Inhaltsverzeichnis

1 Änderungsbedarf

In Vorbereitung auf ein Entfernen von RSA-2048 aus der TI wird die Spezifikation des Konnektorsso überarbeitet, dass

- der Konnektor immer ECC-Zertifikate verwendet
- nur bei G2.0-Karten RSA-Zertifikate verwendet werden, aber über eine Warnung oder Betriebszustand der Anwender informiert wird und eine Sichtbarkeit für die gematik hergestellt wird (Betriebsdaten)
- G2.1-Karten ohne Fehler verwendet werden, auch wenn das RSA-Zertifikat nicht erfolgreich geprüft werden kann
- die Umsetzung muss so erfolgen, dass sie ohne Konfigurationsänderung am Konnektor oder Clientsystem als Autoupdate angewendet werden kann.
- die Schnittstellen zu Primärsystemen muss kompatibel zur aktuellen Version sein, so dass Primärsysteme nicht angepasst werden müssen, damit das geänderte Verhalten wirksam wird.
- zu testende Kombinationen müssen durch eigene Anforderungen abgebildet werden, damit über die Prüfung der AFO-Testabdeckung notwendige Testfälle erfasst werden.

2 Änderung

2.1 Anpassung der Spezifikation

2.1.1 Änderungen in gemSpec_Kon

2.1.1.1 Kapitel 3.2.2 - Organisatorische Anforderungen und Sperrprozesse

A_18930 wird durch A_18930-1 ersetzt

A_18930-01 - Unterstützung von gSMC-K Personalisierungsvarianten
Der Konnektor MUSS unterschiedliche gSMC-K-Personalisierungsvarianten sowohl ~~mit, als auch ohne ECC-Zertifikate~~ für ID.NK.VPN, ID.AK.AUT und ID.SAK.AUT unterstützen:

- nur RSA-Zertifikate
- RSA- und ECC-Zertifikate
- nur ECC-Zertifikate

[<=]

2.1.1.2 Kapitel 3.4 - Betriebszustand

TIP1-A_4512-05 wird durch folgende Anforderung ersetzt:

TIP1-A_4512-06 - Ereignis bei Änderung des Betriebszustandes
Der Konnektor MUSS per Ereignisdienst TUC_KON_256 über Änderungen des Betriebszustandes (Tabelle TAB_KON_503 Betriebszustand_Fehlerzustandsliste) informieren.
Der Konnektor muss dazu für jeden Fehlerzustand $EC mit Error Condition $EC.errorcondition mit verändertem Wert $EC.value den technischen Anwendungsfall TUC_KON_256 „Systemereignis absetzen“ mit folgenden Parametern aufrufen:

```
TUC_KON_256 {
   topic = "OPERATIONAL_STATE/$EC.errorcondition";
   eventType = $EC.type;
   severity = $EC.severity;
   parameters = („Value=$EC.value, $EC.parameterlist”)
}
```

Tabelle 1: TAB_KON_503 Betriebszustand_Fehlerzustandsliste

ErrorCondition (siehe Hinweis 1)	Beschreibung	Type	Severity	max. Feststellungs-zeit	Parameterlist (siehe Hinweis 2)
EC_CardTerminal_ Software_Out_Of_ Date ($ctId)	Software auf Kartenterminal($ctId) ist nicht aktuell	Op	Info	1 day	CtID=$ctId; Bedeutung= $EC.description
EC_CardTerminal_ gSMC-KT_Certificate_ Expires_Soon ($ctId)	Das Zertifikat der gSMC-KT im Kartenterminal($ctId) läuft in weniger als 5 Wochen ab	Op	Info	7 days	CtID=$ctId; Bedeutung= $EC.description
EC_Connector_ Software_Out_ Of_Date	I_KSRS_Download::list_ Updates liefert mindestens eine UpdateInformation mit einer UpdateInformation/Firmware/ FWVersion > aktuelle Version der Konnektorsoftware, deren UpdateInformation/Firmware/ FWPriority = „Kritisch“	Op	Info	1 day	Bedeutung= $EC.description
EC_FW_Update_Available	I_KSRS_Download::list_ Updates liefert mindestens eine UpdateInformation mit einer UpdateInformation/Firmware/ FWVersion > aktuelle Version der Konnektor- oder Kartenterminalsoftware	Op	Info	1 day	Bedeutung= $EC.description
EC_FW_Not_ Valid_Status_ Blocked	Konnektor Firmware muss aktualisiert werden. Zugang zur TI momentan nicht erlaubt.	Sec	Fatal	1 day	Bedeutung= $EC.description

ErrorCondition (siehe Hinweis 1)	Beschreibung	Type	Severity	max. Feststellungs-zeit	Parameterlist (siehe Hinweis 2)
EC_Time_Sync_ Not_Successful	der letzte Synchronisationsversuch der Systemzeit war nicht erfolgreich.	Op	Info	1 sec	LastSyncAttempt= $lastSyncAttempt Timestamp; LastSyncSuccess =$lastSyncSuccess Timestamp; Bedeutung= $EC.description
EC_TSL_Update_ Not_Successful	das letzte Update der TSL war nicht erfolgreich.	Op	Info	1 sec	Bedeutung= $EC.description; LastUpdateTSL= $lastUpdateTSL Timestamp
EC_TSL_Expiring	Systemzeit t mit t > NextUpdate-Element der TSL – 7 Tage und t <= NextUpdate-Element der TSL	Sec	Info	1 day	NextUpdateTSL =$NextUpdate-Element der TSL; Bedeutung= $EC.description
EC_BNetzA_VL_ Update_ Not_Successful	Das letzte Update der BNetzA-VL war nicht erfolgreich	Op	Info	1 sec	LastUpdateBNetz AVL= $lastUpdateBNetz AVLTimestamp; Bedeutung= $EC.description
EC_ BNetzA_VL_ not_valid	Systemzeit t mit t > NextUpdate-Element der BNetzA-VL	Sec	War ning	1 day	NextUpdateBNetz AVL =$NextUpdate-Element der BNetzA-VL; Bedeutung= $EC.description

ErrorCondition (siehe Hinweis 1)	Beschreibung	Type	Severity	max. Feststellungs-zeit	Parameterlist (siehe Hinweis 2)
EC_TSL_Trust_ Anchor_Expiring	Gültigkeit des Vertrauensankers ist noch nicht abgelaufen, läuft aber innerhalb von 30 Tagen ab.	Sec	Info	1 day	ExpiringDateTrust Anchor= Ablaufdatum der Vertrauensanker gültigkeit; Bedeutung= $EC.description
EC_LOG_ OVERFLOW	Wenn im Rahmen der Regeln für die rollierende Speicherung von Logging-Einträgen Einträge gelöscht werden, die nicht älter als SECURITY_LOG_DAYS, LOG_DAYS bzw. FM_ <fmName>_LOG_DAYS sind, tritt der Fehlerzustand ein. Der Fehlerzustand kann nur durch einen Administrator wieder zurückgesetzt werden. Unter Protokoll wird die Liste der auslösenden Protokolle angegeben.	Op	Warning	1 sec	Protokoll=$Protokoll; Bedeutung= $EC.description
EC_CRL_Expiring	Systemzeit t > NextUpdate der CRL – 3 Tage	Sec	Warning	1 day	ExpiringDateCRL= NextUpdate der CRL; Bedeutung= $EC.description

ErrorCondition (siehe Hinweis 1)	Beschreibung	Type	Severity	max. Feststellungs-zeit	Parameterlist (siehe Hinweis 2)
EC_Time_Sync_ Pending_Warning	MGM_LU_ONLINE=Enabled und keine erfolgreiche Synchronisation der Systemzeit seit d Tagen und d > NTP_WARN_PERIOD und d <= NTP_GRACE_PERIOD. Nach einer Korrektur oder Bestätigung der Systemzeit durch einen Administrator muss der Konnektor wie nach einer erfolgreichen Zeitsynchronisation verfahren, d.h., der Tagezähler wird auf 0 zurückgesetzt.	Sec	Warning	1 day	LastSyncSuccess= $lastSyncSuccess Timestamp; Bedeutung= $EC.description
EC_TSL_Out_ Of_Date_Within_ Grace_Period	Systemzeit t mit t > NextUpdate-Element der TSL und t <= NextUpdate-Element der TSL + CERT_TSL_ DEFAULT_GRACE_ PERIOD_DAYS und eine neue TSL ist nicht verfügbar	Sec	Warning	1 day	NextUpdateTSL =$NextUpdate-Element der TSL; GracePeriodTSL =CERT_TSL_ DEFAULT_ GRACE_PERIOD_ DAYS; Bedeutung= $EC.description
EC_CardTerminal_ Not_Available ($ctId)	Kartenterminal($ctId) ist nicht verfügbar. Dieser Betriebszustand bezieht sich auf die als „aktiv" gekennzeichneten KTs.	Op	Error	1 sec	CtID=$ctId; Bedeutung= $EC.description

ErrorCondition (siehe Hinweis 1)	Beschreibung	Type	Severity	max. Feststellungs-zeit	Parameterlist (siehe Hinweis 2)
EC_No_VPN_TI_ Connection	Kein sicherer Kanal (VPN) in die Telematikinfrastruktur aufgebaut. Der Wert 300 sec ist abgeleitet aus der maximalen Verbindungsaufbauzeit bei einem Standortausfall des VPN-Zugangsdienstes.	Op	Error	300 sec	Bedeutung= $EC.description
EC_No_VPN_ SIS_Connection	Kein sicherer Kanal (VPN) zu den Sicheren Internet Services aufgebaut. Der Wert 300 sec ist abgeleitet aus der maximalen Verbindungsaufbauzeit bei einem Standortausfall des VPN-Zugangsdienstes.	Op	Error	300 sec	Bedeutung= $EC.description
EC_No_Online_ Connection	Konnektor kann Dienste im Transportnetz nicht erreichen.	Op	Error	10 sec	Bedeutung= $EC.description
EC_IP_Adresses_ Not_Available	Die IP-Adressen des Netzkonnektors sind nicht oder falsch gesetzt.	Sec	Error	1 sec	Bedeutung= $EC.description
EC_CRL_Out_Of_ Date	Systemzeit t > Next Update der CRL	Sec	Fatal	1 day	NextUpdateCRL= $NextUpdate der CRL; Bedeutung= $EC.description

ErrorCondition (siehe Hinweis 1)	Beschreibung	Type	Severity	max. Feststellungs-zeit	Parameterlist (siehe Hinweis 2)
EC_Firewall_Not_ Reliable	Firewall-Regeln konnten nicht fehlerfrei generiert werden oder beim Laden der Firewall-Regeln ist ein Fehler aufgetreten.	Sec	Fatal	1 sec	Bedeutung= $EC.description
EC_Random_ Generator_ Not_Reliable	Der Zufallszahlengenerator kann die Anforderungen an die zu erzeugende Entropie nicht erfüllen.	Sec	Fatal	1 sec	Bedeutung= $EC.description
EC_Secure_ KeyStore_ Not_Available	Sicherer Zertifikats- und Schlüsselspeicher des Konnektors (gSMC-K oder Truststore) nicht verfügbar	Sec	Fatal	1 sec	Bedeutung= $EC.description
EC_Security_ Log_ Not_Writable	Das Sicherheitslog kann nicht geschrieben werden.	Op	Fatal	1 sec	Bedeutung= $EC.description
EC_Software_ Integrity_ Check_Failed	Eine oder mehrere konnektorinterne Integritätsprüfungen der aktiven Konnektorbestandteile sind fehlgeschlagen.	Sec	Fatal	1 day	Bedeutung= $EC.description

ErrorCondition (siehe Hinweis 1)	Beschreibung	Type	Severity	max. Feststellungs-zeit	Parameterlist (siehe Hinweis 2)
EC_Time_ Difference_ Intolerable	Abweichung zwischen der lokalen Zeit und der per NTP empfangenen Zeit bei der Zeitsynchronisation größer als NTP_MAX_ TIMEDIFFERENCE. Nach einer Korrektur oder Bestätigung der Systemzeit durch einen Administrator muss der Konnektor den Fehlerzustand zurücksetzen.	Sec	Fatal	1 sec	NtpTimedifference= Zeitabweichung; NtpMaxAllowed Timedifference =NTP_MAX_ TIMEDIFFERENCE; Bedeutung= $EC.description
EC_Time_Sync_ Pending_Critical	MGM_LU_ONLINE= Enabled und keine erfolgreiche Synchronisation der Systemzeit seit d Tagen und d > NTP_GRACE_PERIOD Nach einer Korrektur oder Bestätigung der Systemzeit durch einen Administrator muss der Konnektor wie nach einer erfolgreichen Zeitsynchronisation verfahren, d.h., der Tagezähler wird auf 0 zurückgesetzt.	Sec	Fatal	1 day	LastSyncSuccess =$lastSync SuccessTimestamp; NtpGracePeriod= NTP_GRACE_ PERIOD; Bedeutung= $EC.description

ErrorCondition (siehe Hinweis 1)	Beschreibung	Type	Severity	max. Feststellungs-zeit	Parameterlist (siehe Hinweis 2)
EC_TSL_Trust_Anchor_Out_Of_Date	Gültigkeit des Vertrauensankers ist abgelaufen	Sec	Fatal	1 day	ExpiringDateTrustAnchor= Ablaufdatum der Vertrauensanker gültigkeit; Bedeutung= $EC.description
EC_TSL_Out_Of_Date_Beyond_Grace_Period	Systemzeit t mit t > NextUpdate-Element der TSL + CERT_TSL_DEFAULT_GRACE_ PERIOD_DAYS und eine neue TSL ist nicht verfügbar	Sec	Fatal	1 day	NextUpdateTSL =$NextUpdate-Element der TSL; GracePeriodTSL =CERT_TSL_DEFAULT_GRACE_PERIOD_DAYS; Bedeutung= $EC.description
EC_CRYPTOPERATION_ALARM	Gemäß TIP1-A_4597 wurde ein potentieller Missbrauch einer Kryptooperation erkannt. Nur der Administrator kann die Alarmmeldung zurücksetzen.	Sec	Warning	1 min	Operation= $Operationsname; Count=$Summenwert; Arbeitsplatz= $<Liste operationsaufrufenden workplaceIDs>; Meldung= 'Auffällige Häufung von Operationsaufrufen in den letzten 10 Minuten'
EC_OTHER_ERROR_STATE($no)	Herstellerspezifische Fehlerzustände, die per $no (von 1 aufsteigend nummeriert) identifiziert werden. $Type, $Severity und $ParameterList legt der Hersteller nach Bedarf fest.	$Type	$Severity	<= 1 day	Bedeutung= $EC.description

