1 Änderung in gemSpec_Basis_KTR_Consumer

1.1 Vereinheitlichung zu "Clientsystemschnittstelle"

Jedes Vorkommen des Wortes "~~Clientschnittstelle~~", "~~Client-Schnittstelle~~" sowie des Wortes "~~Außenschnittstelle~~" im Dokument wird mit dem Wort "Clientsystemschnittstelle" ersetzt.

Für Freitext-Sektionen betrifft dies:

- Kapitel 6.1.2 im Titel
- Kapitel 6.2.2 im Titel
- Kapitel 6.3.2 im Titel
- Kapitel 6.4.2 im Titel
- Kapitel 5.1.3.3 im Titel
- Kapitel 4.3 im Fließtext:
  - ~~Client-Schnittstelle~~Clientsystemschnittstelle des Moduls

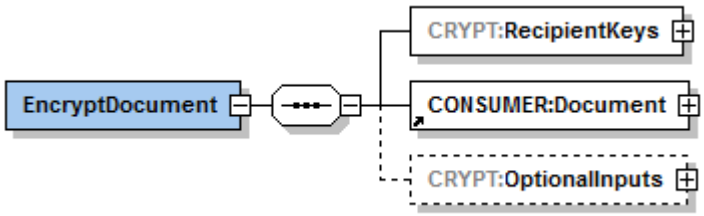
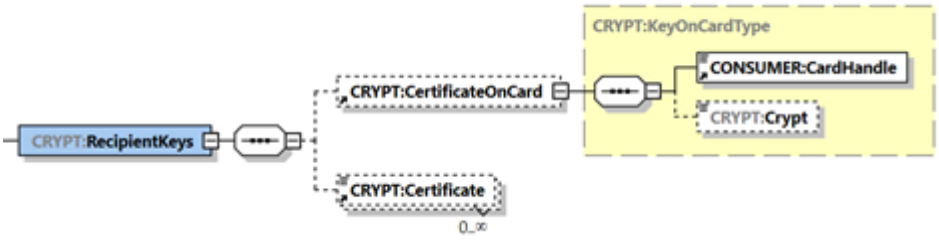
Neben Freitext-Sektionen betrifft dies auch auch die folgenden normativen Anforderungen, welche entsprechend ersetzt werden:

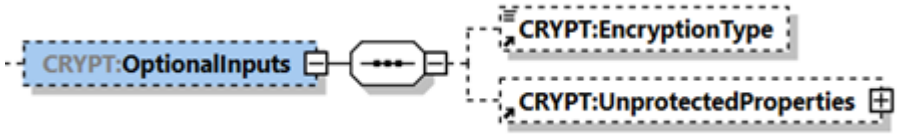
Anforderung A_17510-03 wird mit A_17510-04 ersetzt:

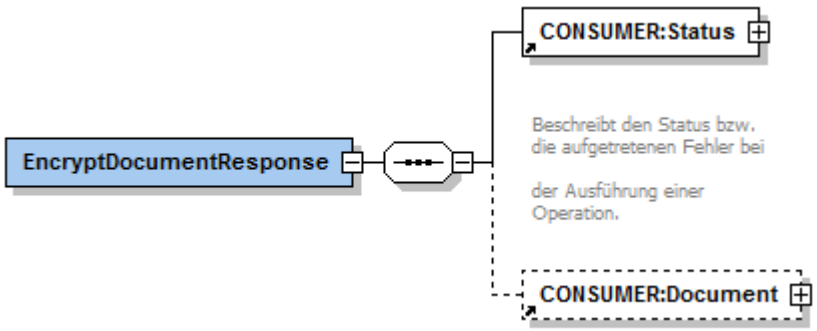
A_17510-04 - Basis- und KTR-Consumer, Operation EncryptDocument
Der Verschlüsselungsdienst des Basis- und KTR-Consumer MUSS an der Clientsystemschnittstelle eine Operation EncryptDocument anbieten.

Tabelle 1: Tab_Operation_EncryptDocument

Name	`EncryptDocument`
Beschreibung	Diese Operation verschlüsselt ein übergebenes Dokument hybrid. Der Dokumententyp XML wird gesondert behandelt. Alle anderen Dokumententypen nutzen die binäre Verschlüsselung Für die hybride Verschlüsselung wird ein asymmetrischer Schlüssel aus einem X.509v3-Zertifikat genutzt. Dieses Zertifikat wird als Parameter übergeben oder auf dem HSM referenziert. Pro Operationsaufruf können mehrere Hybridschlüssel erzeugt werden. Durch das Zertifikat wird festgelegt, ob RSA oder ECC basierte Hybridschlüssel erzeugt werden. Bei Angabe der Zertifikate über

	`CertificateOnCard` (Referenz auf HSM) wird das Verschüsselungsverfahren durch die Angabe in `Crypt` bestimmt. Es können Hybridschlüssel für RSA oder ECC oder beide Verfahren erzeugt werden. Für alle Dokumententypen wird immer das gesamte Dokument verschlüsselt.
Aufrufparameter	
	
	`RecipientKeys` — Identifiziert die Empfänger der zu verschlüsselnden Nachricht über X.509-Zertifikate (öffentliche Schlüssel). Quelle für die Zertifikate kann eine Karte sein, die per CertificateOnCard-Element referenziert wird, oder der Aufrufer, der X.509-Zertifikate im Certificate-Element übergibt.
	`CardHandle` — Identifiziert die zu verwendende Karte mit dem (öffentlichen) Schlüssel. Ist das Element nicht vorhanden, so werden nur Zertifikate per Element `Certificate` übergeben.
	`Crypt` — Der Wert dieses Parameters ist in Tabelle Tab_KeyReference_für_Encrypt/Decrypt spezifiziert und gibt den Typ von Zertifikaten und dadurch das Verfahren für die Erzeugung der Hybridschlüssel vor. (Default-Wert ist RSA)
	`Certificate` — `Certificate` ist ein Base64-kodiertes XML-Element, in dem das Zertifikat, das den asymmetrischen Schlüssel enthält (öffentlicher Schlüssel), DER-kodiert übergeben wird. Es kann eine Liste von Zertifikaten übergeben werden. Dieses Element kann leer sein, wenn ausschließlich

		Zertifikate verwendet werden sollen, die über `CertificateOnCard` angegeben werden.
	`CONSUMER:Document`	Dieses entsprechend [OASIS-DSS] Section 2.4.2 spezifizierte Element enthält das zu verschlüsselnde Dokument, wobei das Kindelement `dss:Base64Data` oder `CONSUMER:Base64XML` verwendet wird. Das zugeordnete Verschlüsselungsverfahren ist • XMLEnc: „http://www.w3.org/TR/xmlenc-core/" für `CONSUMER:Base64XML` • CMS: „urn:ietf:rfc:5652" für `dss:Base64Data`
		
