Wert (Digest des SignedInfo-Elementes), der zur Signatur an die Karte gesendet werden soll.
	2. Für das zu signierende Dokument werden die DTBS zur Signatur im sicheren Kanal an den HBA übermittelt (Aufruf TUC_KON_218 „Signiere“). Dabei werden der Schlüssel und der Algorithmusidentifier über die Tabelle TAB_KON_900 bestimmt.
	3. Falls Schritt 3 fehlgeschlagen ist, weil der PIN.QES-Nutzungszähler abgelaufen ist (erkennbar z. B. daran, dass die Karte einen Autorisierungsfehler zurückmeldet), wird die PIN.QES verifiziert (Aufruf TUC_KON_012 „PIN verifizieren“, nachdem der im Konnektor verwaltetete Sicherheitszustand (CARDSESSION.AUTHSTATE) aktualisiert wurde).

	Am Display des Kartenterminals wird dabei die Jobnummer für den Signaturvorgang angezeigt. Aus der WorkplaceId geht hervor, ob es sich um eine Remote-PIN-Eingabe handelt. Nach der PIN-Verifikation wird erneut die zuvor fehlgeschlagene Signatur in Schritt 3 ausgeführt. 4. Die erstellte Signatur wird mathematisch geprüft. 5. Der ermittelte Signaturwert wird in den zuvor vorbereiteten Signaturprototypen eingefügt. 6. Der Konnektor löst TUC_KON_256 {"SIG/SIGNDOC/NEXT_SUCCESSFUL"; Op; Info; „$Jobnummer"; noLog; $doInformClients } aus.
Varianten/ Alternativen	Alternativ zum Standardablauf kann zu Beginn die maximal erlaubte Stapelgröße SSEC durch Auslesen von EF.SSEC ermittelt werden. Der zu signierende Dokumentenstapel wird in Teilstapel von maximaler Größe SSEC zerlegt. Für jeden Teilstapel wird die PIN.QES verifiziert. Die Dokumente des Teilstapels werden wie im Standardablauf beschrieben signiert. Der Nutzer kann den Vorgang der PIN-Eingabe abbrechen.
Fehlerfälle	(->2) Fehler im Signaturvorgang führen zum Abbruch des gesamten Signaturvorgangs, Fehlercode 4123 (->3) Fehler bei der PIN-Eingabe führen zum Abbruch des Signaturvorgangs (->4) Fehler in mathematischer Prüfung der Signatur führen zum Abbruch des Signaturvorgangs, Fehlercode 4120 Das Verhalten des TUCs bei einem Fehlerfall, einem Timeout der PIN-Eingabe oder beim Abbruch durch den Benutzer ist in Tabelle TAB_KON_192 Verhalten des Konnektors beim Abbruch einer Stapelsignatur beschrieben.
Sicherheitsanforderungen	Zum Aufbau des sicheren Kanals bzw. zur Aushandlung des symmetrischen Schlüssels DARF DF.QES NICHT verlassen werden. Benötigte CVCs des HBA MÜSSEN also bereits vor dem Signaturvorgang eingelesen und gecacht werden. Dies KANN bereits beim Stecken des HBA geschehen. Die in [gemSpec_Krypt#3.1.2] angegebenen Festlegungen der zu unterstützenden Algorithmen MÜSSEN berücksichtigt werden.
Nichtfunktionale Anforderungen	keine
Zugehörige Diagramme	Abbildung PIC_KON_113 Aktivitätsdiagramm zu „QES Signaturen erstellen" Das Diagramm dient nur der Veranschaulichung und ist nicht vollständig. Beispielsweise enthält es nicht die Steuerung durch den Parameter crypt.