ErrorCondition (siehe Hinweis 1)	Beschreibung	Type	Severity	max. Feststellungs-zeit	Parameterlist (siehe Hinweis 2)
EC_NK_Certificate_Expiring	Das C.NK.VPN-Zertifikat läuft bald ab. Systemzeit t > (Ablaufdatum von C.NK.VPN – 180 Tage)	Sec	Warning	1 day	Iccsn=$Iccsn; Serial=$Serialnumber; Bedeutung= $EC.description
EC_NK_Certificate_Expired	Das C.NK.VPN-Zertifikat ist abgelaufen. Systemzeit t > Ablaufdatum von C.NK.VPN	Sec	Fatal	1 day	Iccsn=$Iccsn; Serial=$Serialnumber; Bedeutung= $EC.description
EC_TLS_Client_Certificate_Security	Das für die Authentifizierung gegenüber dem Clientsystem konfigurierte Zertifikat hat ein Sicherheitsniveau von weniger als 120bit. Zu verwenden ist ein RSA -Zertifikat mit mindestens 3000 bit Schlüssellänge oder ein ECC Zertifikat.	Sec	Info	1 day	Bedeutung= $EC.description
EC_G2_HBA_USED($pseudonym)	Der HBA mit dem Angegebenen Pseudonym verfügt nicht über Zertifikate mit 120bit-Sicherheitsniveau und muss vor dem 01.01.2026 getauscht werden.	Op	Warning		ExpiryDate=Ablaufdatum der Karte

ErrorCondition (siehe Hinweis 1)	Beschreibung	Type	Severity	max. Feststellungs-zeit	Parameterlist (siehe Hinweis 2)
EC_G2_SMCB_USED($pseudonym)	Die SMC-B mit dem Angegebenen Pseudonym verfügt nicht über Zertifikate mit 120bit-Sicherheitsniveau und muss vor dem 01.01.2026 getauscht werden.	OP	Waning		ExpiryDate=Ablaufdatum der Karte
EC_VPN_Registration_ECC_Pending	Der Konnektor ist nur mit einem ID.NK.VPN-RSA-Zertifikat beim VPN-Zugangsdienst registriert, obwohl ein ECC-Zertifikat vorhanden ist. Eine neu-Registrierung ist notwendig.	Op	Info	1 day	Bedeutung= $EC.description

Erläuterungen zu TAB_KON_503:

Hinweis 1:
Jeder Fehlerzustand wird durch einen eindeutigen ErrorCondition identifiziert. Dieser kann einen Parameter enthalten. Sind etwa die Kartenterminals mit ctId=47 und das mit ctId=93 nicht erreichbar, so lauten die ErrorCondition „EC_CardTerminal_Not_Available(47)“ und „EC_CardTerminal_Not_Available(93)“.

Hinweis 2:
EC.description referenziert den Text, der in der Spalte „Beschreibung“ des Zustandes spezifiziert wurde.
Hinweis 3:
Beim Absetzen und Subskribieren folgender EventTopics gelten zusätzliche Vorgaben:
- EC_CardTerminal_Software_Out_Of_Date ($ctId)
- EC_CardTerminal_gSMC-KT_Certificate_Expires_Soon ($ctId)
- EC_CardTerminal_Not_Available ($ctId)
- EC_OTHER_ERROR_STATE($no)
- EC_G2_HBA_USED($pseudonym)
- EC_G2_SMCB_USED($pseudonym)

Beim Absetzen des Systemereignisses muss die Schreibweise der obigen EventTopics hinsichtlich der Position der Klammer strikt den Vorgaben aus der Tabelle TAB_KON_503 entsprechen.
Beim Subskribieren der Systemereignisse bei obigen EventTopics muss beliebige Schreibweise im Bezug auf Whitespaces vor und nach den Klammern vom Konnektor toleriert werden.
Wenn obige EventTopics ohne Parameter in Klammern subskribiert werden, so muss der Konnektor das Systemereignis an den Client für jede $ctId bzw. $no absetzen.

[<=]

Neue Anforderung (funktionale Eignung: Test):

A_25803 - Zurücksetzen von EC_G2_HBA_USED($pseudonym) und EC_G2_SMCB_USED($pseudonym)
Der Konnektor MUSS einmal täglich alle aktiven Betriebszustände EC_G2_HBA_USED($pseudonym) und EC_G2_SMCB_USER ($pseudonym), deren $ValidForm älter als 7 Tage ist entfernen. **[<=]**

2.1.1.3 Kapitel 3.5 - Fachliche Anbindung der Clientsysteme

In Kapitel 3.5.1 wird Anforderung TIP1-A_4517-02 durch TIP1-A_4517-04 ersetzt. Dabei sind Änderungen aus C_11484(ML-139831 - Präzisierung der konnektorinternen Schlüsse- und Zertifikatsgenerierung für clientbasierte Authentisierung) enthalten:

TIP1-A_4517-04 - Schlüssel und X.509-Zertifikate für die Authentisierung des Clientsystems erzeugen und exportieren sowie X.509-Zertifikate importieren
Der Konnektor MUSS die Erstellung und den Export von Schlüsselpaaren und dazugehörigen X.509-Zertifikaten für Clientsysteme durch den Administrator über das Managementinterface ermöglichen. Hierbei MUSS der Konnektor dem Administrator die Möglichkeit geben, das kryptographische Verfahren gemäß Tabelle TAB_KON_866 auszuwählen. Als Exportformat MUSS PKCS#12 verwendet werden. Die so erstellten Zertifikate werden zu ANCL_CCERT_LIST angefügt.
Zur Sicherung der PKCS#12-Datei MUSS der Konnektor ein intern generiertes starkes Passwort anbieten, jedoch alternativ auch die Vergabe des Passwortes durch den Administrator ermöglichen. Soll vom Administrator ein alternatives Passwort gewählt werden, MUSS der Konnektor dazu Hinweise bzgl. Passwortlänge und Komplexität geben, DARF dahingehend aber NICHT technischen Einschränkungen durchsetzen.
Der Konnektor MUSS dem Administrator ferner den Import von konnektorfremden X.509-Zertifikaten für Clientsysteme über das Managementinterface ermöglichen. Importierte Zertifikate MÜSSEN den Vorgaben von TAB_KON_866 entsprechen. Die so importierten Zertifikate werden zu ANCL_CCERT_LIST angefügt.
[<=]

Hinweis: In Bezug auf die Kodierung von ECC-Schlüsseln vgl. [gemSpec_Krypt#A_23511].

In Kapitel 3.5.1 wird Anforderung A_21760-01 durch A_21760-02 ersetzt:

A_21760-02 - ID.AK.AUT auf gSMC-K für Authentisierung des Konnektors gegenüber Clientsystemen verwenden
Der Konnektor MUSS dem Administrator das Einschalten der Verwendung von ID.AK.AUT auf der gSMC-K für die Authentisierung des Konnektors gegenüber den Clientsystemen über das Managementinterface ermöglichen. Der Konnektor MUSS dabei das ECC-Zertifikat C.AK.AUT2.XXXX verwenden. Wenn kein ECC Zertifikat personalisiert ist muss das C.AK.AUT.RSA2048 verwendet werden. Das ~~ausgewählte~~ Zertifikat MUSS sowohl für die Serveridentität an der SOAP-Schnittstelle als auch für die Clientidentität an der CETP-Schnittstelle verwendet werden. Das vor einem Update des Konnektor verwendete Zertifikat MUSS unabhängig von der Kryptographie weiter verwendet werden.
[<=]

2.1.1.4 Kapitel 3.6 - Clientsystemschnittstelle

In Kapitel 3.6 wird Anforderung A_21811-03 durch A_21811-04 ersetzt:

A_21811-04 - Vorgaben für generierte und importierte Schlüssel und Zertifikate
Der Konnektor MUSS bezüglich selbst generierter und importierter Schlüssel und Zertifikate für die TLS-Authentisierung gegenüber Primärsystemen und für die Authentisierung des Clientsystems sowie für die Absicherung der Managementschnittstelle die kryptographischen Vorgaben aus [gemSpec_Krypt] durchsetzen und die Verfahren gemäß Tabelle TAB_KON_866 unterstützen.

Tabelle 2 : TAB_KON_866 Unterstützte Verfahren für generierte und importierte Schlüssel und Zertifikate

Verfahren	~~Unterstützung~~Neugenerierung bzw. -import	Bereits generiert/importiert und in Benutzung
RSA-2048	DARF NICHT unterstützt werden	soll nicht verwendet werden
RSA-3072	MUSS unterstützt werden	MUSS unterstützt werden
ECC-256 mit NIST-Kurven	MUSS unterstützt werden	MUSS unterstützt werden
ECC-256 mit brainpool-Kurven	~~soll nicht verwendet~~ DARF NICHT unterstützt werden	soll nicht verwendet werden

Die [gemSpec_Krypt] führt RSA-3072 nicht auf, macht jedoch allgemeine Vorgaben für RSA, die analog auf RSA-3072 anzuwenden sind.
RSA-2048 sowie Brainpool-Zertifikate ~~sollen~~dürfen bei der neuen Anlage von Zertifikaten nicht verwendet werden. Eine Weiternutzung von RSA-2048 sowie Brainpool-Zertifikaten nach einem Update oder dem Import eines Konfigurationsbackups ist zulässig.
[<=]

2.1.1.5 Kapitel 4.1.4 - Kartenterminaldienst

TUC_KON_050 wird angepasst, so dass der Konnektor nur cipher suites anbietet, zu denen er Schlüsselmaterial besitzt. Die bevorzugte Aushandlung von ECC wird durch A_17124-03 und A_17775 erreicht.

Außerdem wird nur noch die ICCSN des Serverzertifikats gegen das gespeicherte Serverzertifikat geprüft, um die Nutzung von ECC auch ohne manuelles Pairing möglich zu machen.

TIP1-A_4545-04 - TUC_KON_050 „Beginne Kartenterminalsitzung“
Der Konnektor MUSS den technischen Use Case „Beginne Kartenterminalsitzung” gemäß TUC_KON_050 umsetzen.

Tabelle 3: TAB_KON_039 – TUC_KON_050 „Beginne Kartenterminalsitzung“

Element	Beschreibung
Name	TUC_KON_050 „Beginne Kartenterminalsitzung“
Beschreibung	TUC_KON_050 baut eine TLS-Verbindung vom Konnektor zum Kartenterminal auf und beginnt eine SICCT-Sitzung. Anschließend erfolgt die 2. Authentifizierung des Kartenterminals (Prüfung SharedSecret).
Auslöser	• Neustart des Konnektors • nach dem Setzen eines Kartenterminals auf „aktiv“ • im Rahmen eines erneuten Verbindungsversuchs
Vorbedingungen	ctId ist in CTM_CT_LIST vorhanden
Eingangsdaten	• ctId (zu verbindendes Kartenterminal) • role (Benutzerrolle; gültig sind: „User“ und „Admin“)
Komponenten	Konnektor, Kartenterminal
Ausgangsdaten	keine
Nachbedingungen	• TLS-Kanal und SICCT-Session mit gewünschter Benutzerrolle aufgebaut, wenn CT.CORRELATION >= "gepairt" • TLS-Kanal und SICCT-Session mit leerem Username, Password und Session ID aufgebaut, wenn CT.CORRELATION <= „zugewiesen“ • Steck-Ereignisse für alle im KT befindlichen Karten ausgelöst, wenn CT.CORRELATION>=„gepairt“

Standardablauf	Setze CT = CTM_CT_LIST(ctId)
	1. Wenn CT.IS_PHYSICAL = Nein:
	prüfen, ob role = „User“
	Wenn CT.CONNECTED =
	Ja: TUC endet erfolgreich
	Nein:
	- Verbindung zu HSM in Slot 1 aufbauen
	- weiter mit Schritt 9
	2. Wenn CT.CONNECTED = Ja
	prüfen, ob CT.ACTIVEROLE = role
	Ja: TUC endet erfolgreich
	Nein:
	- Schließen der Cardterminal Session mit dem Kartenterminalkommando SICCT CLOSE CT SESSION,
	- weiter ab Schritt 6 (halten der TLS-Verbindung und nur Wechsel der Session)
	3. Aufbau einer TLS-Verbindung mit dem Kartenterminal unter Verwendung von ID.SAK.AUT.
	Der Konnektor MUSS genau die Ciphersuiten aus gemSpec_Krypt im Client-Hello Anbieten, zu denen er ID.SAK.AUT Schlüsselmaterial zur Verfügung hat (ECC bzw. RSA). Dabei Prüfung des KT-Zertifikats mittels
	TUC_KON_037 {
	certificate= C.SMKT.AUT;
	qualifiedCheck=not_required;
	offlineAllowNoCheck=true;
	policyList= oid_smkt_aut;
	intendedKeyUsage= intendedKeyUsage(C.SMKT.AUT);
	intendedExtendedKeyUsage=id-kp-serverAuth;
	validationMode=NONE }
	4. Wenn CT.CORRELATION <=„zugewiesen“:
	a. Öffne eine Cardterminal Session mit dem Kartenterminalkommando SICCT INIT CT SESSION (siehe [SICCT#5.10]) mit leerem Username, Password und Session ID
	b. Nur Verbindung in niedriger Korrelation, daher setze CT.CONNECTED = Nein, um fachliche Nutzung des KT zu verhindert
	c. beende TUC erfolgreich
	5. Prüfe, ob das Server-Zertifikat~~aus~~ der TLS-Verbindung den zum Kartenterminal gespeicherten Referenzdaten (CT.SMKT_AUT) übereinstimmt.
	a. Stimmt das Zertifikat nicht überein, prüfe, ob die ICCSN aus dem Server-Zertifikat der TLS-Verbindung mit der ICCSN der zum Kartenterminal gespeicherten Referenzdaten (CT.SMKT_AUT) übereinstimmt. Bei gleicher ICCSN speichere das Server-Zertifikat aus der TLS-Verbindung als neues Referenzdatum (CT.SMKT_AUT) und führe ein Wartungspairing mit dem Kartenterminal durch.

	b. Läuft das Zertifikat CT.SMKT_AUT (oder C.SMKT.AUT, sie müssen hier identisch sein) in weniger als 35 Tagen ab, so geht der Konnektor in den Betriebszustand EC_CardTerminal_gSMC-KT_Certificate_Expires_Soon (ctId) über.
	6. Parallelisierung
	a. Generierung eines zufälligen Werts (Challenge) mit mindestens 16 Byte Länge gemäß [gemSpec_Krypt#2.2] (siehe [gemSpec_KT#DO_KT_0004]),
	b. Öffnen einer Cardterminal Session mit dem Kartenterminalkommando SICCT INIT CT SESSION (siehe [SICCT#5.10]) mit
	- ctId als Adressat
	- Wenn role = User
	dann mit leerem Username, Password und Session ID
	Wenn role = „Admin
	dann mit leerer Session ID aber mit CT.ADMIN_USERNAME und CT.ADMIN_PASSWORD
	7. Senden der Challenge mittels Kartenterminalkommando EHEALTH TERMINAL AUTHENTICATE (siehe [gemSpec_KT#3.7.2]) in der Ausprägung VALIDATE mit:
	- Kartenterminal als Empfänger
	- und mit der in Schritt 6a generierten Challenge im Shared Secret Challenge DO
	8. Prüfe Antwort des Kartenterminals, ob sie einen korrekten Hashwert über Challenge und angehängtes `CT.SHARED_SECRET` gemäß [gemSpec_KT#SEQ_KT_0002] Schritt 4-5 enthält
	9. Setze:
	a. CT.ACTIVEROLE = $role
	b. CT.CONNECTED = Ja
	10 . Wenn TLS-Verbindung neu aufgebaut werden musste, rufe TUC_KON_256 {
	topic = „CT/CONNECTED“;
	eventType = „Op“;
	severity = Info;
	parameters = („CtID=$CT.CTID,
	Hostname=$CT.HOSTNAME“) }
	11. Ermittle alle im KT gesteckten Karten und befülle entsprechend
	CT.SLOTS_USED
	Für jeden in CT.SLOTS_USED gelisteten Slot X zur weiteren internen Bearbeitung TUC_KON_256{
	topic = „CT/SLOT_IN_USE“;
	eventType = Op;
	severity = Info;
	parameters = („CtID=$CT.CTID,
	SlotNo=$CT.SLOTS_USED[X]“);
	doLog = false;
	doDisp = false }
	rufen.