	`CRYPT:OptionalInputs`	Enthält die optionalen Parameter `CRYPT:UnprotectedProperties` und `CRYPT:EncryptionType`.
	`EncryptionType`	Dieses optionale Element bestimmt das Verschlüsselungsverfahren. Es MUSS das Verfahren XMLEnc: „http://www.w3.org/TR/xmlenc-core/" unterstützt werden, wenn das Dokument in `CONSUMER:Base64XML` übergeben wird und CMS: „urn:ietf:rfc:5652", wenn das Dokument in `dss:Base64Data` übergeben wird. Die Verwendung dieses Elements ist aufgrund der impliziten Zuordnung der Verschlüsselungsverfahren zur Methode der Dokumentübergabe nicht erforderlich.
	`CRYPT:UnprotectedProperties`	Dieses optionale Element wird nur für das Verschlüsselungsverfahren CMS ausgewertet (zu verschlüsselndes Dokument ist in `dss:Base64Data` vorhanden). Die Elemente `./UnprotectedProperties/Property/Value/CMSAttribute` müssen base64/DER-kodiert ein vollständiges ASN.1-`Attribute` enthalten, definiert in [CMS# 9.1.AuthenticatedData Type]. Es muss bei der Erstellung des CMS-Containers unter `"unauthAttrs"` aufgenommen werden. Das zugehörige Element

		./UnprotectedProperties/Property/Identifier wird nicht ausgewertet.
Rückgabe		
	Status	Enthält den Ausführungsstatus der Operation.
	Document	Enthält das verschlüsselte Dokument in Base64-codierter Form, wenn die Verschlüsselung erfolgreich durchgeführt wurde. Im Fall XMLEnc wird das verschlüsselte XML-Dokument in CONSUMER:Document/CONSUMER:Base64XML zurückgegeben. Im Fall CMS wird das verschlüsselte Dokument in CONSUMER:Document/dss:Base64data zurückgegeben.
Vorbedingungen	Keine	
Nachbedingungen	Keine	

Vor der Verwendung für die Verschlüsselung MÜSSEN Zertifikate durch den Aufruf von PL_TUC_PKI_VERIFY_CERTIFICATE auf ihre Gültigkeit geprüft werden.
Abgelaufene oder gesperrte Zertifikate MÜSSEN von der Verwendung ausgeschlossen werden.

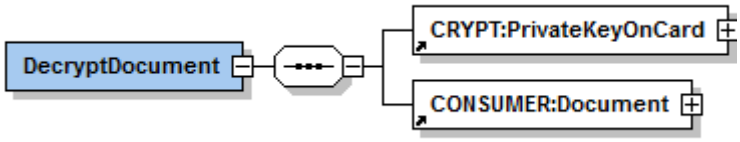
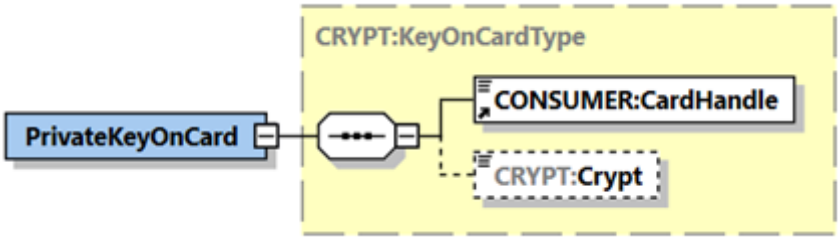
Das Verschlüsseln erfolgt durch Aufruf von PL_TUC_HYBRID_ENCIPHER {
Doc, das zu verschlüsselnde Dokument = CONSUMER:Document;
{Cert(i)}, „Menge der Empfänger-/Ziel-Zertifikate" = RecipientKeys;
Attribute, optionale, zusätzliche Attribute = UnprotectedProperties;
}
Wird ein Zertifikat per CertificateOnCard-Element referenziert, ist dieses vorher durch den HSMProxy zu extrahieren

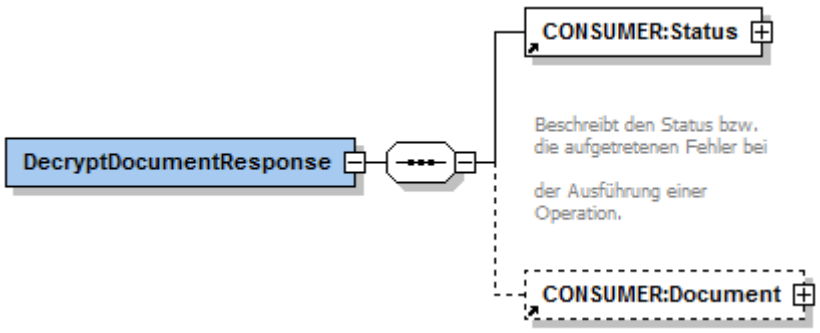
[<=]

Anforderung A_17515-02 wird mit A_17515-03 ersetzt:

A_17515-03 - Basis- und KTR-Consumer, Operation DecryptDocument
Der Verschlüsselungsdienst des Basis- und KTR-Consumer MUSS an der Clientsystemschnittstelle eine Operation `DecryptDocument` anbieten.

Tabelle 2: Tab_Operation_DecryptDocument

Name	`DecryptDocument`	
Beschreibung	Diese Operation entschlüsselt ein hybrid verschlüsseltes Dokument. Es werden die Dokumententypen XML und Andere (Binär) unterstützt. Für die Entschlüsselung wird ein asymmetrischer Schlüssel zu einem X.509v3-Zertifikat genutzt. Das Kryptoverfahren (RSA oder ECC) wird durch den Hybridschlüssel des verschlüsselten Dokuments bestimmt. Liegt eine Verschlüsselung sowohl für RSA, als auch ECC vor, erfolgt vorrangig eine Entschlüsselung mittels des ECC-Schlüssels.	
Aufrufparameter		
		
	`PrivateKeyOnCard`	Identifiziert die zu verwendende Karte mit dem (privaten) Schlüssel.
	`CardHandle`	Identifiziert die Karte.
	`Crypt`	Wird nicht verwendet. Die Auswahl des Kryptoverfahrens erfolgt anhand des Hybridschlüssels des verschlüsselten Dokuments..
	`CONSUMER:Document`	Enthält das base64-codierte Dokument, das entschlüsselt werden soll.