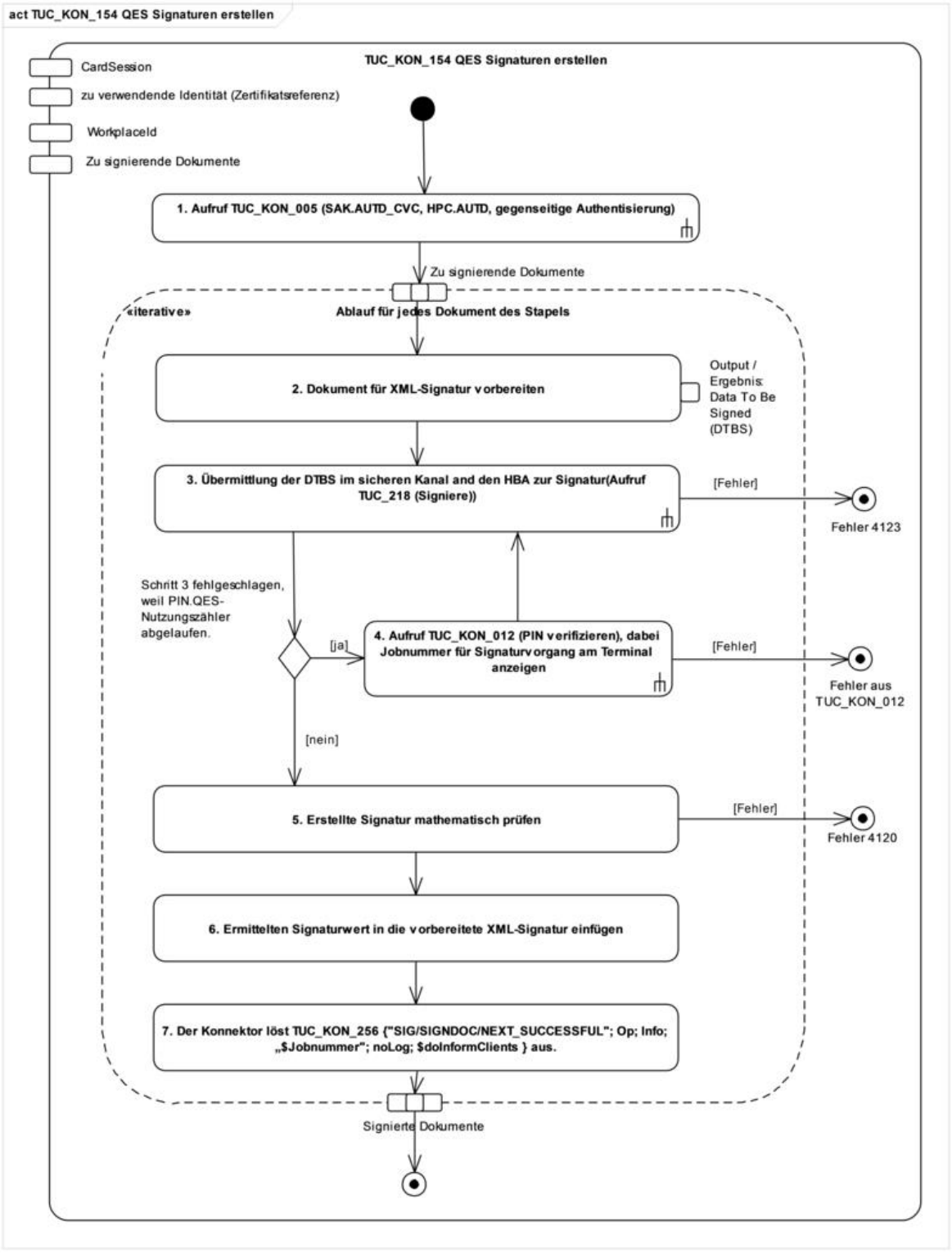


Abbildung 1: PIC_KON_113 Aktivitätsdiagramm zu „QES Signaturen erstellen"

Tabelle 3: TAB_KON_126 Fehlercodes TUC_KON_154 „QES Signaturen erstellen"

Fehlercode	ErrorType	Severity	Fehlertext

Neben den Fehlercodes der aufgerufenen technischen Use Cases können folgende weitere Fehlercodes auftreten:			
4120	Security	Error	Kartenfehler
4123	Security	Error	Fehler bei Signaturerstellung

[<=]

TIP1-A_4652-02 - TUC_KON_168 „Einzelsignatur QES erstellen"
Der Konnektor MUSS den technischen Use Case TUC_KON_168 „Einzelsignatur QES erstellen" umsetzen.

Tabelle 4: TAB_KON_293 - TUC_KON_168 „Einzelsignatur QES erstellen"