Element	Beschreibung
Varianten/ Alternativen	Keine.

Fehlerfälle	Fehler in den folgenden Schritten des Ablaufs führen zu:
	Aufruf von TUC_KON_256 {
	topic = "CT/TLS_ESTABLISHMENT_FAILURE";
	eventType = $ErrorType;
	severity = $Severity;
	parameters = („CtID=$CT.ID, Name=$CT.HOSTNAME,
	Error=$Fehlercode, Bedeutung=$Fehlertext“) }
	Abbruch der Verarbeitung mit den ausgewiesenen Fehlercodes
	
	(→1): Admin-Rolle für logische KTs nicht möglich (hätte bei korrekter
	Implementierung nicht stattfinden dürfen), Fehlercode: 4032
	(→1): Verbindungsaufbau zu HSM fehlgeschlagen, Fehlercode: 4032
	(→3): Fehler im TLS-Verbindungsaufbau bzw. Zertifikatsprüfung, Fehlercode: 4028
	und setze CT.CONNECTED auf „Nein“
	(→3): TLS-Verbindung konnte nicht innerhalb von CTM_TLS_HS_TIMEOUT Sekunden aufgebaut werden , Fehlercode: 4028 und setze CT.CONNECTED auf „Nein“
	(→5a): ICCSN-Vergleich Fehlgeschlagen, Fehlercode:
	4029
	und setze CT.CORRELATION auf „gepairt“
	und setze CT.CONNECTED auf „Nein“
	und terminiere TLS-Verbindung
	(→6b): Hinterlegte KT-Admin-Credentials fehlerhaft, Fehlercode: 4030
	und in die User-Session zurückzuwechseln (damit das KT für den normalen Fachbetrieb weiterhin zur Verfügung steht)
	(→8): Prüfung auf Nachweis SharedSecret fehlgeschlagen, Fehlercode
	4029
	und setze CT.CORRELATION auf „gepairt“
	und setze CT.CONNECTED auf „Nein“
	und terminiere TLS-Verbindung

Nichtfunktionale Anforderungen	keine
Zugehörige Diagramme	Abbildung PIC_KON_110 Aktivitätsdiagramm zu „Beginne Kartenterminalsitzung

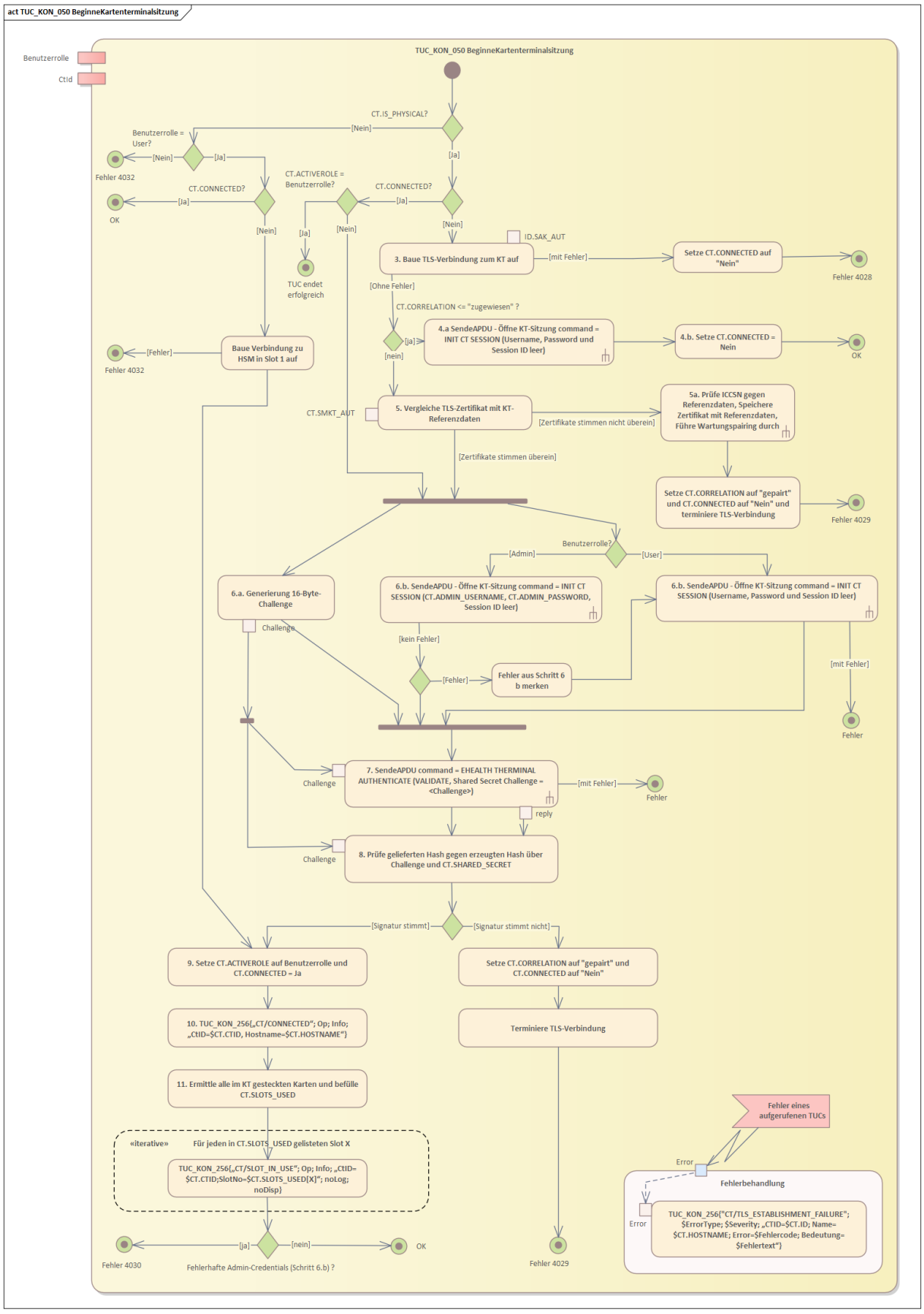


Abbildung 1: PIC_KON_110 Aktivitätsdiagramm zu „Beginne Kartenterminalsitzung"

Tabelle 4: TAB_KON_523 Fehlercodes TUC_KON_050 „Beginne Kartenterminalsitzung“

Fehlercode	ErrorType	Severity	Fehlertext
Neben den Fehlercodes der aufgerufenen technischen Use Cases können folgende weitere Fehlercodes auftreten:			
4028	Technical	Error	Fehler beim Versuch eines Verbindungsaufbaus zu KT
4029	Security	Error	Fehler bei der KT-Authentisierung. KT möglicherweise manipuliert
4030	Security	Error	Admin-Werte für KT fehlerhaft
4032	Technical	Error	Verbindung zu HSM konnte nicht aufgebaut werden

[<=]

TIP1-A_4548-03 - TUC_KON_053 „Paire Kartenterminal“
Der Konnektor MUSS den technischen Use Case „Paire Kartenterminal” gemäß TUC_KON_053 umsetzen.

Tabelle 5: TAB_KON_041 – TUC_KON_053 „Paire Kartenterminal“

Element	Beschreibung
Name	TUC_KON_053 „Paire Kartenterminal“
Beschreibung	TUC_KON_053 führt das Pairing zwischen dem Konnektor und einem eHealth-Kartenterminal durch.
Auslöser	Dialoge zur Administration des Konnektors. Der Administrator hat ein Kartenterminal im Dialog der Managementschnittstelle ausgewählt und das Pairing aufgerufen.
Vorbedingungen	• KT ist in CTM_CT_LIST vorhanden • CT.CORRELATION = „zugewiesen“ • CT.IS_PHYSICAL = Ja
Eingangsdaten	• ctId
Komponenten	Konnektor, Kartenterminal
Ausgangsdaten	• Keine
Nachbedingungen	- CT.CORRELATION = „aktiv“, wenn Pairing erfolgreich - CT.CORRELATION = „zugewiesen“, wenn Pairing nicht erfolgreich - CT.CONNECTED = „Ja“, wenn Pairing erfolgreich

Standardablauf	Setze CT = CTM_CT_LIST(ctId)
	1. Prüfe `CT.VALID_VERSION = true`
	2. Aufbau einer TLS-Verbindung mit dem Kartenterminal unter Verwendung von ID.SAK.AUT.
	Der Konnektor MUSS dabei genau die ciphersuiten aus gemSpec_Krypt im Client-Hello Anbieten, zu denen er ID.SAK.AUT Schlüsselmaterial zur Verfügung hat (RSA bzw. ECC).
	Dabei:
	a. Speichern des präsentierten KT-Zertifikats in CT.SMKT_AUT
	b. Prüfung des KT-Zertifikats mittels TUC_KON_037{
	certificate = C.SMKT.AUT;
	qualifiedCheck = not_required;
	offlineAllowNoCheck = true;
	policyList = oid _smkt_aut;
	intendedKeyUsage= intendedKeyUsage(C.SMKT.AUT)
	;
	intendedExtendedKeyUsage = id-kp-serverAuth;
	validationMode = NONE }
	3. Der Konnektor entnimmt den Fingerprint dem KT-Zertifikat und stellt
	dies dem Administrator im Dialog der Managementschnittstelle dar.
	Der Konnektor fordert den Administrator auf, den Fingerprint zu
	akzeptieren oder zurückzuweisen.
	4. Wenn der Administrator den Fingerprint bestätigt,
	a. generiert der Konnektor einen neuen Schlüssel, das
	Shared Secret ShS.KT.AUT gemäß [gemSpec_Krypt#2.2] (siehe [gemSpec_KT#3.7]) und speichert es in `CT.SHARED_SECRET`
	b. und eröffnet der Konnektor mit dem Kartenterminalkommando SICCT INIT CT SESSION (siehe
	[SICCT#5.10]) mit
	- ctId als Adressat
	- und mit leerem Username, Password und Session ID
	eine Cardterminal Session.
	5. Der Konnektor sendet mittels Kartenterminalkommando EHEALTH
	TERMINAL AUTHENTICATE (siehe [gemSpec_KT#3.7.2]) in der Ausprägung CREATE mit
	- ctId als Empfänger
	- und mit dem in Schritt 4.a generierten Schlüssel im Shared Secret DO und der Display Message
	„`KT:$CT.MAC_ADRESS MIT KON:$MGM_KONN_HOSTNAME PAIREN OK?`“, wobei die MAC-Adresse mit Trenner im folgenden Format dargestellt werden MUSS: „AABBCC:DDEEFF“
	das Shared Secret an das Kartenterminal.
	6. Der Konnektor prüft ob in der Antwort des Kartenterminals eine

Element	Beschreibung
	korrekte Signatur des Shared Secrets gemäß [gemSpec_KT#SEQ_KT_0001] Schritt 7, ausgeführt mit dem Schlüssel zum Zertifikat CT.SMKT_AUT vorliegt. 7. CT.CORRELATION wird auf „gepairt" gesetzt 8. TLS-Verbindung, die zum Pairen diente, beenden und zuvor das Kartenterminalkommando SICCT CLOSE CT SESSION mit ctId als Adressat senden 9. Automatischer Zustandsübergang CT.CORRELATION = „gepairt" nach „aktiv" (implizite Aktion des Administrators durch Aufruf von TUC_KON_053). 10. „Arbeits"-TLS-Verbindung neu aufbauen durch Aufruf TUC_KON_050 { ctId; role = „User"}
Varianten/ Alternativen	(→4): weist der Administrator den Fingerprint in Schritt 3 ab, wird TUC_KON_053 nach Ausführung folgender Aktivitäten beendet: 4.1.a) Löschen von CT.SMKT_AUT 4.1.b) Abbau der TLS-Verbindung, Setzen von CT.CONNECTED auf „Nein"

Fehlerfälle	Fehler in den folgenden Schritten des Ablaufs führen zu:
	a) Aufruf von TUC_KON_256 {
	topic = "CT/ERROR";
	eventType = $ErrorType;
	severity = $Severity;
	parameters = („CtID=$ctId, Name=$CT.HOSTNAME,
	Error=$Fehlercode,
	Bedeutung=$Fehlertext");
	doDisp = false }
	b) Löschen von CT.SMKT_AUT, CT.SHARED_SECRET
	c) Direkte Anzeige der Fehlermeldung für den Administrator
	d) Abbruch der Verarbeitung mit den ausgewiesenen Fehlercodes
	
	(→1) Version des KT wird nicht unterstützt, Fehlercode: 4042
	(→2b) Zertifikat ist zeitlich nicht gültig, Fehlercode: 1021 (CERTIFICATE_NOT_VALID_TIME)
	(→2) Fehler im TLS Verbindungsaufbau bzw. Zertifikatsprüfung, Fehlercode: 4040
	(→4b) Fehler in SICCT INIT CT SESSION, Fehlercode: 4041 mit Angabe des SICCT-Fehlers
	(→5) Fehler in EHEALTH TERMINAL AUTHENTICATE, Fehlercode: 4041 mit Angabe des SICCT-Fehlers
	(→6) Signaturprüfung fehlgeschlagen, Fehlercode: 4041

Zugehörige Diagramme	Siehe PIC_KON_057

Tabelle 6: TAB_KON_113 Fehlercodes TUC_KON_053 „Paire Kartenterminal"

Fehlercode	ErrorType	Severity	Fehlertext
Neben den Fehlercodes der aufgerufenen technischen Use Cases können folgende weitere Fehlercodes auftreten:			
4040	Security	Error	Fehler beim Versuch eines Verbindungsaufbaus zum KT
4041	Technical	Error	Fehler im Pairing, SICCT-Fehler(Nur wenn dieser Fehler wegen eines Fehlers auf der SICCT-Schnittstelle auftritt, ist der SICCT-Fehlercode mit anzugeben.): <SICCT-Fehler>
4042	Technical	Error	Die Version des Kartenterminals wird nicht unterstützt

Hinweis zu Fehler 4041:
Nur wenn dieser Fehler wegen eines Fehlers auf der SICCT-Schnittstelle auftritt, ist der SICCT-Fehlercode mit anzugeben.