Rückgabe		
	`Status`	Enthält den Ausführungsstatus der Operation.
	`Document`	Enthält das entschlüsselte Dokument in Base64-codierter Form. Im Fall der Verschlüsselung mit XMLEnc wird das entschlüsselte XML-Dokument in `CONSUMER:Document/CONSUMER:Base64XML` zurückgegeben. Im Fall der Verschlüsselung mit CMS wird das entschlüsselte Dokument in `CONSUMER:Document/dss:Base64data` zurückgegeben.
Vorbedingungen	Keine	
Nachbedingungen	Keine	

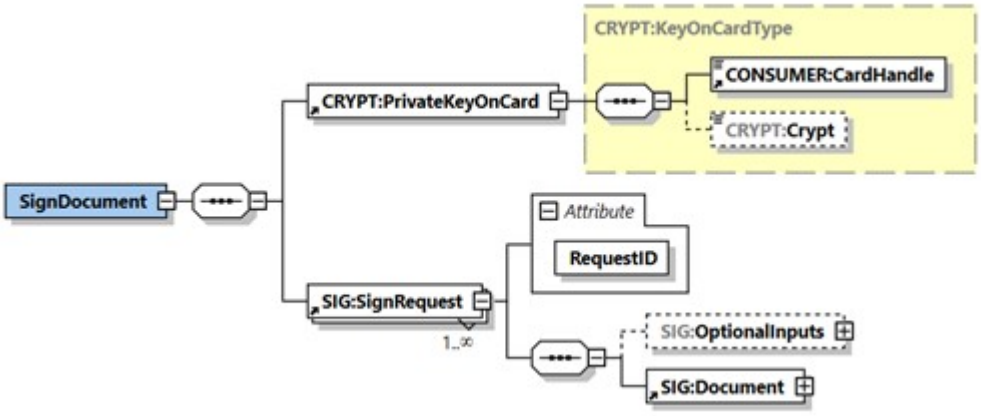
Das Entschlüsseln erfolgt durch Aufruf von PL_TUC_HYBRID_DECIPHER {
  D, "das verschlüsselte Dokument =`CONSUMER:Document;`
  Id, "(Identität des) Empfänger" =`PrivateKeyOnCard;`
}
[<=]

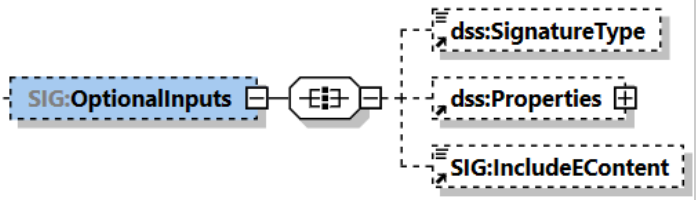
Anforderung A_17525-02 wird mit A_17525-03 ersetzt:

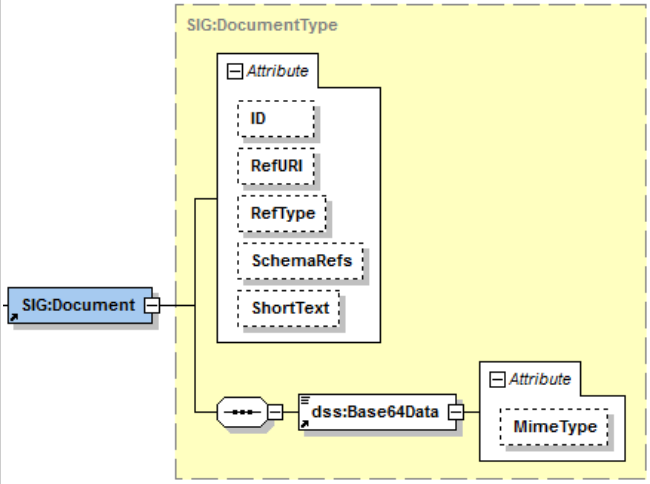
A_17525-03 - Basis- und KTR-Consumer, Operation SignDocument
Der Signaturdienst des Basis- und KTR-Consumer MUSS an der Clientsystemschnittstelle eine an [OASIS-DSS] angelehnte Operation `SignDocument` wie in Tabelle Tab_Operation_SignDocument beschrieben anbieten.

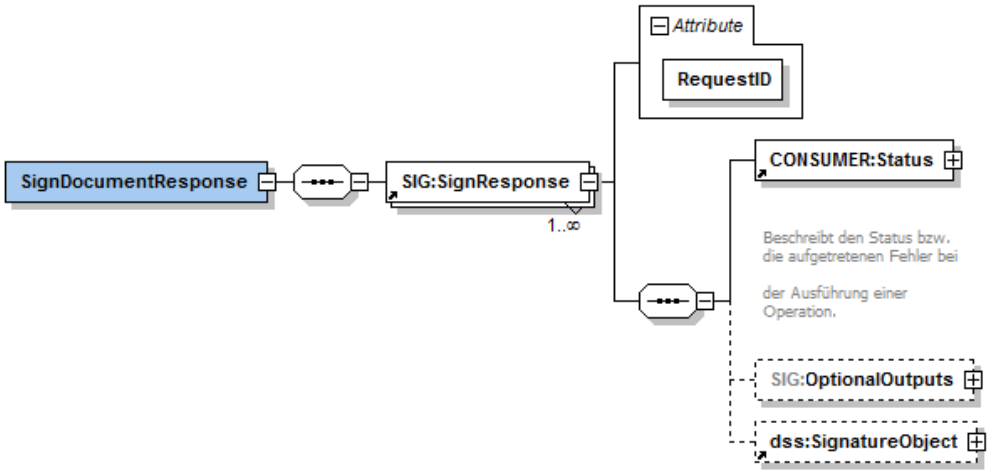
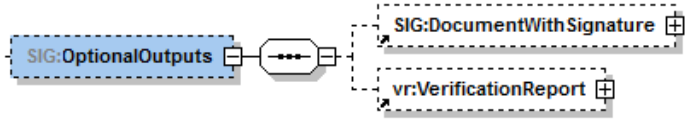
Tabelle 3: Tab_Operation_SignDocument

Name	**`SignDocument`**
Beschreibung	Diese Operation lehnt sich an [OASIS-DSS] an. Sie enthält voneinander unabhängige SignRequests. Jeder SignRequest erzeugt eine Signatur für ein Dokument. Zur Signaturerzeugung werden Schlüssel und Zertifikate eines HSM benutzt. Es wird ausschließlich der Signaturtyp "CMS-Signatur" gemäß

	[RFC 5652] (URIurn:ietf:rfc:5652) und das Profil CAdES-BES gemäß[CAdES] verwendet.
Aufruf-parameter	

`PrivateKeyOnCard`	Identifiziert die zu verwendende Karte mit dem (privaten) Schlüssel.
`CardHandle`	Identifiziert die zu verwendende Signaturkarte.
`Crypt`	Dieser Parameter steuert die Auswahl der Zertifikate und Schlüssel für die Signaturerstellung. Die Werte sind in der Tabelle Tab_Zertifikate_für_Sign/VerifyDocument vorgegeben. (Default-Wert ist RSA)
`SIG:SignRequest`	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:OptionalInputs`	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 in dss:Base64Data.
	dss:SignatureType	Durch dieses in [OASIS-DSS] (Abschnitt 3.5.1) beschriebene Element kann der generelle Typ der zu erzeugenden Signaturen angegeben werden. Es muss der Signaturtyp CMS-Signatur (URI urn:ietf:rfc:5652) unterstützt werden. Fehlt dieses Element, so muss der Signaturtyp CMS-Signatur (URI urn:ietf:rfc:5652) implizit verwendet werden.
	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 . Es dürfen genau die folgenden Attribute ./SignedProperties/Property/Value/CMSAttribute und ./UnsignedProperties/Property/Value/CMSAttribute enthalten sein. 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:IncludeEContent	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.