Element	Beschreibung
Name	TUC_KON_168 "Einzelsignatur QES erstellen"
Beschreibung	Es wird ein Dokument technisch mit einer Signatur versehen. Im Gegensatz zum TUC_KON_154 „QES Signaturen erstellen" wird hier nur eine einzelne Signatur ohne vorhergehendes C2C erstellt. Die Übertragung der DTBS erfolgt ohne Secure Messaging.
Auslöser	TUC_KON_150 Dokumente QES signieren
Vorbedingungen	Die Ressourcen Signaturkarte und Kartenterminal sind für den Vorgang reserviert. DF.QES ist selektiert.
Eingangsdaten	• zu signierendes Dokument • CardSession (HBAx) • zu verwendende Identität (Zertifikatsreferenz) • crypt: [SIG_CRYPT_QES] - *optional*; *default und* Werte*bereich*: siehe TAB_KON_862-01 Dieser Parameter steuert, ob RSA-basierte oder ECC-basierte Signaturen erzeugt werden. • WorkplaceId
Komponenten	Konnektor, Kartenterminal, Signaturkarte (HBAx)
Ausgangsdaten	• Signiertes Dokument
Standardablauf	1. Das zu signierende Dokument wird, soweit noch nicht erfolgt, für die Signatur vorbereitet. Ein Ergebnis dieser Vorbereitung sind die DTBS: der Hash-Wert (Digest des SignedInfo-Elementes), der zur Signatur an die Karte gesendet werden soll. 2. Für das zu signierende Dokument werden die DTBS zur Signatur an den HBAx übermittelt (Aufruf TUC_KON_218 „Signiere"). Dabei werden der Schlüssel und der Algorithmusidentifier über die Tabelle TAB_KON_900

	bestimmt. Jeder Fehler führt zum Abbruch des Signaturvorgangs 3. Die erstellte Signatur wird mathematisch geprüft. Der ermittelte Signaturwert wird in den zuvor gemäß des entsprechenden Signaturformates vorbereiteten Signaturprototypen eingefügt.
Varianten/ Alternativen	keine
Fehlerfälle	Das Verhalten des TUCs bei einem Fehlerfall, einem Timeout der PIN-Eingabe oder beim Abbruch durch den Benutzer ist in Tabelle TAB_KON_192 Verhalten des Konnektors beim Abbruch einer Stapelsignatur beschrieben. (→3) Fehler in mathematischer Prüfung der Signatur: Abbruch mit 4120
Nichtfunktionale Anforderungen	keine
Zugehörige Diagramme	keine

Tabelle 5: TAB_KON_590 Fehlercodes TUC_KON_168 „Einzelsignatur QES erstellen"

Fehlercode	ErrorType	Severity	Fehlertext
Neben den Fehlercodes der aufgerufenen technischen Use Cases können folgende weitere Fehlercodes auftreten.			
4120	Security	Error	Kartenfehler

[<=]

TIP1-A_4655-02 - TUC_KON_150 „Dokument QES signieren„
Der Konnektor MUSS den technischen Use Case TUC_KON_150 „Dokumente QES signieren" umsetzen.

Tabelle 6: TAB_KON_755 – TUC_KON_150 „Dokumente QES signieren"

Element	Beschreibung
Name	TUC_KON_150 "Dokumente QES signieren"
Beschreibung	Im Rahmen von Fachanwendungen werden ein oder mehrere Dokumente mit einer qualifizierten elektronischen Signatur versehen. Es werden die QES_DocFormate unterstützt.
Auslöser	Aufruf durch ein Clientsystem (Operation SignDocument) oder ein Fachmodul.

Vorbedingungen	Die Signaturkarte muss gesteckt sein.

Eingangsdaten	• signRequests
	(Liste von Signaturaufträgen)
	Jeder Signaturauftrag (SignRequest) kapselt:
	  • documentsToBeSigned
	(Zu signierendes Dokument bzw. zu signierende Dokumente);
	darin u.a.
	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-, PDF-Signatur)
	  • includeRevocationInfo [Boolean]: – optional; Default: true
	(Dieser optionale Parameter steuert die Einbettung von OCSP Antworten in die Signatur; siehe Operation SignDocument, Parameter SIG:IncludeRevocationInfo)
	• cardSession
	(Kartensitzung. Unterstützte Kartentypen: HBAx)
	• crypt [SIG_CRYPT_QES] - *optional*;
	default und Wertebereich: siehe TAB_KON_862-01
	(Dieser Parameter steuert, ob RSA-basierte oder ECC-basierte Signaturen erzeugt werden.)
	• workplaceId

Komponenten	Konnektor, Kartenterminal, Signaturkarte (HBAx)
Ausgangsdaten	• signedDocuments (Liste der signierten Dokumente)
Standardablauf	Der Konnektor KANN die Schritte 1 bis 4 in einer beliebigen Reihenfolge durchführen. 1. Der Signaturtyp und die Signaturvariante werden für jedes Dokument der Liste entsprechend signatureType und SignatureVariant festgelegt (ggf. in optionalInputs enthalten). Wenn SignatureType oder SignatureVariant nicht übergeben wurden, wird das dem Dokumentformat entsprechende Default-Verfahren gewählt (siehe TAB_KON_583 – Default-Signaturverfahren). 2. Für alle Dokumente des Stapels wird die Zulässigkeit des Kartentyps geprüft. Das für die Signatur zu nutzende