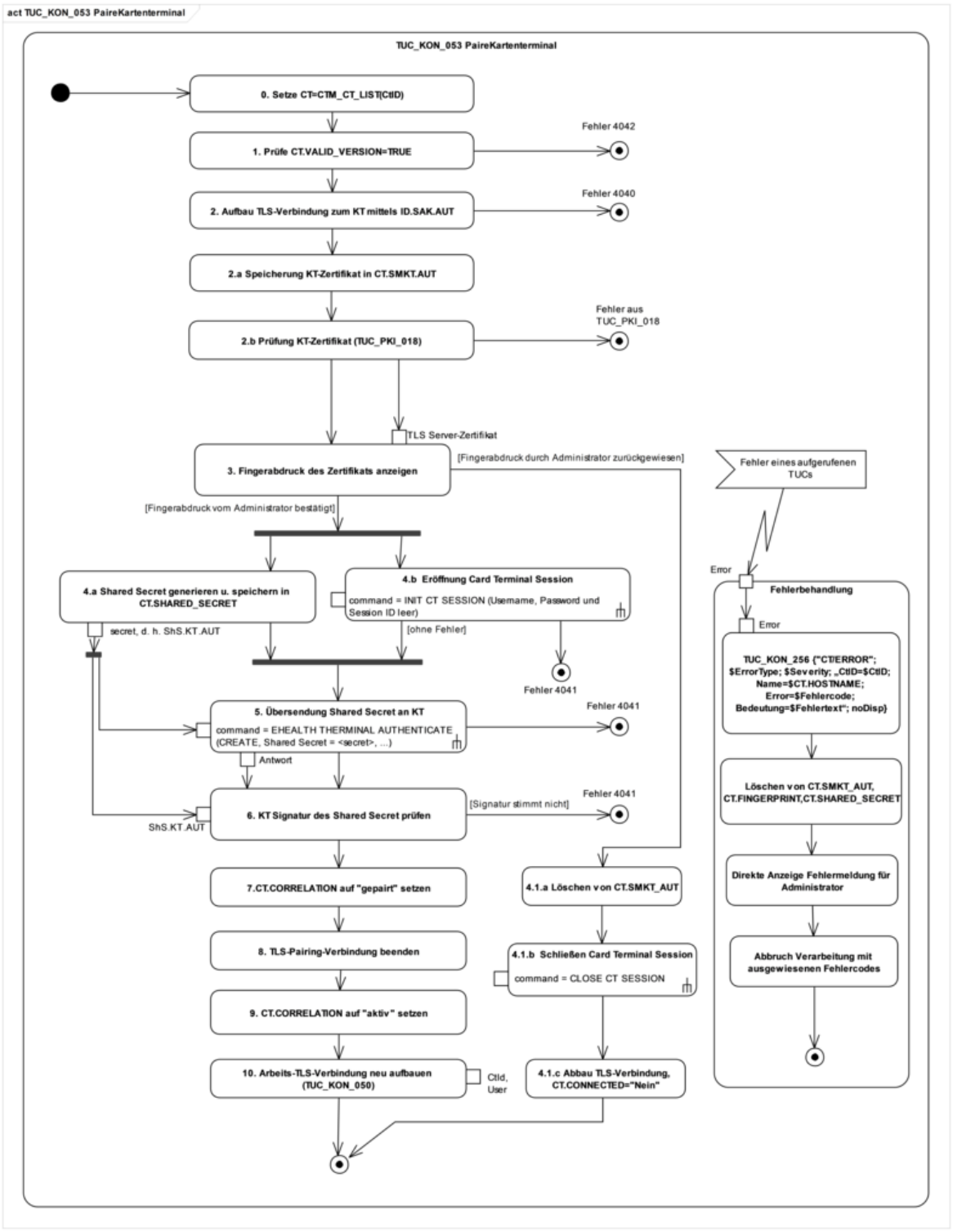


Abbildung 2: PIC_KON_057 Aktivitätsdiagramm zu „PaireKartenterminal"

[<=]

TIP1-A_4553-02 - Einsichtnahme in und Aktualisierung der Kartenterminaleinträge
Die Managementschnittstelle MUSS es einem Administrator ermöglichen zu jedem CT-Object-Eintrag in CTM_CT_LIST die Werte gemäß Tabelle TAB_KON_529 einsehen zu können:
Zu jedem Eintrag MUSS der Administrator TUC_KON_055 „Befülle CT-Object" auslösen können.

Tabelle 7: TAB_KON_529 Anzeigewerte zu einem Kartenterminalobjekt

ReferenzID	Belegung	Bedeutung und Administrator-Interaktion
Geräte kenndaten		
CT.CTID	Identifier	Eindeutige, statische Identifikation des Kartenterminals
CT.IS_PHYSICAL	Ja/Nein	Kennzeichnung, ob es sich um ein logisches oder physisches Kartenterminal handelt (siehe auch TAB_KON_522 Parameterübersicht des Kartenterminaldienstes)
CT.MAC_ADRESS	MAC-Adresse	die MAC-Adresse des Kartenterminals
CT.HOSTNAME	String	SICCT-Terminalname des Kartenterminals, auch als FriendlyName bezeichnet
CT.IP_ADRESS	IP-Adresse	die IP-Adresse des Kartenterminals
CT.TCP_PORT	Portnummer	der TCP-Port des SICCT-Kommandointerpreters des Kartenterminals
CT.SLOTCOUNT	Nummer	Anzahl der Slots des Kartenterminals
CT.SLOTS_USED	Liste	Liste der mit Karten belegten Slots
CT.PRODUCT INFORMATION	Inhalt Product Information.xsd	die Herstellerinformationen zum Kartenterminal gemäß [gemSpec_OM]
CT.EHEALTH_ INTERFACE_ VERSION	Version	Die EHEALTH-Interface-Version des Kartenterminals, die mittels des SICCT-Kommandos GET STATUS aus dem Element VER des Discretionary Data Objects ermittelt wurde
CT.VALID_ VERSION	Boolean	True, wenn die Version des Kartenterminals (CT.EHEALTH_INTERFACE_VERSION) durch den Konnektor unterstützt wird, d.h. zu den in CTM_SUPPORTED_KT_VERSIONS passt
Pairingdaten		
CT.SMKT_AUT	X.509-Cert	C.SMKT.AUT-Zertifikat des Kartenterminals, gespeichert im Rahmen des Pairings

ReferenzID	Belegung	Bedeutung und Administrator-Interaktion
CT.SMKT_AUT.CRYPT	RSA ECC	Kryptographie des gespeicherten C.SMKT.AUT-Zertifikats
Verbindungs daten		
CT. CORRELATION	bekannt zugewiesen gepairt aktiv aktualisierend	Der Korrelationsstatus zum Konnektor: • bekannt (über Service Announcement/Service Discovery gelernte Kartenterminals), • zugewiesen (durch den Administrator aus dem Bereich der bekannten Kartenterminals oder manuell konfigurierte Kartenterminals), • gepairt (Pairing erfolgreich aber noch nicht zum Verbindungsaufbau freigegeben) • aktiv (durch Administrator zum Verbindungsaufbau freigegeben), • aktualisierend (ein laufender Updatevorgang, ausgelöst durch den Konnektor; Der Zustand tritt ein, wenn der Kartenterminaldienst das Event „KSR/UPDATE/START" fängt und endet mit dem Event „KSR/UPDATE/END"),
CT.CONNECTED	Ja/Nein	Der Verfügbarkeitsstatus des Kartenterminals (Ja = nach Aufbau der TLS-Verbindung und erfolgter zweiter Authentifizierung)
CT.ACTIVEROLE	User/Admin	Benutzerrolle, die für die aktuelle Session verwendet wird
KT-Admin-Credentials		
CT.ADMIN_ USERNAME	String	Username des Administrators am KT

[<=]

2.1.1.6 Kapitel 4.1.5 - Kartendienst

Unter TIP1-A_4988* wird folgender Freitext hinzugefügt:

Gemäß [gemSpec_eGK_ObjSys_G2.1#4.6] gibt es Karten, deren RSA-Zertifikatscontainer vorhanden, jedoch nicht befüllt sind.

In Kapitel 4.1.5 wird Anforderung A_23614-01 durch A_23614-02 ersetzt:

A_23614-02 - SMC-B Prüfung bei Steckvorgang
Wenn der Konnektor einen Steckvorgang für eine SMC-B erkennt, MUSS der Konnektor - im Anschluss an die in TUC_KON_001 geforderten Aktionen- das ECC-Signaturzertifikat der SMC-B wie folgt prüfen. Wenn kein ECC-Zertifikat vorhanden ist, MUSS das RSA-Zertifikat geprüft werden:

- Wenn MGM_LU_ONLINE= "Enabled"

  TUC_KON_037 „Zertifikat prüfen"{
  certificate = C.HCI.OSIG;
  qualifiedCheck = required;
  offlineAllowNoCheck = true;
  validationMode = OCSP;
  getOCSPResponses = includeRevocationInfo}

- Ist das Ergebnis der Statusprüfung "good", MUSS die OCSP-Response im Konnektor gespeichert werden.
  Die maximale Dauer der Speicherung von SMC-B-Informationen im Konnektor ist in TIP1-A_4558 festgelegt.
- Ist das Ergebnis der Statusprüfung nicht "good" MUSS der Konnektor ein Event auslösen: ("SMC-B Status not good") TUC_KON_256 „Systemereignis absetzen" {

  topic = „CERT/CARD/STATUS";
  eventType = Op;
  severity = Warning;
  parameters = („CARD_TYPE=$Type,
  ICCSN=$ICCSN,
  CARD_HANDLE=$CardHandle,
  CardHolderName=$CardHolderName,
  CertName=$Name von certificate,
  ExpirationDate=$validity"

  CARD_CERTSTATUS= $CARD.CERTSTATUS

  doLog=false;
  doDisp = true }

- Wenn MGM_LU_ONLINE= "Disabled",

  TUC_KON_037 „Zertifikat prüfen"{
  certificate = C.HCI.OSIG;
  qualifiedCheck = required;
  offlineAllowNoCheck = true;
  validationMode = NONE }

  und Systemereignis senden

  (SMC-B Status not available") TUC_KON_256 „Systemereignis absetzen" {

  topic = „CERT/CARD/STATUS";
  eventType = Op;
  severity = Warning;

parameters = („CARD_TYPE=$Type,
ICCSN=$ICCSN,
CARD_HANDLE=$CardHandle,
CardHolderName=$CardHolderName,
CertName=$Name von certificate,
ExpirationDate=$validity“,
CARD_CERTSTATUS= "NotAvailable")
doLog=false;
doDisp = true }

Außerdem MUSS der Konnektor für das ECC-AUT-Zertifikat der SMC-B (C.HCI.AUT) den Zertifikatsablauf wie folgt prüfen. Wenn kein ECC-Zertifikat vorhanden ist, MUSS das RSA-Zertifikat geprüft werden:
TUC_KON_033 „Zertifikatsablauf prüfen“ {
cardSession;
doInformClients = true }**[<=]**

In Kapitel 4.1.5 wird Anforderung A_23311 durch A_23311-01 ersetzt:

A_23311-01 - HBA Prüfung bei Steckvorgang
Wenn der Konnektor einen Steckvorgang für einen HBAx erkennt, MUSS der Konnektor - im Anschluss an die in TUC_KON_001 geforderten Aktionen- das ECC-Signaturzertifikat des HBAx wie folgt prüfen. Wenn kein ECC-Zertifikat vorhanden ist, MUSS das RSA-Zertifikat geprüft werden:

- Wenn MGM_LU_ONLINE= "Enabled" und die Verbindung zum VPN-Konzentrator TI aufgebaut ist

  TUC_KON_037 „Zertifikat prüfen“{
  certificate = C.HP.QES;
  qualifiedCheck = required;
  offlineAllowNoCheck = true;
  validationMode = OCSP;
  getOCSPResponses = includeRevocationInfo}
- Ist das Ergebnis der Statusprüfung "good", MUSS die OCSP-Response im Konnektor gespeichert werden.
  Die maximale Dauer der Speicherung von HBA-Informationen im Konnektor ist in TIP1-A_4558 festgelegt.
- Ist das Ergebnis der Statusprüfung nicht "good" MUSS der Konnektor ein Event auslösen: ("HBA Status not good") TUC_KON_256 „Systemereignis absetzen“ {

  topic = „CERT/CARD/STATUS”;
  eventType = Op;
  severity = Warning;
  parameters = („CARD_TYPE=$Type,
  ICCSN=$ICCSN,
  CARD_HANDLE=$CardHandle,
  CardHolderName=$CardHolderName,
  CertName=$Name von certificate,
  ExpirationDate=$validity“

CARD_CERTSTATUS= $CARD.CERTSTATUS

doLog=false;
doDisp = true }

- Wenn MGM_LU_ONLINE= "Disabled",

TUC_KON_037 „Zertifikat prüfen“{
certificate = C.HP.QES;
qualifiedCheck = required;
offlineAllowNoCheck = true;
validationMode = NONE }

und Systemereignis senden

(HBA Status not available") TUC_KON_256 „Systemereignis absetzen“ {

topic = „CERT/CARD/STATUS”;
eventType = Op;
severity = Warning;
parameters = („CARD_TYPE=$Type,
ICCSN=$ICCSN,
CARD_HANDLE=$CardHandle,
CardHolderName=$CardHolderName,
CertName=$Name von certificate,
ExpirationDate=$validity“,

CARD_CERTSTATUS= "NotAvailable")

doLog=false;
doDisp = true }

Außerdem MUSS der Konnektor für das ECC-AUT-Zertifikat des HBAx (C.HP.AUT) den Zertifikatsablauf wie folgt prüfen. Wenn kein ECC-Zertifikat vorhanden ist, MUSS das RSA-Zertifikat geprüft werden:
TUC_KON_033 „Zertifikatsablauf prüfen“ {
cardSession;
doInformClients = true }
[<=]

Zum Monitoring von G2.0 Karten wird folgende Anforderungen ergänzt (funktionale Eignung: Test):

A_25801 - Loggen von G2.0 Karten nach dem Stecken.
Der Konnektor MUSS immer wenn eine G2.0 SMC-B oder ein G2.0 HBA gesteckt wird im Anschluss an TUC_KON_001 folgende Aktionen ausführen:
1. $pseudonym berechnen als erste 10 Zeichen von SHA256(ICCSN, Ablaufdatum)
2. einen Protokolleintrag erstellen mit TUC_KON_271 „Schreibe Protokolleintrag“ {
topic=„CARD/G2“;
eventType=Op;
severity=Warnung;
parameters =(„CardType=[HBA,SMC-B],
Pseudonym=$pseudonym,
cardholder=CARD.CARDHOLDER
ExpiryDate=$Ablaufdatum
message=Karte vor dem 1.1.2026 tauschen")}
3. Setze abhängig vom $cardType denBetriebszustand

EC_G2_SMCB_USED($pseudonym) oder EC_G2_HBA_USED($pseudonym) mit den Parameter $expiryDate. Setze oder aktualisiere $ValidFrom auf die aktuellen Systemzeit.