		Fehlt dieses Element oder ist der Wert = "'false', wird die Signaturvariante "detached" verwendet, ansonsten "enveloping".
Rückgabe		
	`SIG:SignResponse`	Eine `SignResponse` kapselt den ausgeführten Signaturauftrag pro Dokument. Die Zuordnung zwischen `SignRequest` und `SignResponse` erfolgt über die RequestID.
	`CONSUMER:Status`	Enthält den Status der ausgeführten Operation pro SignRequest.
	`SIG:OptionalOutputs`	Enthält optionale Ausgangsparameter. Dieses Element wird durch den Basis- und KTR-Consumer nicht befüllt. 
	`SIG:DocumentWithSignature`	Dieses Element wird durch den Basis- und KTR-Consumer nicht befüllt.
	`vr:VerificationReport`	Dieses Element wird durch den Basis- und KTR-Consumer nicht befüllt.

	dss:SignatureObject	Enthält im Erfolgsfall die erzeugte Signatur in Form eines dss:SignatureObject-Elements gemäß [OASIS-DSS] (Abschnitt 3.2). Der Signaturwert wird im XML-Element dss:SignatureObject/dss:Base64Signature übergeben. Der Signatur-Typ (CMS Signatur) in dss:SignatureObject/dss:Base64Signature/@Type Die XML-Elemente dss:SignatureObject/ds:Signature dss:SignatureObject/dss:Timestamp dss:SignatureObject/dss:SignaturePtr dss:SignatureObject/dss:Other werden nicht verwendet.
Vorbedingungen	Keine	
Nachbedingungen	Keine	

Das Signieren erfolgt durch Aufruf von PL_TUC_SIGN_DOCUMENT_nonQES {
  IDENTIFIKATOR = PrivateKeyOnCard;
  DOKUMENT = SIG:Document;
  DOKUMENTTYPE = dss:SignatureType;
}

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

Tabelle 4: Tab_Zertifikate_für_Sign/VerifyDocument(nonQeS)

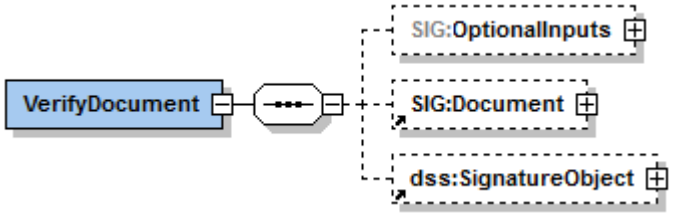
Karte	Crypt (Wert)	**KeyReference (Verify)**	**KeyReference (Sign)**
		in DF.ESIGN	in DF.ESIGN
SM-B (KTR/Org) (HSM)	RSA	EF.C.HCI.OSIG.R2048	PrK.HCI.OSIG.R2048
	ECC	EF.C.HCI.OSIG.E256	PrK.HCI.OSIG.E256
	RSA_ECC	EF.C.HCI.OSIG.R2048 EF.C.HCI.OSIG.E256	PrK.HCI.OSIG.R2048 PrK.HCI.OSIG.E256

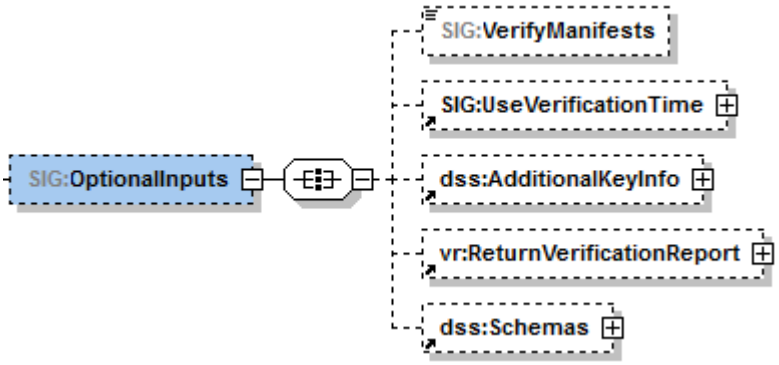
[<=]

Anforderung A_17526-02 wird mit A_17526-03 ersetzt:

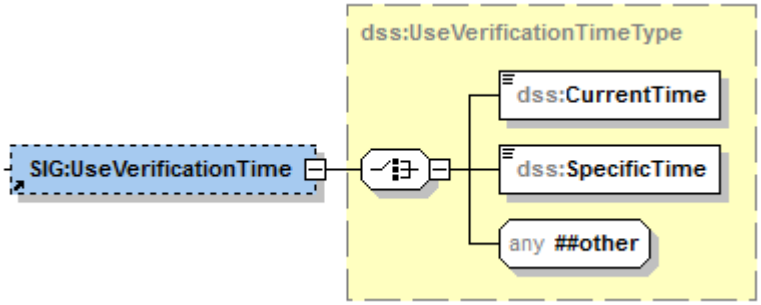
A_17526-03 - Basis- und KTR-Consumer, Operation VerifyDocument
Der Signaturdienst des Basis- und KTR-Consumer MUSS an der Clientsystemschnittstelle eine Operation VerifyDocument wie in Tabelle Tab_Operation_VerifyDocument beschrieben anbieten.

Tabelle 5: Tab_Operation_VerifyDocument

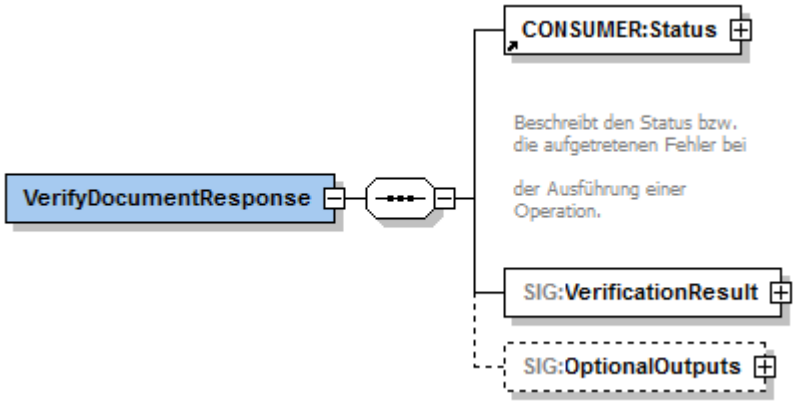
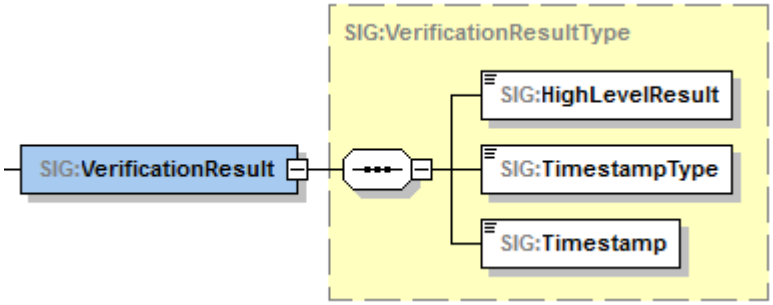
Name	`VerifyDocument`	
Beschreibung	Diese Operation verifiziert die Signatur eines Dokumentes. Der Basis- und KTR-Consumer MUSS jede konform zur Clientsystemschnittstelle SignDocument erzeugte Signatur durch VerifyDocument prüfen können. Das Ergebnis der Prüfung wird, wenn gefordert, in Form eines standardisierten Prüfberichts in einer `VerificationReport`-Struktur gemäß [OASIS-VR] zurückgeliefert.	
Aufruf-parameter		
	`SIG: OptionalInputs`	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: SignatureObject`	Enthält die zu prüfende Signatur, wenn sie nicht im Dokument selbst eingebettet ist ([OASIS-DSS] Kapitel 4.1). Die Signatur wird in `ss:Base64Signature` mit entsprechend gesetztem `Type`-Attribut (siehe `SignatureType`) übergeben, wobei der nachfolgende Werte unterstützt werden muss: • **CMS-Signatur urn:ietf:rfc:5652**