	Zertifikat wird anhand des Kartentyps und des Parameters crypt ausgewählt.
	3. Es werden die Voraussetzungen für die Signatur geprüft. Dies erfolgt im TUC_KON_152 „Signaturvoraussetzungen für QES prüfen".
	Wenn includeRevocationInfo=true, dann setze ocspResponses auf Rückgabewert von TUC_KON_152.
	4. Die am Signaturvorgang beteiligten Ressourcen (Signaturkarte sowie PIN Pad und Display des PIN-Eingabe-Kartenterminals) werden für die exklusive Nutzung durch diesen Signaturvorgang reserviert. Die Reservierung der Signaturkarte erfolgt durch Aufruf von TUC_KON_023 „Karte reservieren" {
	cardSession;
	doLock = true }.
	5. Zum Vorbereiten der Dokumente für die Signatur wird TUC_KON_155 „Dokumente zur Signatur vorbereiten" mit ocspResponses aufgerufen.
	Die Zugriffe auf die Signaturkarte in den Schritten 6 bis 7 müssen im DF.QES erfolgen.
	6. Die Signaturerstellung wird durch den Anwender autorisiert. Dies erfolgt durch Aufruf von TUC_KON_012 „PIN verifizieren" { cardSession;
	workplaceId;
	pinRef = PIN.QES;
	verificationType = Mandatorisch }
	Wenn nur ein zu signierendes Dokument vorhanden ist und der Einfachsignaturmodus aktiviert ist (siehe Konfigurationsparameter SAK_SIMPLE_SIGNATURE_ MODE), wird in Schritt 7 Variante a) durchgeführt, ansonsten Variante b).
	7. Variante a) Die Signatur wird erstellt. Dies erfolgt gemäß TUC_KON_168 „Einzelsignatur QES erstellen".
	Variante b) Die Signaturen werden erstellt. Dies erfolgt gemäß TUC_KON_154 „QES-Signaturen erstellen".
	8. Es wird DF.QES verlassen, um den PIN-Status der PIN.QES zurückzusetzen. Der im Konnektor verwaltetete Sicherheitszustand (CARDSESSION.AUTHSTATE) ist zu aktualisieren.
	9. Die reservierten Ressourcen (Signaturkarte sowie PIN-Pad und
	Display des PIN-Eingabe-Kartenterminals) werden wieder freigegeben.
	Zur Freigabe der Signaturkarte wird TUC_KON_023 „Karte reservieren"
	cardSession;
	doLock = false }
	aufgerufen.
	10. Die signierten Dokumente werden an den Aufrufer zurückgegeben.

Varianten/ Alterna-tiven	Der Nutzer kann den Vorgang bei der Autorisierung (Schritt 6) abbrechen. Hierbei sind die gleichen Regeln anzuwenden wie im Fehlerfall (s. Fehlerfälle).
Fehlerfälle	Fehler in den folgenden Schritten des Standardablaufs führen zum Abbruch mit den ausgewiesenen Fehlercodes: (->1) Ungültige Angabe des Signaturtyps oder Signaturvariante: Fehlercode 4111 Übergabe eines für die QES nicht unterstützten Dokumentformats: Fehlercode 4110 (->2) Kartentyp nicht zulässig für Signatur: Fehlercode 4126 (->5) Fehler bei der Reservierung von Ressourcen: Fehlercode 4060 (->7b) Karte ist kein HBA, sondern HBA-Vorläuferkarte: Fehlercode 4118 Im Fehlerfall, inklusive Timeout bei der PIN-Eingabe, oder bei Abbruch durch den Benutzer (Fehler 4049): a) … MUSS DF.QES verlassen werden b) … MÜSSEN alle reservierten Ressourcen freigegeben werden c) … MUSS der Fehler immer an das Clientsystem zurückgemeldet werden
Nichtfunktionale Anforderungen	keine
Zugehörige Diagramme	Abbildung PIC_KON_114 Aktivitätsdiagramm zu „Dokument QES signieren“