[<=]

Funkt.Eignung: Test

TIP1-A_4565-02 wird ersetzt durch:

TIP1-A_4565-03 - TUC_KON_001 „Karte öffnen"
Der Konnektor MUSS den technischen Use Case „Karte öffnen" gemäß TUC_KON_001 umsetzen.

Tabelle 8: TAB_KON_734 – TUC_KON_001 „Karte öffnen"

Element	Beschreibung
Name	TUC_KON_001 „Karte öffnen"
Beschreibung	Der TUC initialisiert ein Card-Object basierend auf einer physikalischen Karte und fügt es CM_CARD_LIST zu. Die Karte kann erst im Anschluss unter Verwendung des erzeugten CardHandles verwendet werden.
Auslöser	Der Kartenterminaldienst meldet das Belegen eines KT-Slots
Vorbedingungen	• In ctId/slotId steckt eine Karte
Eingangsdaten	• ctId (Kartenterminalidentifikator) • slotId (Nummer des Kartenslots)
Komponenten	Karte, Kartenterminal, Konnektor
Ausgangsdaten	Keine

Standardablauf	1. Prüfe, ob unter ctId und slotId ein Eintrag in CM_CARD_LIST vorhanden ist. Wenn bereits ein Eintrag vorhanden ist, lösche diesen.
	2. Erzeuge neuen Card-Object-Eintrag in CM_CARD_LIST und
	a) Generiere CARD.CARDHANDLE. mit folgenden Anforderungen:
	
	- Das CardHandle MUSS innerhalb CM_CARD_LIST eindeutig sein.
	- Ein ungültig gewordenes CardHandle DARF innerhalb von 48h NICHT als neues CardHandle vergeben werden.
	b) Befülle CARD.CTID und CARD.SLOTNO mit den Eingangsdaten
	c) Ermittle und befülle (soweit durch Karte unterstützt) die folgenden Daten:
	- CARD.ICCSN
	- CARD.TYPE (mögliche Werte siehe Tabelle TAB_KON_500 Wertetabelle Kartentypen)
	- CARD.CARDVERSION
	- CARD.INSERTTIME (=aktuelle Systemzeit)
	- CARD.CARDHOLDERNAME (aus X.509-AUT-Zertifikat)
	- CARD.KVNR (nur für eGK, aus C.CH.AUT: unveränderbarer Teil der KVNR)
	- CARD.CERTEXPIRATIONDATE (=validity aus X.509-AUT-Zertifikat)
	
	X.509-AUT-Zertifikat bezeichnet für eGK das C.CH.AUT-Zertifikat, für HBAx das C.HP.AUT-Zertifikat und für SMC-B das C.HCI.AUT-Zertifikat. Wenn vorhanden, ist das ECC-Zertifikat zu verwenden, andernfalls das RSA-Zertifikat.
	
	3. Rufe TUC_KON_256{
	topic = „CARD/INSERTED“;
	eventType = Op;
	severity = Info;
	parameters = <Params>}
	mit <Params> belegt aus dem CARD-Object: „CardHandle=$, CardType=$, CardVersion=$, ICCSN=$,CtID=$, SlotID=$, InsertTime=$, CardHolderName=$, KVNR=$, CertExpirationDate=$“
	
	In CardVersion sind die Werte
	- COSVERSION und
	- OBJECTSYSTEMVERSION
	aus CARD.CARDVERSION einzutragen. Für eGK G1+ ist zusätzlich die
	- DATASTRUCTUREVERSION
	aus CARD.CARDVERSION einzutragen. CardVersion kann weitere

Element	Beschreibung
	Werte aus CARD.CARDVERSION enthalten.
Varianten/ Alternativen	Im Falle der KVK gibt es kein EF.ATR, EF.GDO und EF.DIR. Es wird daher lediglich der ATR ausgewertet, den das Kartenterminal beim Stecken der Karte liefert.
Fehlerfälle	(-> 2c) Karte/Kartenterminal antwortet mit einer spezifischen Fehlermeldung, Fehlercode <gemäß [gemSpec_COS]/[SICCT]> Auch im Fehlerfall wird Schritt 3 durchlaufen. Wenn nicht alle zu einem Kartentyp notwendigen Daten von der Karte gelesen werden konnten, dann wird Schritt 3 mit CardType=UNKNOWN ausgeführt. Auch im Fehlerfall wird Schritt 3 durchlaufen. Wenn nicht alle zu einem Kartentyp notwendigen Daten von der Karte gelesen werden konnten, dann wird Schritt 3 mit CardType=UNKNOWN ausgeführt.
Nichtfunktionale Anforderungen	Keine
Zugehörige Diagramme	Keine

[<=]

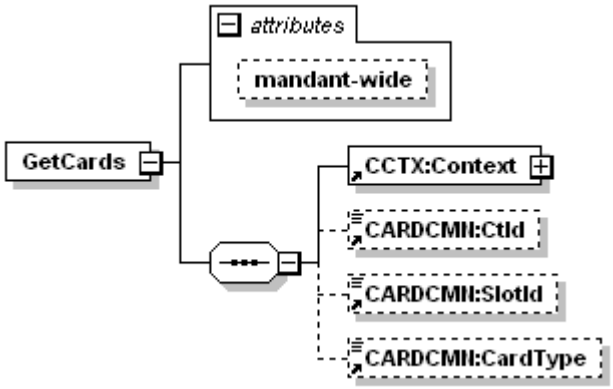
2.1.1.7 Kapitel 4.1.6 - Systeminformationsdienst

In Kapitel 4.1.6.5.2 wird Anforderung TIP1-A_4605 durch TIP1-A_4605-02 ersetzt:

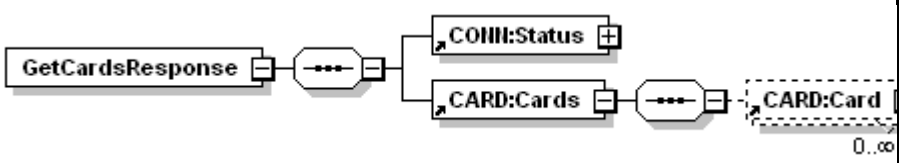
Der Rückgabeparameter CertificateExpirationDate wird so beschrieben werden, dass dieser nicht irgendein Zertifikat einer Dualpersonaliserten Konnektorinstanz zurückgibt, sondern explizit die ECC-Variante.

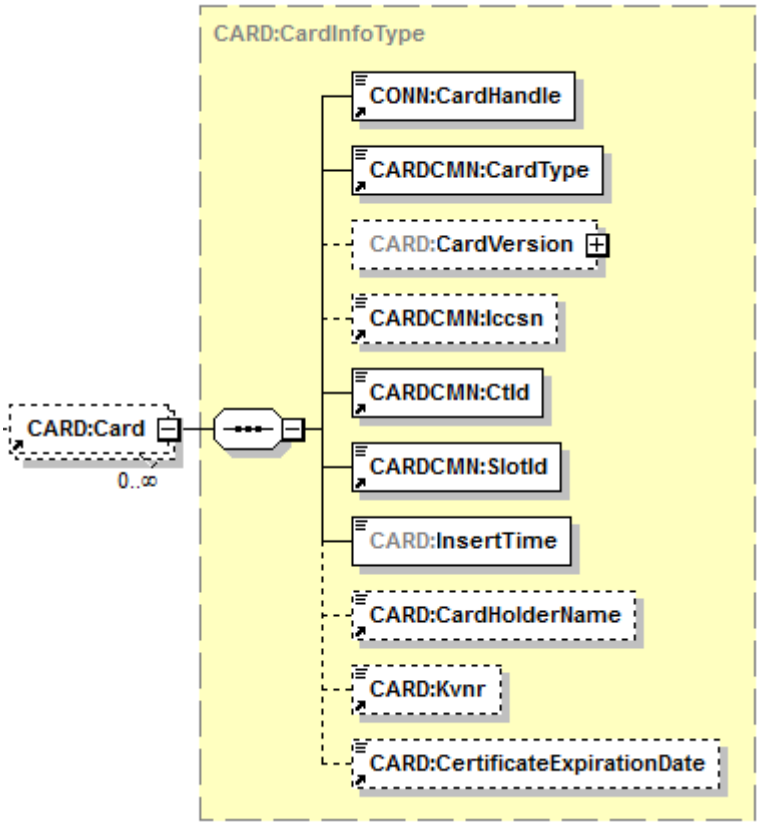
TIP1-A_4605-02 - Operation GetCards
Der Konnektor MUSS an der Außenschnittstelle eine Operation GetCards, wie in Tabelle TAB_KON_565 „Operation GetCards“ beschrieben, anbieten und MUSS dabei Kartentypen aus Tabelle TAB_KON_500 Wertetabelle Kartentypen unterscheiden.

Tabelle 9: TAB_KON_565 Operation GetCards

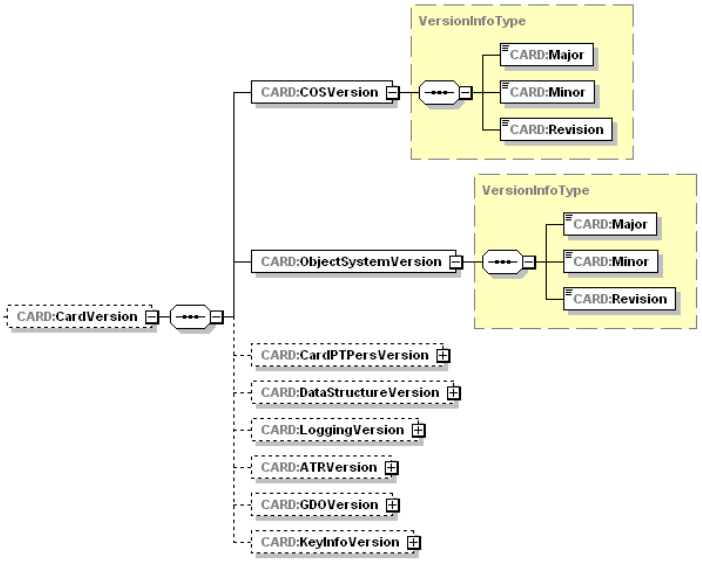
<table>
<tr><td>Name</td><td>GetCards</td></tr>
<tr><td>Beschreibung</td><td>Liefert Informationen zu den in den Kartenterminals verfügbaren Karten zurück, die in Kartenterminals stecken, auf die Mandant und Clientsystem zugreifen dürfen. Insbesondere umfasst die Information die sog. Karten-Handles. Die Karten-Handles können bei anderen Konnektoraufrufen zur Adressierung von Karten genutzt werden.</td></tr>
<tr><td>Aufrufparameter</td><td> 

<table>
<tr><th>Name</th><th>Beschreibung</th></tr>
<tr><td>@mandant-wide</td><td>Wenn „true“, werden alle Karten zurückgegeben, auf die der Mandant und das aufrufende Clientsystem zugreifen dürfen. Wenn „false“ (Standardbelegung), werden nur Karten zurückgegeben, auf die von dem im Aufrufkontext spezifizierten Arbeitsplatz zugegriffen werden darf.</td></tr>
<tr><td>Context</td><td>Aufrufkontext</td></tr>
<tr><td>CtId</td><td>Identifikation des Kartenterminals. Wenn angegeben, werden nur die Karten zurückgeliefert, die in diesem Kartenterminal verfügbar sind.</td></tr>
<tr><td>SlotId</td><td>Nummer des Slots, beginnend bei 1.
Wird zusätzlich zur CtId auch SlotId übergeben, so wird die Karte zurückgegeben, die in dem angegebenen Slot des mit CtId adressierten Kartenterminals steckt.</td></tr>
</table>
</td></tr>
</table>

Name	GetCards	
	CardType	Ein Kartentyp gemäß Tabelle TAB_KON_500 „Wertetabelle Kartentypen" als optionaler Filter. Wenn angegeben, werden nur Karten vom spezifizierten Typ zurückgegeben. Unterstützt werden die Kartentypen EGK, KVK, HBAx, SM-B, SMC-KT und UNKNOWN.
Antwort		
	Name	**Beschreibung**
	Status	Ergebnis der Operation

Name	GetCards
	Im Element `Cards` wird die Liste der gesteckten Karten zurückgegeben. Für jede Karte wird dabei ein `Card`-Element angegeben. Leere Slots der Kartenterminals sind in dieser Liste nicht enthalten. 

Name	Beschreibung
`Card Handle`	Handle, mit dem die Karte in Folgeaufrufen adressiert werden kann. Der Konnektor garantiert, dass dieses Handle die gesteckte Karte eindeutig identifiziert und bei Entfernen der Karte aus dem Kartenterminal ungültig wird. Auch für nicht erkannte Karten (z. B. bei falscher Steckrichtung der Karte) SOLL der Konnektor gültige Handles liefern (sofern das Kartenterminal in diesem Fall in der Lage ist, das entsprechende Ereignis „Karte wurde gesteckt" zu liefern), damit diese Karten z. B. zum Auswurf adressiert werden können.

Name	GetCards	
	CardType	Erkannter Typ der Karte. Siehe Tabelle TAB_KON_500 Wertetabelle Kartentypen,
	Card Version	 Der Konnektor MUSS in CardVersion bei eGK, HBA und SM-B/SMC-KT der Generation 2 die Versionsinformationen gemäß [gemSpec_Karten_Fach_TIP] übergeben, für G1+ aus /MF/EF.Version. Bei KVK, HBA-VK und unbekannten Karten MUSS das Element weggelassen werden.
	Iccsn	Falls auslesbar, die ICC-Serial-Number der Karte. Im Fall der KVK wird das optionale Element Iccsn nicht zurückgegeben.
	CtId	Identifikation des Kartenterminals, in dem die Karte steckt.
	SlotId	Nummer des Slots (beginnend bei 1), in dem die Karte steckt.