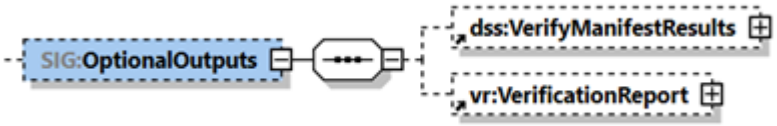
SIG: VerifyManifests	Dieses Element wird durch den Basis-/KTR-Consumer nicht verwendet.



SIG: UseVerification Time	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	Dieses Element wird durch den Basis-/KTR-Consumer nicht verwendet.
vr: Return VerificationReport	Durch dieses in [OASIS-VR] spezifizierte Element kann die Erstellung eines ausführlichen Prüfberichtes angefordert werden.
dss:Schemas	Dieses Element wird durch den Basis-/KTR-Consumer nicht verwendet.

<table>
<tr><td rowspan="4">Rückgabe</td><td colspan="2"> 
</td></tr>
<tr><td>Status</td><td>Enthält den Ausführungsstatus der Operation.</td></tr>
<tr><td>SIG:
Verifi
cation
Result</td><td> 

Das Element <code>Sig:VerificationResult</code> enthält das Ergebnis der Prüfung als Ampel, den Typ des zugehörigen angenommenen Signaturzeitpunkts und der angenommene Signaturzeitpunkt selbst.</td></tr>
<tr><td>SIG:
High
Level
Result</td><td>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</td></tr>
</table>

	SIG: Timestamp Type	Der Typ des angenommenen Signaturzeitpunkts mit folgenden Werten: • SIGNATURE_EMBEDDED_TIMESTAMP: in der Signatur eingebetter Zeitpunkt Ermittelter_Signaturzeitpunkt_Eingebettet • SYSTEM_TIMESTAMP: Systemzeit des Consumers 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: Optional Outputs	Enthält (angelehnt an dss:OptionalOutputs, wie in Abschnitt 2.7 von [OASIS-DSS] beschrieben) optionale Ausgangselemente: 
	dss: Verify Manifest Results	Dieses Element wird durch den Basis-/KTR-Consumer nicht verwendet.
	vr: Verification Report	Dieses in [OASIS-VR] spezifizierte Element wird zurückgeliefert, falls das ReturnVerificationReport-Element als Eingabeparameter verwendet wurde.
Vorbe-dingungen	Keine	
Nachbe-dingungen	Keine	

SigningTime ist der zu prüfende Signaturzeitpunkt. Dieser ergibt sich wie folgt:

1. SigningTime = Benutzerdefinierter_Zeitpunkt, wenn SIG:UseVerificationTime Angaben enthält, sonst

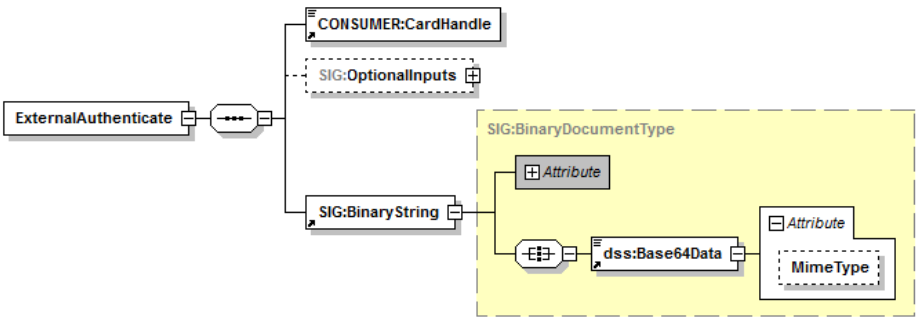
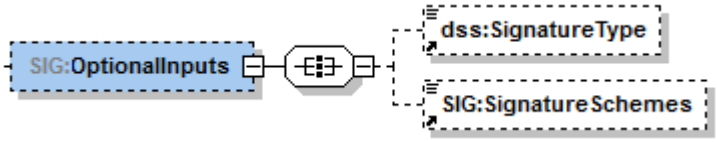
2. `SigningTime` = `Ermittelter_Signaturzeitpunkt_Eingebettet` wenn die Signatur einen Signaturzeitpunkt enthält, sonst
3. `SigningTime` = `Ermittelter_Signaturzeitpunkt_System`, die Systemzeit.

Das Verifizieren erfolgt durch Aufruf von PL_TUC_VERIFY_DOCUMENT_nonQES {
SIGNED_DOCUMENT = `SIG:Document;`
CERTIFICATE = extrahiert aus `SIG:Document;`
SIGNATURE = `dss: SignatureObject ;`
TIME_REFERENCE = `SigningTime;`
}.
[<=]

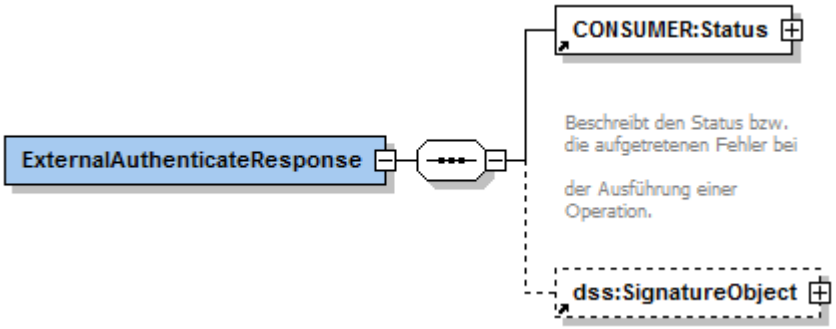
Anforderung A_17578-01 wird mit A_17578-02 ersetzt:

A_17578-02 - Basis- und KTR-Consumer, Operation ExternalAuthenticate
Der Signaturdienst des Basis- und KTR-Consumer MUSS an der Clientsystemschnittstelle die Operation `ExternalAuthenticate` wie in Tabelle Tab_Operation_ExternalAuthenticate beschrieben anbieten.