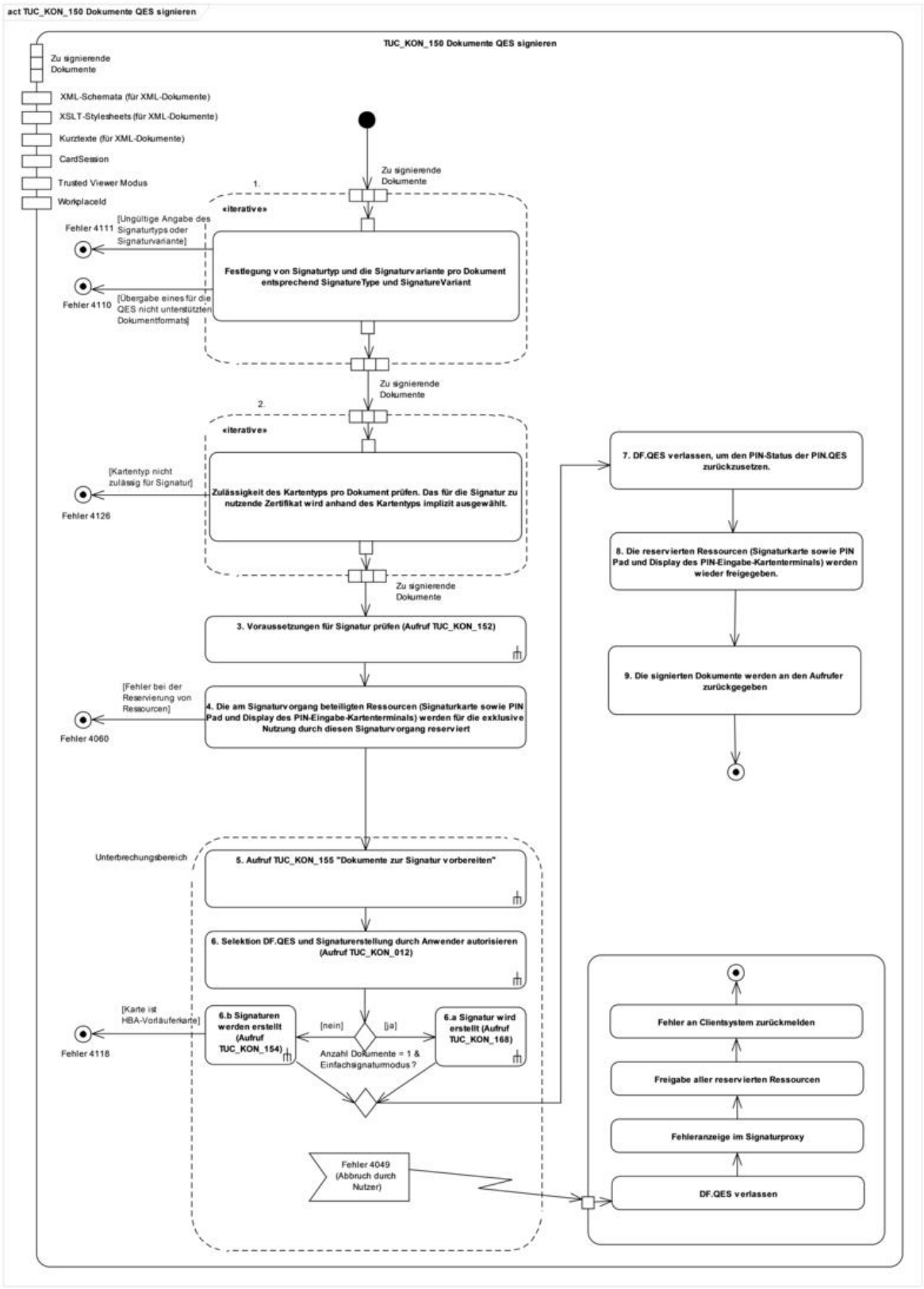


Abbildung 2: PIC_KON_114 Aktivitätsdiagramm zu „Dokument QES signieren"

Tabelle 7: TAB_KON_128 Fehlercodes TUC_KON_150 „Dokument QES signieren“

Fehlercode	ErrorType	Severity	Fehlertext
Neben den Fehlercodes der aufgerufenen technischen Use Cases können folgende weitere Fehlercodes auftreten:			
4060	Technical	Error	Ressource belegt
4110	Technical	Error	ungültiges Dokumentformat (%Format%) Der Parameter Format enthält das übergebene Dokumentformat.
4111	Technical	Error	ungültiger Signaturtyp oder Signaturvariante
4118	Technical	Error	Stapelsignaturen werden nur für den HBA unterstützt. Mit HBA-Vorläuferkarten sind nur Einzelsignaturen möglich.
4126	Security	Error	Kartentyp nicht zulässig für Signatur
4049	Technical	Error	Abbruch durch den Benutzer