Name	GetCards	
	InsertTime	Gibt den Zeitpunkt an, zu dem der Konnektor die Karte erkannt hat. Die Zeit wird mit dem Datentyp DateTime in folgendem Format angegeben: yyyy-mm-ddThh:mm:ss+hh:mm Es ist also – gemäß ISO 8601 [ISO8601] – die lokale Zeit und die Differenz zur UTC anzugeben.
	CardHolderName	Name des Karteninhabers bzw. der Institution/Organisation (subject.commonName). Bei KVK und unbekannten Karten MUSS das Element weggelassen werden.
	Kvnr	KVNR (Unveränderbarer Teil) MUSS bei eGK belegt werden. Bei allen anderen Karten MUSS das Element weggelassen werden.
	CertificateExpirationDate	Ablaufdatum des Zertifikates (AUT bzw. OSIG). Bei KVK und unbekannten Karten MUSS das Element weggelassen werden. Bei dualpersonalisierten Karten ist lediglich das Ablaufdatum des ECC Zertifikats zurückzugeben
Vorbedingungen	Keine.	
Nachbedingungen	Der Zustand der Karten und der Kartenterminals bleibt unverändert.	
Hinweise	Der Aufruf darf nur den im Konnektor verwalteten aktuellen Zustand der Karte liefern und keine Abfragen an die Kartenterminals absetzen.	

Der Ablauf der Operation GetCards ist in Tabelle TAB_KON_566 Ablauf GetCards beschrieben:

Tabelle 10: TAB_KON_566 Ablauf GetCards

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.

Nr.	Aufruf Technischer Use Case oder Interne Operation	Beschreibung

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

3.	TUC_KON_253 „Liefere Karten_Liste“	Die Liste der Karten wird erstellt und zurückgegeben. Tritt hierbei ein Fehler auf, so bricht die Operation mit dem Fehler des TUCs ab.
		Wenn `@mandant-wide=true` dann ermittle die Liste der Karten für alle Arbeitsplätze des Mandanten für das angegebene Clientsystem durch den Aufruf
		TUC_KON_253 „Liefere Karten_Liste“ {
		clientSystemId = $context.clientsystemId;
		cardTerminalId = CtId;
		slotId = SlotId;
		mandantId = $context.mandantId;
		cardType = CardType }
		Wenn `@mandant-wide=false` dann ermittle die Liste der Karten für den Arbeitsplatz des Mandanten für das angegebene Clientsystem entsprechend $context durch den Aufruf
		TUC_KON_253 „Liefere Karten_Liste“ {
		workplaceId= $context.workplaceId;
		clientSystemId = $context.clientsystemId;
		cardTerminalId = CtId;
		slotId = SlotId;
		mandantId = $context.mandantId;
		cardType = CardType }

Die Fehlerfälle der Operation GetCards sind in Tabelle TAB_KON_567 Fehlercodes „GetCards dargestellt:

Tabelle 11: TAB_KON_567 Fehlercodes „GetCards“

Fehlercode	ErrorType	Severity	Fehlertext
Neben den Fehlercodes der aufgerufenen technischen Use Cases können folgende weiteren Fehlercodes auftreten:			
4000	Technical	Error	Syntaxfehler

[<=]

2.1.1.8 Kapitel 4.1.7 - Verschlüsselungsdienst

In Kapitel 4.1.7.1 wird TAB_KON_747 wie folgt angepasst:

- Für crypt=RSA und crypt=RSA_ECC wird das Zertifikat abhängig von der Kartengeneration gewählt. Eine parallele RSA Verschlüsselung bei crypt=RSA_ECC entfällt, da sie die gesamte Verschlüsselung unnötig unsicher machen würde.
- HBA-Vorläuferkarten sind nicht mehr im Feld vorhanden. Daher können die entsprechende Einträge gelöscht werden.
- Die Spalte CRYPT sowie KeyReference werden aufgrund der Änderungen von TIP1-A_4621-06 entfernt.
- Da nur durch Fachmodule Nutzbar, kommen für eGKs nur NFDM, AMTS, VSDM, ePA in Frage. Das ePA 2.x Fachmodul fällt bis PTV6 Rollout weg. Neue Fachmodule soll es nicht geben. Also kann die Zeile für eGK gestrichen werden.

A_17746-01 - Einsatzbereich und Vorgaben für Ver- und Entschlüsselung (ECC-Migration)

Der Konnektor MUSS für die kartenbasierte Ver- und Entschlüsselung die Zertifikate und Schlüssel in Abhängigkeit vom kryptographischen Verfahren unter Berücksichtigung des Einsatzbereiches aus TAB_KON_747_01 ermitteln.

Tabelle 12: TAB_KON_747_01 KeyReference für Encrypt-/DecryptDocument

Karte	~~KeyReference~~	Zertifikat (Encrypt) ...in DF.ESIGN	Schlüssel (Decrypt) ...in DF.ESIGN	Einsatzbereich	
				Außen-schnittstelle	Fachmodul-schnittstelle
HBA	~~C.ENC~~	**[ab G2.1] :** EF.C.HP.ENC.E256 **[G2.0] :** EF.C.HP.ENC.R2048	PrK.HP.ENC.E256 PrK.HP.ENC.R2048	Ja	Ja
SM-B	~~C.ENC~~	**[ab G2.1] :** EF.C.HCI.ENC.E256 **[G2.0] :** EF.C.HCI.ENC.R2048	PrK.HP.ENC.E256 PrK.HCI.ENC.R2048	Ja	Ja
~~HBA-VK~~	~~C.ENC~~	~~EF.C.HP.ENC~~	~~PrK.HP.ENC~~	~~Ja~~	~~Ja~~

Karte	~~KeyReference~~	Zertifikat (Encrypt) ...in DF.ESIGN	Schlüssel (Decrypt) ...in DF.ESIGN	Einsatzbereich	
				Außen-schnittstelle	Fachmodul-schnittstelle
~~eGK~~	~~C.ENC~~	~~**[ab G2.1] :** C.CH.ENC.E256 **[G2.0] :** C.CH.ENC.R2048~~	~~PrK.CH.ENC.E256 PrK.CH.ENC.R2048~~	~~Nein~~	~~Ja~~

[<=]

In Kapitel 4.1.7.1 wird TAB_KON_859 gestrichen:

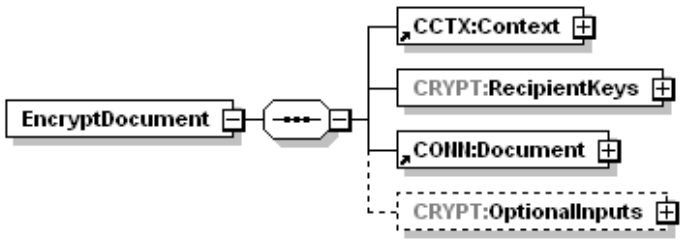
~~**Tabelle 13: TAB_KON_859 Werteliste und Defaultwert des Parameters crypt bei hybrider Verschlüsselung**~~

~~**Typname**~~	~~**Werteliste**~~	~~**Defaultwert**~~	~~**Bedeutung**~~
~~ENC_CRYPT~~	~~RSA ECC RSA_ECC~~	~~RSA ECC~~	~~Werteliste des Parameters crypt bei der hybriden Verschlüsselung Es wird gem. TAB_KON_747 verschlüsselt. RSA: Es wird RSA-2048 basiert verschlüsselt. ECC: Es wird ECC-256 basiert verschlüsselt. RSA_ECC: Es wird dual RSA-2048 basiert und ECC-256 basiert verschlüsselt. Es wird als Fehlerfall gewertet, wenn nicht alle beiden Zertifikate weder (RSA- noch und ECC-Zertifikat) von der Karte geladen werden konnten., und als Warnung, wenn nur ein Zertifikat geladen werden konnte.~~

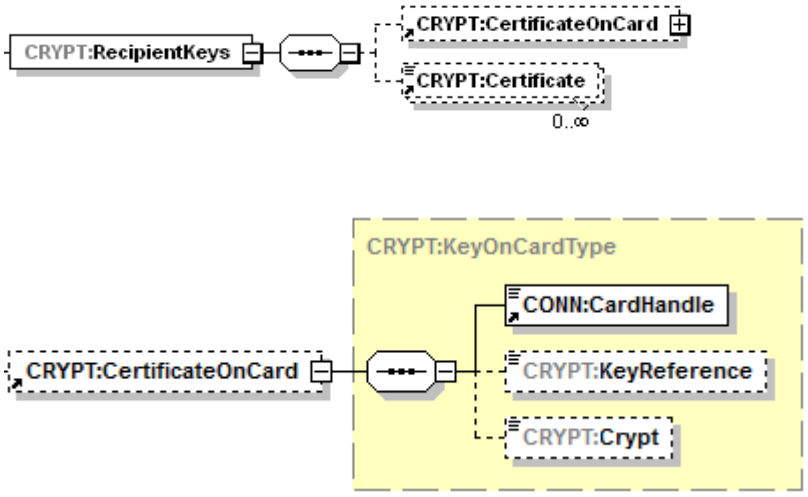
In Kapitel 4.1.7.5.1 wird Anforderung TIP1-A_4621-05 durch TIP1-A_4621-06 ersetzt:

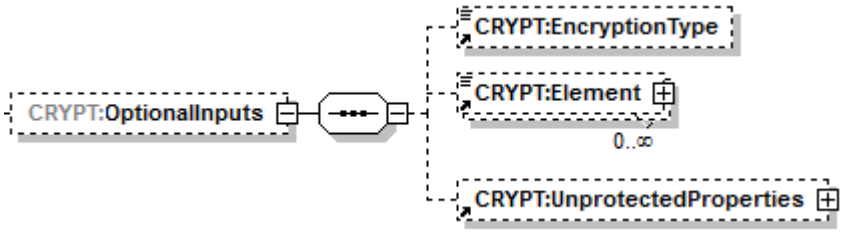
TIP1-A_4621-06 - Operation EncryptDocument
Der Basisdienst Verschlüsselungsdienst des Konnektors MUSS an der Clientschnittstelle eine Operation EncryptDocument anbieten.

Tabelle 14: TAB_KON_071 Operation EncryptDocument

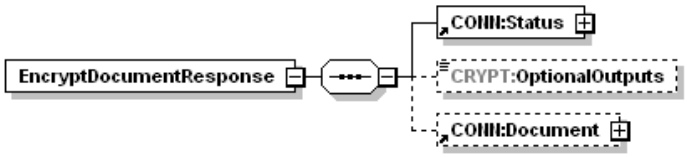
Name	`EncryptDocument`
Beschreibung	Diese Operation verschlüsselt ein übergebenes Dokument hybrid. Es werden die Dokumententypen `Alle_DocFormate` unterstützt. Für die hybride Verschlüsselung wird ein asymmetrischer Schlüssel aus einem X.509v3-Zertifikat genutzt. Dieses Zertifikat kann von einer Karte kommen oder als Parameter übergeben werden. Pro Operationsaufruf können mehrere Hybridschlüssel erzeugt werden. Übergibt der Aufrufer die Zertifikate beim Aufruf, wird passend zum öffentlichen Schlüssel aus dem jeweiligen Zertifikate ein RSA-basiertes oder ECC-basiertes Verschlüsselungsverfahren verwendet. Wenn das Verschlüsselungszertifikat von einer Karte kommt, bestimmt die Kartengeneration das Verschlüsselungsverfahren. Der ~~kann der Aufrufer durch Angabe des Kryptoverfahrens~~ ~~über den~~ Parameter `crypt` ~~steuern~~ ist weitgehend wirkungslos, ~~ob Hybridschlüssel für RSA oder für ECC oder beide dual für ECC sowie RSA erzeugt werden. Das Defaultverhalten ist die Hybridschlüsselerzeugung für RSA ECC.und entspricht dem Verhalten aus der Version 6.1.0 der Schnittstelle.~~ Es werden die folgenden Karten unterstützt: HBAx und SM-B. Die Operation EncryptDocument DARF das Verschlüsseln mit der eGK NICHT unterstützen. Für alle Dokumenttypen wird immer das gesamte Dokument verschlüsselt. Vom Konnektor zu verschlüsselnde Dokumente müssen konform zu den Anforderungen in Kapitel 3.1.1 "Dokumentformate" sein.
	

Name	Beschreibung
`Context`	Aufrufkontext: • MandantID, ClientSystemID, WorkplaceId verpflichtend • UserID verpflichtend bei HBAx, bei SM-B nicht ausgewertet

Name	EncryptDocument	
	 Das RecipientKeys-Element identifiziert die Empfänger der zu verschlüsselnden Nachricht über X.509-Zertifikate (öffentliche Schlüssel). Quelle für die Zertifikate kann eine gesteckte 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.
	`KeyReference`	Der Parameter wird nicht ausgewertet - Optional; ~~Der Wert dieses Parameters ist in Tabelle TAB_KON_747 KeyReference für Encrypt-/DecryptDocument spezifiziert. Ist der Parameter nicht angegeben, gilt der Default-Wert C.ENC.~~
	`Crypt`	Der Parameter wird nicht ausgewertet - Optional; Das zu verwendende Zertifikat ist kartenabhängig gemäß Tabelle TAB_KON_747 zu wählen. ~~Default: siehe TAB_KON_859~~ ~~Wertebereich: [ENC_CRYPT] aus TAB_KON_859~~ ~~Gibt den Typ von Zertifikaten vor, die von der per CardHandle referenzierten Karte für die Erzeugung der Hybridschlüssel gemäß Tabelle TAB_KON_747 verwendet werden.~~

Name	EncryptDocument	
	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. Der Konnektor MUSS eine Liste von mindestens 1000 Zertifikaten unterstützen. Kommt das Zertifikat ausschließlich von einer Karte, dann kann dieser Parameter weggelassen werden.
		
	CONN: Document	Dieses entsprechend [OASIS-DSS] Section 2.4.2 spezifizierte Element enthält das zu verschlüsselnde Dokument, wobei die Kindelemente CONN:Base64XML und dss:Base64Data verwendet werden. Im Fall dss:Base64Data wird ein etwaig übergebenes MIME-Type-Attribut nicht ausgewertet.
		
	CRYPT: OptionalInputs	Enthält eine Auswahl der folgenden unten näher erläuterten (optionalen) Eingabeparameter:
		

Name	EncryptDocument	
	EncryptionType	Zu wählendes Verschlüsselungsverfahren, wobei folgende URI vorgesehen sind: • XMLEnc: „http://www.w3.org/TR/xmlenc-core/" • CMS: „urn:ietf:rfc:5652" • S/MIME: „urn:ietf:rfc:5751" Im Fall XMLEnc wird ein Base64-codiertes XML-Dokument im Element CONN:Document/CONN:Base64XML übergeben. In den Fällen CMS und S/MIME wird ein Base64-codiertes Binär-Dokument im Element CONN:Document/dss:Base64Data übergeben . Ist der Parameter EncryptionType nicht gesetzt, dann gilt folgendes Default-Verhalten: Für ein im Element CONN:Document/CONN:Base64XML übergebenes XML-Dokument wird als Verschlüsselungsverfahren [XMLEnc] angewandt, und für ein im Element CONN:Document/dss:Base64Data übergebenes Dokument wird das Verschlüsselungsverfahren CMS angewandt. XML-Documente werden nach Type=http://www.w3.org/2001/04/xmlenc#Element verschlüsselt. Im Fall S/MIME ist das in CONN:Document/dss:Base64Data übergebene Dokument eine MIME-Nachricht. Der Konnektor KANN die Operation im Fall S/MIME mit Fehlercode 4279 beenden.
	Element	
	Element	Der Parameter wird nicht ausgewertet.
	UnprotectedProperties	
	CRYPT:UnprotectedProperties	Dieses optionale Element wird im CMS-Fall (EncryptionType = urn:ietf:rfc:5652) ausgewertet. 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.