Tabelle 6: Tab_Operation_ExternalAuthenticate

<table>
<tr><th>Name</th><td colspan="2">ExternalAuthenticate</td></tr>
<tr><th>Beschreibung</th><td colspan="2">Diese Operation versieht einen Binärstring der maximalen Länge 512 Bit mit einer nicht-qualifizierten elektronischen Signatur (nonQES).
Dazu wird das Signaturverfahren PKCS#1 oder ECDSA verwendet.</td></tr>
<tr><th>Aufrufparameter</th><td colspan="2"> 
</td></tr>
<tr><td></td><th>Name</th><th>Beschreibung</th></tr>
<tr><td></td><td>CONSUMER:
CardHandle</td><td>Identifiziert die zu verwendende Signaturkarte.</td></tr>
<tr><td></td><td>SIG:
Optional
Inputs</td><td>Enthält optionale Eingangsparameter:
 
</td></tr>
</table>

	SIG: BinaryString	Dieses Element enthält im Kindelement dss:Base64Data den zu signierenden Binärstring. Das XML Attribut SIG:BinaryString/dss:Base64Data/@MimeType MUSS den Wert "application/octet-stream" haben. Die maximale Länge des Binärstrings beträgt 512 Bit entsprechend der maximal zu erwartenden Hash-Größe. Aus der Länge des Binärstrings wird auf das verwendete Hashverfahren geschlossen. Es werden folgende Längen unterstützt: • 256 Bit: SHA-256 (OID 2.16.840.1.101.3.4.2.1) • 384 Bit: SHA-384 (OID 2.16.840.1.101.3.4.2.2) • 512 Bit: SHA-512 (OID 2.16.840.1.101.3.4.2.3) Im Falle des Signaturverfahrens RSASSA-PKCS1-v1_5 werden SHA-256, SHA-384 und SHA-512 unterstützt. Im Falle des Signaturverfahrens RSASSA-PSS wird SHA-256 unterstützt. Im Falle des Signaturverfahrens ECDSA wird SHA-256 unterstützt. Für die Signaturerstellung gilt: • Im Falle des Signaturverfahrens RSASSA-PKCS1-v1_5 wird die Ausführung der Methode EMSA-PKCS1-v1_5-ENCODE nach [RFC3447], Abschnitt 9.2, mit Schritt 2, Erstellung des DigestInfo-Datenfeldes begonnen. • Im Falle des Signaturverfahrens RSASSA-PSS wird die Ausführung der Methode EMSA-PSS-ENCODE nach [RFC3447], Abschnitt 9.1.1, mit Schritt 3 begonnen. • Im Falle des Signaturverfahrens ECDSA erfolgt die Signaturerstellung gemäß [BSI-TR-03111]#4.2.1. Als Eingangsparameter wird der Hash vom Aufrufer in SIG: BinaryString übergeben.
	dss: SignatureType	Durch dieses in [OASIS-DSS] (Abschnitt 3.5.1) beschriebene Element wird der Typ der zu erzeugenden Signatur bestimmt. Als Signaturtyp wird unterstützt : • **PKCS#1-Signatur** Durch Übergabe der URI urn:ietf:rfc:3447 wird eine PKCS#1 (Version 2.1) Signatur gemäß [RFC3447] erzeugt, die als dss:Base64Signature mit der oben genannten URI zurückgeliefert wird. • **ECDSA-Signatur** Durch Übergabe der URI urn:bsi:tr:03111:ecdsa wird eine ECDSA Signatur gemäß [BSI-TR-03111]#4.2.1 erzeugt, die als

		`dss:Base64Signature` mit der oben genannten URI zurückgeliefert wird. Andere `SignatureType`-Angaben führen zu einer Fehlermeldung. Fehlt dieses Element, so wird ebenfalls der Signaturtyp PKCS#1-Signatur verwendet.
	`SIG: Signature Schemes`	Durch dieses Element wird für PKCS#1-Signaturen zwischen den folgenden SignatureScheme-Optionen unterschieden: • RSASSA-PSS • RSASSA-PKCS1-v1_5 Fehlt dieses Element, so wird als Default-SignatureScheme RSASSA-PSS gewählt.
Rückgabe		
	`CONSUMER: Status`	Enthält den Status der ausgeführten Operation.

	`dss: Signature Object`	Enthält im Erfolgsfall die erzeugte Signatur in Form eines `dss:SignatureObject`-Elements gemäß [OASIS-DSS] (Abschnitt 3.2).
		Der Signaturwert wird im XML-Element `dss:SignatureObject/dss:Base64Signature` übergeben. Das XML-Attribut `dss:SignatureObject/dss:Base64Signature/@Type` kennzeichnet durch den Wert:
		• urn:ietf:rfc:3447 den Signatur-Typ PKCS#1
		bzw.
		• urn:bsi:tr:03111:ecdsa den Signatur-Typ ECDSA.
		Die XML-Elemente
		`dss:SignatureObject/ds:Signature`
		`dss:SignatureObject/dss:Timestamp`
		`dss:SignatureObject/dss:SignaturePtr`
		`dss:SignatureObject/dss:Other`
		werden nicht verwendet.

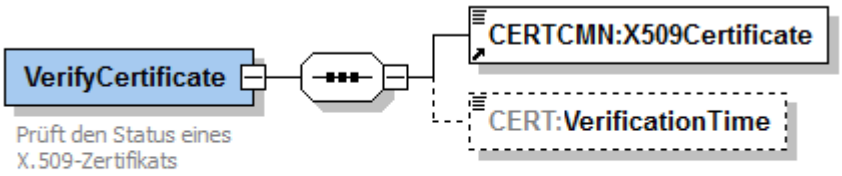
Vorbedingungen	Keine
Nachbedingungen	Keine