[<=]

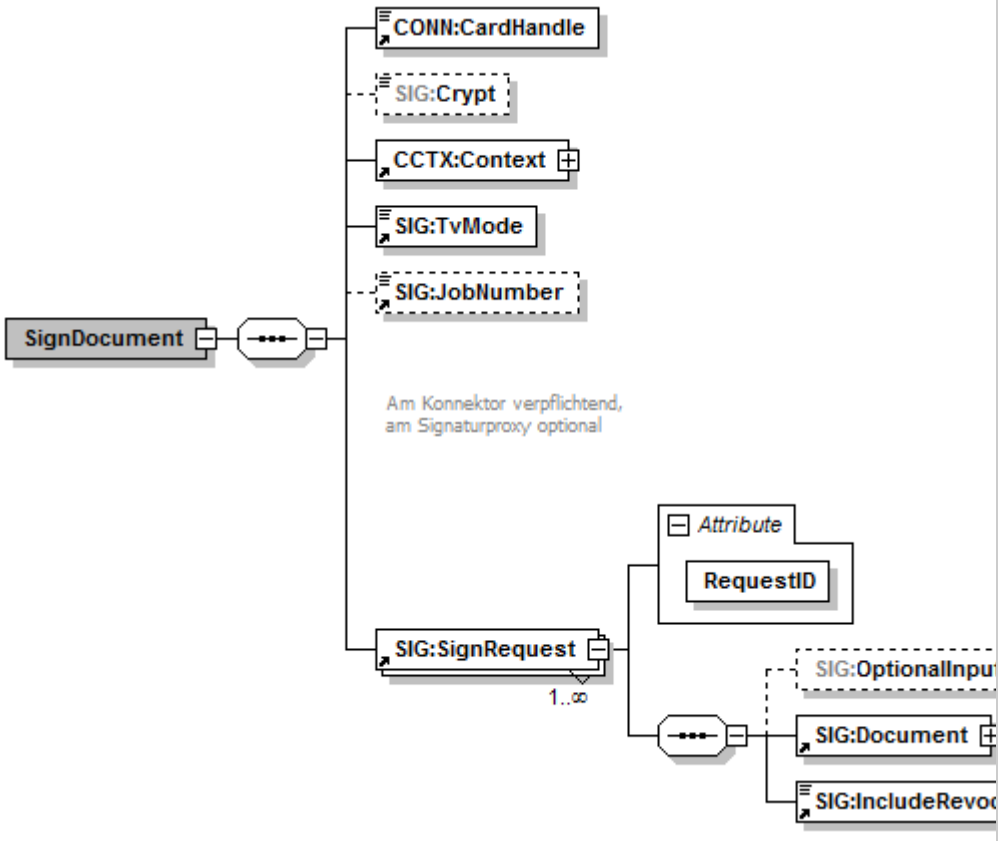
In Kapitel 4.1.8.5.1:

1.1.1.1.2 SignDocument (nonQES und QES)

TIP1-A_5010-05 - Operation SignDocument (nonQES und QES)
Der Signaturdienst des Konnektors MUSS an der Clientschnittstelle eine an [OASIS-DSS] angelehnte Operation SignDocument anbieten**.**

Tabelle 8: TAB_KON_065 Operation SignDocument (nonQES und QES)

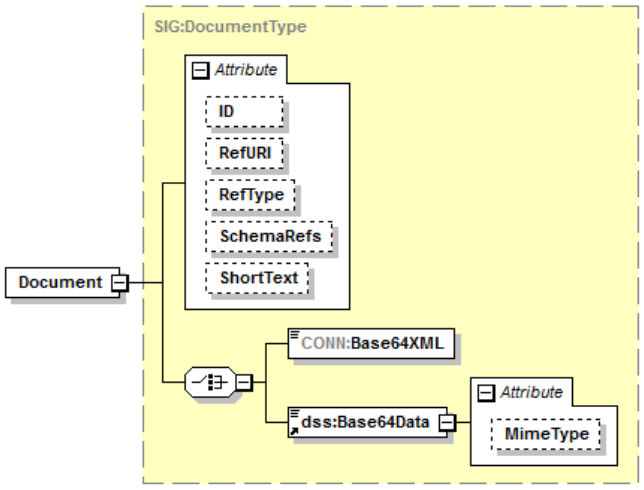
Name	`SignDocument`

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.
Aufruf-parameter	

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.
	SIG: Crypt	Der Parameter crypt steuert die Auswahl der Zertifikate und Schlüssel für die Signaturerstellung abhängig von der durch cardHandle adressierten Karte gemäß TAB_KON_900. Defaultwert: • gemäß TAB_KON_862-01 für die QES • gemäß TAB_KON_863 für die nonQES.
	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.