Name	EncryptDocument	
Rückgabe		
	Status	Enthält den Ausführungsstatus der Operation.
	CRYPT: Optional Outputs	Kann – in zukünftigen Versionen der Spezifikation – optionale Ausgabeparameter enthalten.
	CONN: Document	Enthält das verschlüsselte Dokument in base64-codierter Form, wenn die Verschlüsselung erfolgreich durchgeführt wurde. Im Fall XMLEnc wird das Base64-codierte verschlüsselte XML-Dokument im Element CONN:Document/CONN:Base64XML zurückgegeben. Im Fall CMS wird das Base64-codierte Binär-Dokument im Element CONN:Document/dss:Base64Data zurückgegeben. Im Fall S/MIME wird die Base64-codierte S/MIME-Nachricht im Element CONN:Document/dss:Base64Data zurückgegeben. Das Attribut CONN:Document/dss:Base64Data/@MimeType wird auf „application/pkcs7-mime" gesetzt. Die S/MIME-Nachricht hat Content-Transfer-Encoding: base64.
Fehler	Bei Auftreten eines Fehlers im Standardablauf werden Fehlercodes entsprechend TAB_KON_141 gemeldet.	
Vorbedingungen	Keine	
Nachbedingungen	Keine	

Der Ablauf der Operation EncryptDocument ist in Tabelle TAB_KON_746 Ablauf EncryptDocument beschrieben:

Tabelle 15: TAB_KON_746 Ablauf EncryptDocument

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. Wird die Operation für einen nicht unterstützten Kartentypen aufgerufen, so bricht die Operation mit Fehler 4058 ab.
2.	checkDocumentFormat	Der Konnektor prüft für das zu verschlüsselnde Dokument, ob die Vorgaben aus Kapitel 3.1.1 eingehalten sind. Bei Verletzung einer Vorgabe bricht der Konnektor die Verarbeitung mit einem entsprechenden Fehlercode ab.

3.	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.

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

5.	TUC_KON_070 „Daten hybrid verschlüsseln“	Die hybride Verschlüsselung wird durchgeführt. Tritt hierbei ein Fehler auf, bricht die Operation ab. Die KeyInfo, d.h. die Liste der Hybridschlüssel inklusive des bei ihrer Erzeugung verwendeten Zertifikates, sind dabei in das Dokument einzubetten.

Tabelle 16: TAB_KON_141 Fehlercodes „EncryptDocument“

Fehlercode	ErrorType	Severity	Fehlertext
Neben den Fehlercodes der aufgerufenen technischen Use Cases und Fehlercodes aus Kapitel 3.1.1 können folgende weitere Fehlercodes auftreten:			
4000	Technical	Error	Syntaxfehler
4001	Security	Error	Interner Fehler
4058	Security	Error	Aufruf nicht zulässig
4279	Technical	Error	S/MIME-Funktionalität nicht unterstützt"

Fehlercode	ErrorType	Severity	Fehlertext
4283	Technical	Error	Dokument zu groß

[<=]

2.1.1.9 Kapitel 4.1.8 - Signaturdienst

Neue Anforderung (funktionale Eignung: test)

A_25834 - QES-Signaturprüfung mit potentiell unsicheren Algorithmen
Der Konnektor MUSS eine ansonsten fehlerfreie QES-Signatur mit Prüfergebnis VALID belegen, wenn die Algorithmen nach [ALGCAT] zum Zeitpunkt der Prüfung nicht mehr als sicher eingestuft wird. Der verminderte Beweiswert der Signatur MUSS dem Aufrufer über SIG:VerifyDocumentResponse/SIG:OptionalOutputs/vr:VerificationReport/vr:IndividualReport/dss:Result/dss:ResultMessage="veraltete Signatur mit vermindertem Beweiswert" zurückgemeldet werden. Die ResultMessage MUSS unterbleiben, wenn der Beweiswert mit einer Gegensignatur erhalten wurde (TIP1-A_5682-01)**[<=]**

Zur Signaturerstellung wird folgender informativer Text ergänzt.

Das Verhalten für die Signaturerstellung wird die Vertrauenswürdigkeit der CA des Signaturzertifikats gesteuert.

TIP1-A_5682 wird abgelöst durch TIP1-A_5682-01

TIP1-A_5682-01 - Nicht geeignete Algorithmen im VerificationReport
Der Konnektor MUSS im VerificationReport einer QES-Signaturprüfung ausweisen, wenn die für die Signatur verwendeten Algorithmen zum Zeitpunkt der Prüfung nach dem Algorithmenkatalog [ALGCAT] als nicht geeignet eingestuft werden.
Der Konnektor DARF eine Signaturalgorithmus NICHT als ungeeignet ausweisen, wenn:

- das Hashverfahren für den Dokumentenhash als sicher eingestuft wird und
- die QES-Signatur durch eine QES-Signatur mit geeignetem Algorithmus gegensigniert wurde und
- der Algorithmus der QES-Signatur zum Zeitpunkt der Gegensignatur geeignet war.

Eine ansonsten fehlerfreie Signatur mit beweiswerterhaltender Gegensignatur ist als VALID zu werten.

[<=]

TIP1-A_4672-05 wird abgelöst durch TIP1-A_4672-06

TIP1-A_4672-06 - TUC_KON_151 „QES-Dokumentensignatur prüfen“
Der Konnektor MUSS den technischen Use Case TUC_KON_151 „QES-Dokumentensignatur prüfen” umsetzen.

Tabelle 17: TAB_KON_591 - TUC_KON_151 „QES-Dokumentensignatur prüfen“

Element	Beschreibung
Name	TUC_KON_151 „QES-Dokumentensignatur prüfen”

Element	Beschreibung
Beschreibung	Es wird die QES eines Dokuments geprüft. Dabei werden die Signaturverfahren laut Tabelle 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. Es wird geprüft, ob die verwendeten Kryptoalgorithmen mit ihren Parametern in [ALGCAT] als "recommended" oder "legacy" geführt werden. Die Prüfung erfolgt mit der Systemzeit oder dem Signaturzeitpunkt einer gültigen Gegensignatur (siehe TIP1-A_5682-01). Das Prüfergebnis wird gemäß A_25834 verarbeitet. **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_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 jüngste 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

Element	Beschreibung
	Signatur eingebettet. Die Warnung PROVIDED_OCSP_RESPONSE_NOT_VALID wird **nicht** an den Aufrufer zurückgemeldet. 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_779-01 "Profilierung der Signaturformate" und Tabelle TAB_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. (→ 2 **„CoreValidation"**) Interner Fehler: 4001, Signatur des Dokuments ungültig: 4115, Signatur umfasst nicht das gesamte Dokument: 4262 (→3 **„CheckSignatureCertificate"**) Interner Fehler: 4001, Signaturzertifikat ermitteln ist fehlgeschlagen: 4206. (→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 18: 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
4264	Technical	Warning	Ein oder mehrere Zertifikate ignoriert

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

Tabelle 19: 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.

VerificationResult für gesamtes Dokument (VerificationResult/HighLevelResult)	
	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: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.

[<=]

Hinweis:

nonQES-Signaturen finden in der TI nur für kurzlebige Artefakte Anwendung. Daher wird keine Prüfung der Algorithmen gegen [ALGCAT] implementiert. Die Nutzung geeigneter Algorithmen wird über die Vertrauenswürdigkeit der CA und die Zulassung der verwendeten Komponenten gesteuert.

TIP-A-4629 wir durch TIP-A-4629 ersetzt:

TIP1-A_4629-01 - Unterstützte Karten QES-Erstellung
Der Signaturdienst MUSS für die QES-Erstellung die Kartentypen HBA G2.0 und höher ~~, HBA-qSig und ZOD_2.0~~ unterstützen.
[<=]

In Kapitel 4.1.8.1.1 wird A_17768 durch A_17768-01 ersetzt und TAB_KON_900 wie folgt angepasst:

- Für crypt=RSA und crypt=RSA_ECC wird das Zertifikat abhängig von der Kartengeneration gewählt.
- HBA-Vorläuferkarten sind nicht mehr im Feld vorhanden. Daher können die entsprechende Einträge gelöscht werden.
- Die Spalte CRYPT wird aufgrund der Änderungen von TIP1_A_5010-11 entfernt.
- nonQES für eGK wird entfernt da es kein Fachmodul gibt, das mit eGK signiert

A_17768-01 - Zertifikate und Schlüssel für Signaturerstellung und Signaturprüfung (QES und nonQES)
Der Konnektor MUSS bei der Signaturerstellung und Signaturprüfung (QES und nonQES) die Zertifikate und Schlüssel gemäß den Vorgaben in TAB_KON_900 ermitteln.

Tabelle 20: TAB_KON_900 Zertifikate und private Schlüssel für Signaturerstellung und Signaturprüfung (QES und nonQES)

Karte	~~Crypt~~	Zertifikat (Verify)	Schlüssel (Sign)	Einsatzbereich	
				Außen-schnittstelle	Fachmodul-schnittstelle
QES		**...in DF.QES**			
HBA	~~RSA~~	**[ab G2.1]:** EF.C.HP.QES.E256 **[G2.0]:** EF.C.HP.QES.R2048	**[ab G2.1]:** PrK.HP.QES.E256 **[G2.0]:** PrK.HP.QES.R2048	ja	ja
	~~ECC~~	~~EF.C.HP.QES.E256~~	~~PrK.HP.QES.E256~~	~~ja~~	~~ja~~
	~~RSA_ECC~~	~~**[ab G2.1]**: EF.C.HP.QES.E256 **[G2.0]**: EF.C.HP.QES.R2048~~	~~**[ab G2.1]**: PrK.HP.QES.E256 **[G2.0]**: PrK.HP.QES.R2048~~	~~ja~~	~~ja~~
~~HBA-VK~~	~~RSA~~	~~EF.C.HP.QES~~	~~PrK.HP.QES~~	~~ja~~	~~ja~~

Karte	~~Crypt~~	Zertifikat (Verify)	Schlüssel (Sign)	Einsatzbereich	
				Außen-schnittstelle	Fachmodul-schnittstelle
nonQES		**...in DF.ESIGN**			
SM-B	~~RSA~~	**[ab G2.1]:** EF.C.HCI.OSIG.E256 **[G2.0]:** EF.C.HCI.OSIG.R2048	**[ab G2.1]:** PrK.HCI.OSIG.E256 **[G2.0]:** PrK.HCI.OSIG.R2048	ja	ja
	~~ECC~~	~~EF.C.HCI.OSIG.E256~~	~~PrK.HCI.OSIG.E256~~	~~ja~~	~~ja~~
	~~RSA_ECC~~	~~**[ab G2.1]**: EF.C.HCI.OSIG.E256 **[G2.0]**: EF.C.HCI.OSIG.R2048~~	~~**[ab G2.1]**: PrK.HCI.OSIG.E256 **[G2.0]**: PrK.HCI.OSIG.R2048~~	~~ja~~	~~ja~~
~~eGK~~	~~RSA~~	~~EF.C.CH.AUT.E256~~	~~PrK.CH.AUT.E256~~	~~nein~~	~~ja~~
	~~ECC~~	~~EF.C.CH.AUT.E256~~	~~PrK.CH.AUT.E256~~	~~nein~~	~~ja~~
	~~RSA_ECC~~	~~**[ab G2.1]**: EF.C.CH.AUT.E256 **[G2.0]**: EF.C.CH.AUT.R2048~~	~~**[ab G2.1]**: PrK.CH.AUT.E256 **[G2.0]**: PrK.CH.AUT.R2048~~	~~nein~~	~~ja~~

[<=]

In Kapitel 4.1.8.1.1 wird TAB_KON_862-01 gestrichen:

~~Tabelle 21: 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~~	~~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).~~

In Kapitel 4.1.8.1.1 wird TAB_KON_863 gestrichen:

~~**Tabelle 22: TAB_KON_863 Werteliste und Defaultwert des Parameters crypt bei nonQES-Erzeugung**~~

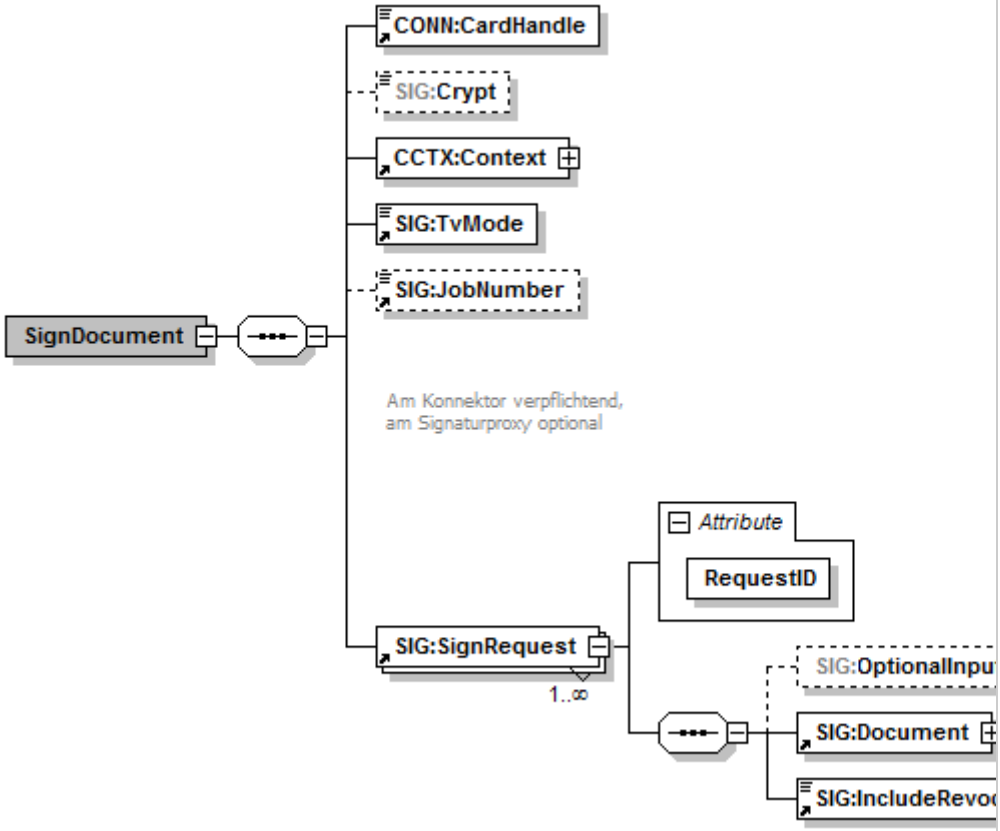
~~**Typname**~~	~~**Werteliste**~~	~~**Defaultwert**~~	~~**Bedeutung**~~
~~SIG_CRYPT_nonQES~~	~~RSA ECC RSA_ECC~~	~~RSA~~	~~Werteliste des Parameters crypt bei der bei der Erzeugung einer nonQES-Signatur RSA: Es wird eine RSA-2048 Signatur erzeugt. ECC: Es wird eine ECC-256 Signatur erzeugt. RSA_ECC: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