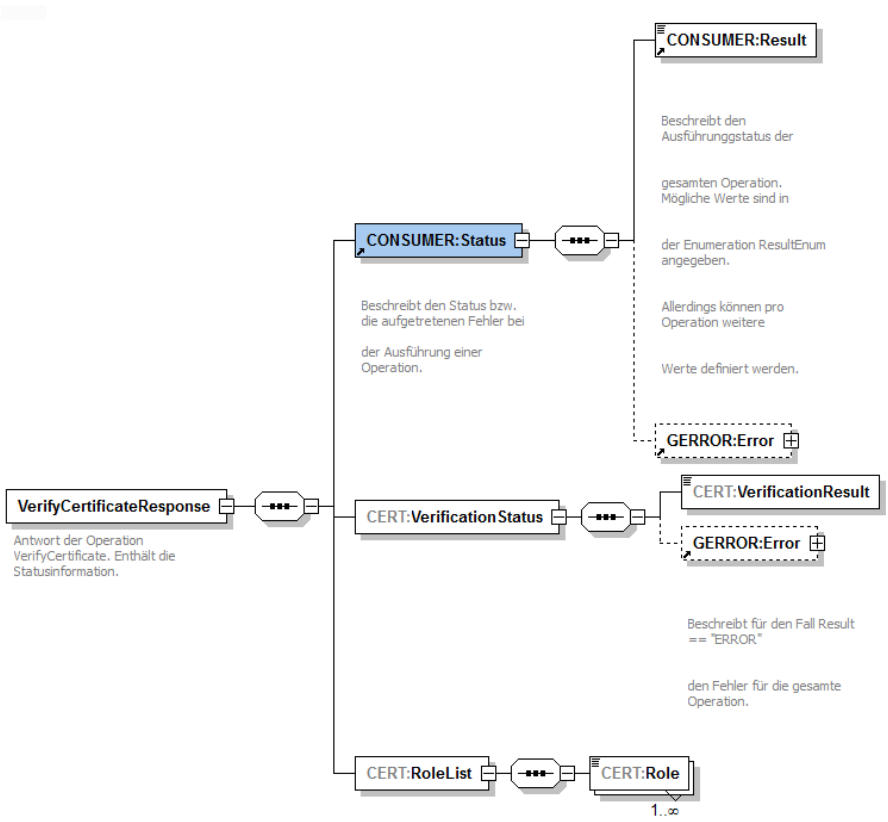
Das Signieren erfolgt durch Aufruf von PL_TUC_SIGN_HASH_nonQES {
  IDENTIFIKATOR = `CardHandle`;
  SIGNATURVERFAHREN = `SIG:SignatureSchemes`;
  HASHWERT = `SIG:BinaryString`;
}
[<=]

Anforderung A_17429-01 wird mit A_17429-02 ersetzt:

A_17429-02 - Basis- und KTR-Consumer, Operation VerifyCertificate
Der Zertifikatsdienst des Basis- und KTR-Consumer MUSS an der Clientsystemschnittstelle eine Operation `VerifyCertificate` wie in Tabelle Tab_Operation_VerifyCertificate beschrieben anbieten.

Tabelle 7: Tab_Operation_VerifyCertificate

Name	**`VerifyCertificate`**	
Beschreibung	Prüft den Status eines Zertifikats.	
Aufrufparameter		
	Name	**Beschreibung**
	`CERTCMN: X509Certificate`	Enthält das base64-codierte Zertifikat, dessen Binärstruktur wiederum ASN.1-codiert (gemäß [gemSpec_PKI]) vorliegt.
	`CERT: VerificationTime`	Der für die Prüfung zu verwendende Referenzzeitpunkt. Falls der Parameter nicht angegeben ist, wird als Referenzzeitpunkt die Systemzeit verwendet.

<table>
<tr><td rowspan="4">Rückgabe</td><td colspan="2"> 
</td></tr>
<tr><td>Status</td><td>Enthält den Ausführungsstatus der Operation.</td></tr>
<tr><td>CERT:VerificationStatus</td><td>Enthält eines der drei möglichen Prüfungsergebnisse in CERT:VerificationResult
VALID
INCONCLUSIVE
INVALID
sowie weiter Details zu den Zuständen „INCONCLUSIVE“ und „INVALID“ in GERROR:Error.</td></tr>
<tr><td>CERT:RoleList</td><td>OIDs der im Zertifikat gespeicherten Rollen.</td></tr>
<tr><td>Vorbe-dingungen</td><td colspan="2">Keine</td></tr>
<tr><td>Nachbe-dingungen</td><td colspan="2">Keine</td></tr>
</table>

Der Ablauf der Operation `VerifyCertificate` ist in Tabelle Tab_Ablauf_VerifyCertificate beschrieben:

Tabelle 8: Tab_Ablauf_VerifyCertificate

Nr.	Aufruf Technischer Use Case oder Interne Operation	Beschreibung

1.	PL_TUC_PKI_ VERIFY_CERTIFICATE	Die Zertifikatsprüfung erfolgt durch Aufruf von PL_TUC_PKI_VERIFY_CERTIFICATE {
		Zu prüfendes Zertifikat = `CERTCMN:X509Certificate;`
		Referenzzeitpunkt = `CERT:VerificationTime;`
		PolicyList = keine Einschränkung;
		KeyUsage = empty;
		ExtendedKeyUsage = empty;
		OCSP-Graceperiod = empty;
		Offline-Modus = nein;
		OCSP-Response = empty ;
		Timeout = empty;
		TOLERATE_OCSP_FAILURE = ja;
		}

2.		Wenn der Prüfprozess fehlerhaft war und nicht zu einem Ergebnis im Sinne eines VerificationResult führt, wird eine FaultMessage erzeugt. War der Prüfprozess erfolgreich, wird eine VerifyCertificateResponse mit • CONSUMER:Status/CONSUMER:Result=OK, • dem VerificationStatus (als Ergebnis der Zertifikatsprüfung) und • den ermittelten Rollen-OIDs erzeugt. Ein Prüfergebnis „INCONCLUSIVE" bzw. „INVALID" wird in CERT:VerificationStatus/GERROR:Error mit den zugehörigen Fehlermeldungen detailliert (in diesem Fall kann CONSUMER:Status/CONSUMER:Result=OK oder CONSUMER:Status/CONSUMER:Result=Warning gesetzt sein).

Tabelle 9: Tab_Übersicht_VerificationResult_VerifyCertificate

CERT:VerificationResult	Bedeutung
VALID	Wenn Gültigkeit zu Referenzzeitpunkt: "gültig"

	Mathematische Gültigkeit:"gültig" OCSP-Prüfung: Online gültig
INVALID	Wenn mindestens ein Wert von (Gültigkeit zu Referenzzeitpunkt, Mathematische Gültigkeit, OCSP-Prüfung) „ungültig“„Prüffehler“ oder „gesperrt" ist.
INCONCLUSIVE	Wenn OCSP-Prüfung „unbekannt“ und die andere Werte „gültig“ sind.

[<=]

Anforderung A_17341 wird mit A_17341-01 ersetzt:

A_17341-01 - Basis- und KTR-Consumer, LDAPv3-Operationen an der Clientsystemschnittstelle
Der Basis- und KTR-Consumer MUSS an der Clientsystemschnittstelle die folgenden LDAPv3-Operationen für den Zugriff auf den Verzeichnisdienst der TI gemäß [RFC4511] anbieten.