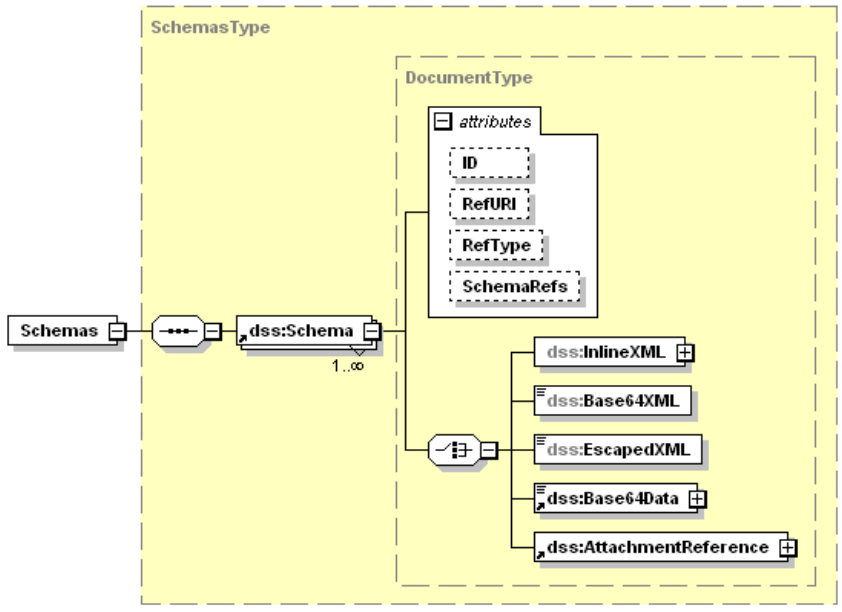
	SIG: Include Revocation Info	Durch diesen verpflichtenden Schalter kann der Aufrufer die Einbettung von zum Zeitpunkt der Signaturerstellung vorliegenden Sperrinformationen anfordern. Es wird ausschließlich die zu erstellende Signatur betrachtet, d.h. es erfolgt keine Einbettung von Sperrinformationen für bereits enthaltene Signaturen. Für nicht-qualifizierte elektronische Signaturen (nonQES) wird diese Funktionalität nicht unterstützt. Für PDF-Signaturen werden keine Sperrinformationen eingebettet.

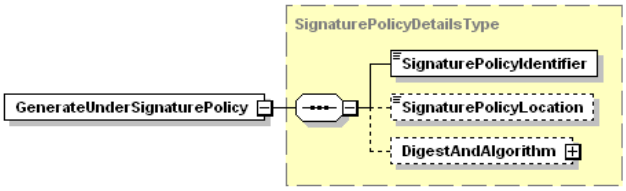
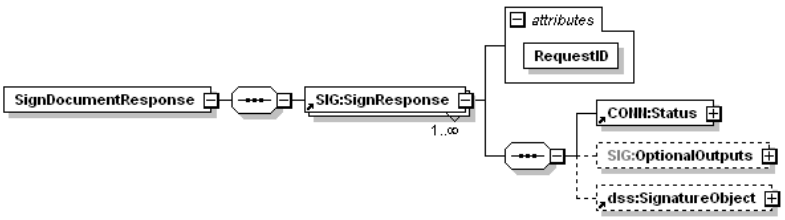
	dss:SignatureType	Durch dieses in [OASIS-DSS] (Abschnitt 3.5.1) beschriebene Element kann der generelle Typ der zu erzeugenden Signaturen spezifiziert werden. Hierbei MÜSSEN folgende Signaturtypen unterstützt werden: • **XML-Signatur** Durch Übergabe der URI urn:ietf:rfc:3275 wird die Erstellung von 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).

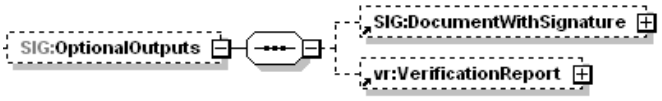
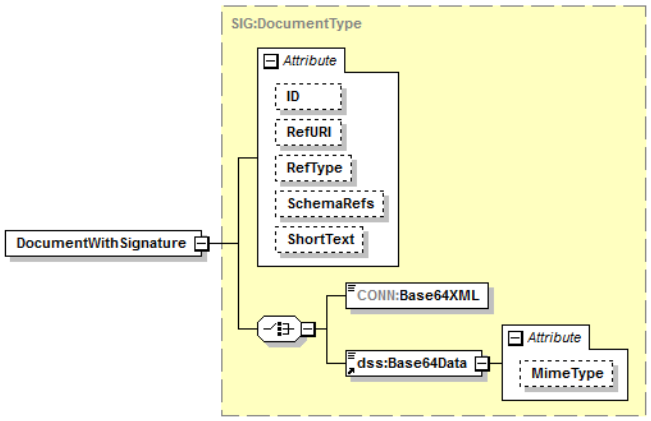
		Die Signaturtypen „XML-Signatur, CMS-Signatur, PDF-Signatur, S/MIME-Signatur" DÜRFEN für QES der HBAx nur mit dem QES-Zertifikat erfolgen, für nonQES nur mit dem OSIG-Zertifikat der SM-B. In jedem diese Anforderung verletzenden Fall MUSS der Fehler 4058 (Aufruf nicht zulässig) zurückgeliefert werden. 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. Die in [OASIS-DSS] (Abschnitt 2.5, XPath c) beschriebene Deklaration von Namespace-Prefixes im dss:SignaturePlacement-Element muss nicht unterstützt werden. Bei anderen Signaturtypen wird das Element ignoriert und eine Warnung (Fehlercode 4197, Parameter SignaturePlacement wurde ignoriert) zurückgeliefert.

	dss: Return Updated Signature	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/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. Die XML-Elemente SignaturePolicyLocation DigestAndAlgorithm werden nicht verwendet. 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: Sign Response	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: Optional Outputs	Enthält (angelehnt an `dss:OptionalOutputs`) optionale Ausgangsparameter: 
	SIG: Document With Signature	 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	
Nachbedingungen	Keine	

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

Tabelle 9: 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 Zugriffsberechtigung“	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 10: 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 zulässigen Zertifikate und Schlüssel sind in TAB_KON_900 aufgelistet.
[<=]

--------- (bis hierher in gemSpec_KON eingebaut: 30.07.2020) ----------

Vorgemerkt: ggf. auch Änderung in gemILF_PS: Kapitel 4.4.1: Erstellen digitaler Signaturen

1.1.2 Erstellen digitaler Signaturen

.....

<PTV4> Nach der Einführung von elliptischen Kurven auf TI-Signaturkarten der Generation G2.1 ist es möglich, mittels des optionalen Parameters Crypt auszuwählen, ob mit ECC- oder RSA-Zertifikaten signiert wird.

Tabelle 11 Tab_ILF_PS_Steuerung_Signaturalgorithmus

Parameter Crypt	Signaturkarte Objektsystemversion < 4.4.0 oder HBA-V (Kartengeneration noch nicht G2.1)	Signaturkarte Objektsystemversion >= 4.4.0 (ab Kartengeneration G2.1)
nicht verwendet	RSA-Signatur	ECC-Signatur
"ECC"	keine Signatur, Fehlermeldung	ECC-Signatur
"RSA"	RSA-Signatur	RSA-Signatur
"RSA_ECC"	RSA-Signatur	ECC-Signatur

Sämtliche Konnektoren können mit elliptischen Kurven erstellte Signaturen validieren. Dennoch werden zunächst mit dem PTV4-Konnektor ausschließlich RSA-Signaturen erstellt. Erst wenn die Migration hin zu ECC vollständig ist, werden die Optionen „ECC" und „RSA_ECC" in Tabelle Tab_ILF_PS_Steuerung_Signaturalgorithmus nutzbar sein und das Defaultverhalten hin zu „ECC" geändert.
~~Daher ist sichergestellt, dass innerhalb der TI sämtliche Signaturen validiert werden können, die ohne Angabe des `Crypt` Parameters erstellt wurden, egal welche Signaturkarten zum Einsatz kommen. Dabei kommt das Defaultverhalten des Konnektors (Verhalten "RSA_ECC" bzw. keine `Crypt` Belegung) zum Tragen, bei dem der verwendete Signaturalgorithmus von der Objektsystemversion der Signaturkarte abhängig ist.~~

Bei Bedarf (etwa für Verwendungszwecke der Signatur außerhalb der TI) kann das Default-Verhalten des Konnektors dennoch durch Auswahl von RSA übersteuert werden, so dass der Konnektor unabhängig von der Signaturkarte auf eine Verwendung von RSA festgelegt wird.

</PTV4>