- Bind Operation
- Unbind Operation
- Search Operation
- Abandon Operation

Andere LDAPv3-Operationen MÜSSEN mit dem LDAP-Fehler unwillingToPerform (53) beantwortet werden.
Fehler MÜSSEN gemäß [RFC4511]#Appendix A behandelt werden.**[<=]**

In Kapitel 5.1.1.2 die Anforderung A_17474 mit A_17474-01 ersetzt:

A_17474-01 - Anzeige IP-Routinginformationen
Der Basis- und KTR-Consumer MUSS über eine ~~die Managements~~Schnittstelle die konfigurierten IP-Routen und die aktuelle IP-Routingtabelle mit mindestens folgenden Informationen anzeigen:

- Forwarding Status
- Zieladresse/Präfix
- Gateway (Next-Hop)
- Routing Typ
- Routing Preference.

[<=]

In Kapitel 5.1.3.4 die Anforderung A_17513 mit A_17513-01 ersetzt:

A_17513-01 - Konfigurationsparameter Namensdienst und Dienstlokalisierung
Der Administrator des Basis- und KTR-Consumer MUSS die aufgelisteten Parameter in Tabelle 5 über eine ~~die Managements~~Schnittstelle konfigurieren und die aufgelisteten Parameter in Tabelle 6 ausschließlich einsehen können.
Nach jeder Änderung MUSS sichergestellt werden, dass die Änderungen sofort am autoritativen bzw. am Caching Nameserver zur Verfügung stehen.

Tabelle 10: Konfigurationsparameter Namensdienst

ReferenzID	Belegung	Bedeutung und Administrator-Interaktion
DNS_SERVERS_ CONSUMER	Liste von IP-Adressen der DNS-Server	Liste von DNS-Servern, die zur Namensauflösung von Namensräumen in der Einsatzumgebung verwendet werden. Der Administrator MUSS die Liste von DNS-Servern, die die DNS_DOMAIN_CONSUMER auflösen, bearbeiten können. Die IP-Adressen der DNS-Server KÖNNEN auf den Adressbereich der ANLW_LAN_IP_ADDRESS eingeschränkt sein.
DNS_DOMAIN_ CONSUMER	DNS Domainname	DNS Domainname, der von einem DNS-Server der Einsatzumgebung aufgelöst wird. Der Name DARF NICHT mit einem „.“ beginnen.

Tabelle 11: Einsehbare Konfigurationsparameter Namensdienst

ReferenzID	Belegung	Bedeutung
DNS_SERVERS_TI	Liste von IP-Adressen der DNS-Server	Liste von DNS-Servern, die zur Namensauflösung des Namensraums der TI verwendet werden
DNS_TOP_LEVEL_ DOMAIN_TI	DNS Domainname	Top Level Domain des Namensraumes TI

[<=]

1.2 Weitere Änderungen zur Vereinheitlichung von Schnittstellenbezeichnungen

In der Spezifikation kommen weiterhin noch andere Sonderbezeichnungen von Schnittstellen vor. Da deren Bedeutung ebenfalls nicht hinreichend definiert sind, werden diese Bezeichnungen auf die generische Bezeichnung "Schnittstelle" geändert.

In Kapitel 6.6.1 wird der Titel wie folgt geändert:

~~Transport~~Schnittstelle für Kartenkommandos

In Kapitel 6.6.1 wird der Freitext wie folgt geändert:

Die folgenden Anforderungen betreffen die gemäß [gemSpec_Systemprozesse_dezTI#ENV_TUC_CARD_APDU_TRANSPORT] zu beschreibende ~~Transport~~Schnittstelle.

In Kapitel 6.6.1 wird A_18097 durch A 18097-01 ersetzt:

A_18097-01 - Schnittstelle für Kartenkommandos
Wenn der Basis-Consumer Smartcards unterstützt, MUSS er eine sichere Schnittstelle für die Übertragung von Smartcard-APDUs gemäß [CT-API] implementieren.
[<=]

In Kapitel 6.6.1 wird A_18100 durch A 18100-01 ersetzt:

A_18100-01 - Ergänzende Standards für Kartenkommandos
Der Basis-Consumer KANN eine Schnittstelle für die Übertragung von SmartCard-APDUs auf Basis des SICCT-Protokolls gemäß [CCID] und unter Verwendung der vom Hersteller des Kartenterminals ggf. bereitgestellten Hardwaretreiber implementieren.
[<=]

In Kapitel 6.6.2 wird der Freitext wie folgt geändert:

Anwendungsfälle zur PIN-Verwaltung, zur Kartenfreischaltung oder weiterer Fachanwendungen können die Eingabe eines PIN- oder PUK-Geheimnisses erfordern. Der Zugriff auf Karten der TI erfolgt über die Systemprozesse PL_TUC_CARD_*. Der Basis-Consumer als Realisierungsumgebung der Systemprozesse muss seinerseits die von der Plattform geforderten Schnittstellen gemäß [gemSpec_Systemprozesse_dezTI#ENV_TUC_CARD_SECRET_INPUT] implementieren, um die Kommunikation der Plattform mit dem Benutzer zu ermöglichen.

Die ~~KommunikationssSchnittstelle für den Transport von Kartenkommandos~~ ist in Kapitel 6.6.1 ~~Transportschnittstelle für Kartenkommandos~~ beschrieben und umfasst das Kartenterminal, Eingabemedium und Hinweistexte an den Benutzer. Diese kann je nach Konfiguration an einem Gerät als Kartenterminal oder auch eine Kombination aus Bildschirmausgabe, Kartenterminal-PIN-Pad und/oder Tastatureingabe erfolgen.

In Kapitel 6.6.2 wird A_18107 durch A 18107-01 ersetzt:

A_18107-01 - Schnittstelle zur Eingabe des PIN/PUK-Geheimnisses
Wenn der Basis-Consumer Smartcards unterstützt, MUSS er eine Operation gemäß [gemSpec_Systemprozesse_dezTI#ENV_TUC_CARD_SECRET_INPUT] zur Eingabe eines PIN/PUK-Geheimnisses und Weiterleitung an eine Smartcard mit folgenden Parametern implementieren:
Eingabeparameter:

- Identifikator
- Aktion
- minLength
- maxLength
- commandApduPart

Rückgabewerte

- responseApdu

[<=]

In Kapitel 6.6.2 wird A_18108 durch A 18108-01 ersetzt:

A_18108-01 - Umsetzung ENV_TUC_CARD_SECRET_INPUT
Wenn der Basis-Consumer Smartcards unterstützt, MUSS er die Abbildung der Eingabeparameter auf die Rückgabewerte der Operation ENV_TUC_SECRET_INPUT derart

umsetzen, dass

- die Eingabeparameter `Identifikator` und `Aktion` für einen Hinweistext an den Benutzer verwendet werden, welche Aktion auf welchem konkreten Kartenobjekt (z.B. Name einer PIN) durchgeführt wird,
- der `commandApduPart` ~~an der Eingabeschnittstelle~~ um das Benutzergeheimnis ergänzt wird,
- der `commandApduPart` über die ~~TransportsS~~chnittstelle für Kartenkommandos an die Karte gesendet wird

und die Antwortnachricht der Karte als `responseApdu` an den Aufrufer zur Auswertung zurückgegeben wird.
[<=]