In Kapitel 4.1.8.5.1 "SignDocument (nonQES und QES)" wird AFO TIP1_A_5010-10 durch TIP1_A_5010-11 ersetzt:

TIP1-A_5010-11 - Operation SignDocument (nonQES und QES)
Der Signaturdienst des Konnektors MUSS an der Clientschnittstelle eine an [OASIS-DSS] angelehnte Operation SignDocument anbieten**.**

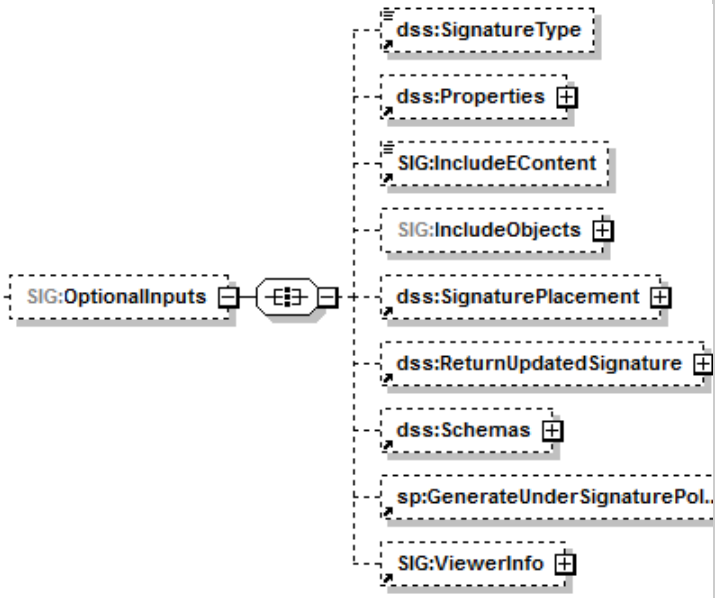
Tabelle 23: TAB_KON_065 Operation SignDocument (nonQES und QES)

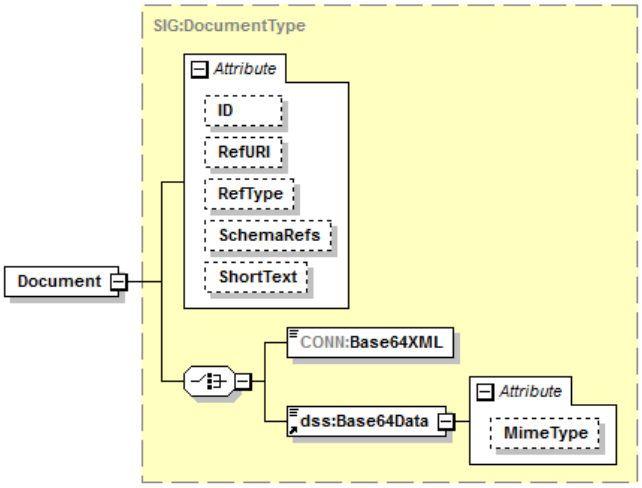
Name	`SignDocument`
Beschreibung	Diese Operation lehnt sich an [OASIS-DSS] an. Sie enthält voneinander unabhängige SignRequests. Jeder SignRequest erzeugt eine Signatur für ein Dokument. Für die qualifizierte elektronische Signatur (QES) werden die `QES_DocFormate` unterstützt. Für nicht-qualifizierte 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.

Name	SignDocument
Aufruf-parameter	

Name	Beschreibung
CONN: CardHandle	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 wird nicht ausgewertet - Optional; Der zu verwendende Schlüssel ist kartenabhängig gemäß Tabelle TAB_KON_900 zu wählen. ~~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.~~

Name	SignDocument	
	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. Enthält der Aufruf mehr als die unterstützte Anzahl von SignRequests, bricht die Operation mit Fehler 4000 ab. Es sind mindestens 50 SignRequests zu unterstützen.

Name	SignDocument	
	SIG: Optional Inputs	Enthält optionale Eingangsparameter (angelehnt an dss:OptionalInputs gemäß [OASIS-DSS] Section 2.7): 

Name	SignDocument	
	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.

Name	SignDocument	
	SIG: Include Revocation Info	Durch diesen verpflichtenden Schalter kann der Aufrufer die Einbettung von Sperrinformationen, die unmittelbar nach dem Zeitpunkt der Signaturerstellung eingeholt wurden, anfordern. Es wird ausschließlich die zu erstellende Signatur betrachtet, d.h. es erfolgt keine Einbettung von Sperrinformationen für bereits enthaltene Signaturen. Für PDF-Signaturen werden keine Sperrinformationen eingebettet.

	dss: Signature Type	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. • **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. Die nachfolgenden Werte KÖNNEN unterstützt werden: • **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

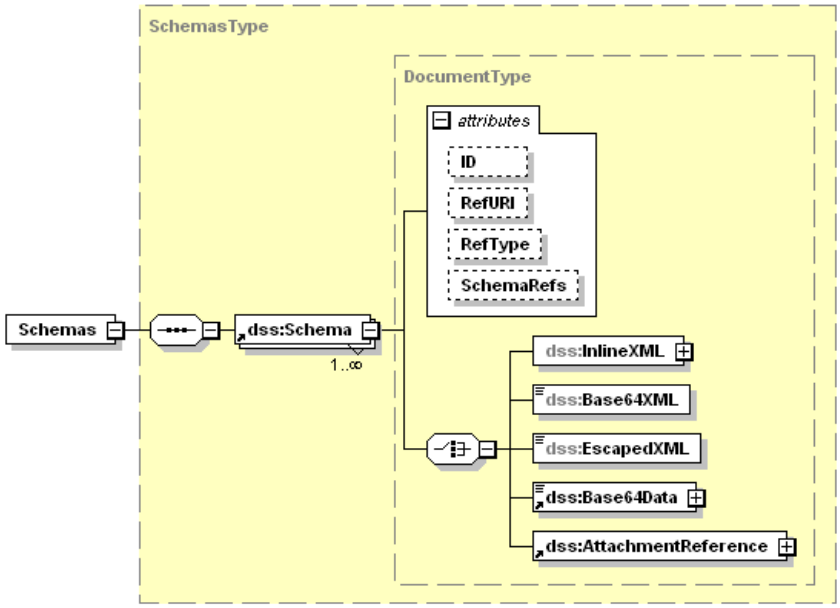
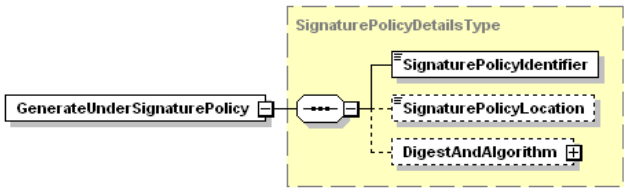
Name	SignDocument	
		(Ungültiger Signaturtyp oder Signaturvariante) zurückgeliefert. Der Konnektor KANN die Operation im Fall S/MIME mit Fehlercode 4279 beenden. Andere SignatureType-Angaben führen zu einer Fehlermeldung 4111 (Ungültiger Signaturtyp oder Signaturvariante). Die Signaturtypen „XML-Signatur, 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.

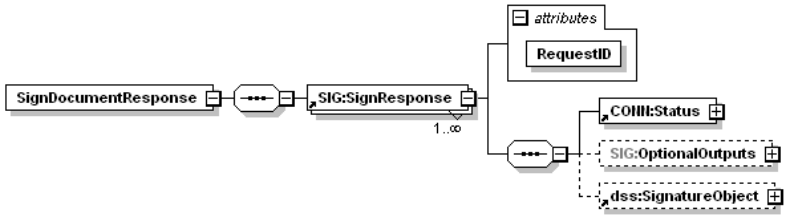
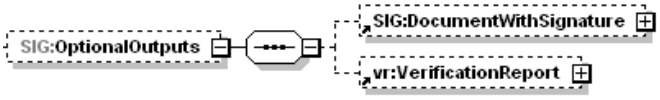
Name	SignDocument	

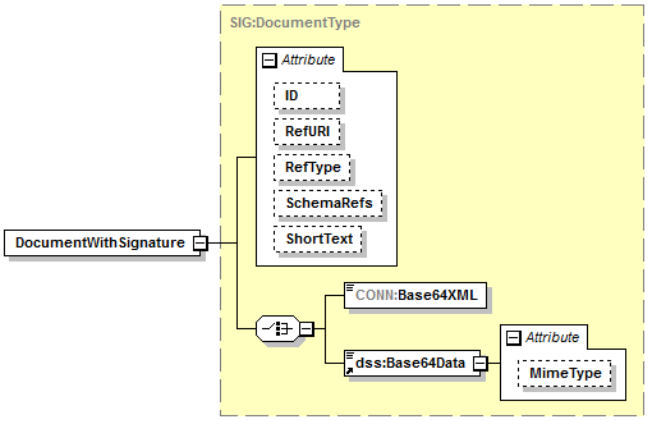
	dss:
	Properties
		Durch dieses in [OASIS-DSS] (Abschnitt 3.5.5) definierte Element können zusätzliche signierte und unsignierte Eigenschaften (Properties) bzw. Attribute in die Signatur eingefügt werden.
		Unterstützt werden genau folgende Attribute:
		Im CMS-Fall (SignatureType = urn:ietf:rfc:5652) 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.
		Die Übergabe der Attribute
		• `ContentType`
		• `SigningTime`
		• `MessageDigest`
		• `SigningCertificate` und `SigningCertificateV2`
		wird ignoriert und es wird die Warnung 4273 zurück gegeben.

	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.

Name	SignDocument	
	dss:SignaturePlacement	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:SignaturePlacement darf nur zusammen mit einer unterstützten Signaturrichtlinie verwendet werden (sp:SignaturePolicyIdentifier muss entsprechend gesetzt sein).
	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/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.

Name	SignDocument	
		
	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: Generate Under Signature Policy	 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.

Name	**SignDocument**	
	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: Optional Outputs	Enthält (angelehnt an dss:OptionalOutputs) optionale Ausgangsparameter: 

Name	SignDocument	
	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 24: 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.
Wenn für die CardSession die Komfortsignatur aktiviert ist (CARDSESSION.SIGNMODE = Comfort) wird Schritt 4 c) ausgeführt. Andernfalls wird Schritt 4 b) ausgeführt.		
4b)	TUC_KON_150 „Dokumente QES signieren“	Die QES wird erzeugt. Tritt hierbei ein Fehler auf, bricht die Operation ab.
4c)	TUC_KON_170 „Dokumente mit Komfort signieren“	Eine Komfortsignatur wird erzeugt. Tritt hierbei ein Fehler auf, bricht die Operation ab.

Nr.	Aufruf Technischer Use Case oder Interne Operation	Beschreibung
5)	TUC_KON_160 „Dokumente nonQES signieren“	Die nonQES wird erzeugt. Tritt hierbei ein Fehler auf, bricht die Operation ab.

Tabelle 25: 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
4273	Technical	Warning	Attribute im Parameter dss:Properties wurden ignoriert
4279	Technical	Error	S/MIME-Funktionalität nicht unterstützt
4283	Technical	Error	Dokument zu groß

Die zulässigen Zertifikate und Schlüssel sind in TAB_KON_900 aufgelistet.
[<=]

2.1.1.10 Kapitel 4.1.9 - Zertifikatsdienst

In Kapitel 4.1.9.6.1 wird A_18931 durch A_18931-01 ersetzt:

A_18931-01 - Anzeige Personalisierungs-Status gSMC-K-X.509-Zertifikate
Der Konnektor MUSS dem Administrator die X.509-Zertifikate der verbauten gSMC-Ks gemäß TIP1-A_4506 anzeigen. Aus der Anzeige MUSS der Personalisierungs-Status der X.509-Zertifikate ersichtlich sein. ~~(dual RSA- und ECC-personalisiert oder nur RSA-personalisiert).~~

gSMC-K Zertifikate, welche im Zuge der Laufzeitverlängerung nicht auf einer Karte, sondern im nichtflüchtigen Speicher des Konnektors persistent abgelegt sind MÜSSEN hier ebenfalls angezeigt werden.**[<=]**

In Kapitel 4.1.9.6.1 wird unter TIP1-A_4506 folgender Freitext ergänzt:

Hinweis: Es existieren Karten, welche lediglich mit RSA-Objekten bzw. lediglich mit ECC-Objekten bestückt sind. Die unbestückten Verzeichnisse können dabei leer sein oder auch gar nicht existieren.

In Kapitel 4.1.9.1 Funktionsmerkmalweite Aspekte wird der Freitext unter A_17295-01 wie folgt angepasst:

- Aspekte zu HBA-VK werden entfernt
- Ein Hinweis, dass es auch singlepersonalisierte Karten sowohl für RSA als auch ECC gibt, wird hinzugefügt

Bei der Zertifikatsprüfung wird ein übergebenes Zertifikat oder ein Zertifikat einer referenzierten Karte geprüft. Das konkrete Zertifikatsobjekt einer Karte ist abhängig vom Kartentyp und dem gewählten kryptographischen Verfahren. Die folgende Tabelle führt auf, welche Zertifikatsobjekte einer Karte in Abhängigkeit vom kryptographischen Verfahren für die jeweilige Zertifikatsreferenz ausgewählt werden.

Hinweis: Es existieren Karten, welche lediglich mit RSA-Objekten bzw. lediglich mit ECC-Objekten bestückt sind. Die unbestückten Verzeichnisse können dabei leer sein oder auch gar nicht existieren.

Tabelle 26: TAB_KON_858 Kartenobjekt in Abhängigkeit vom kryptographischen Verfahren

CertRef	Kartentyp	Objekt der Karte in Abhängigkeit vom kryptographischen Verfahren (Crypt)	
		RSA	ECC
C.AUT	~~HBA-VK~~	~~EF.C.HP.AUT~~	-
	HBA	EF.C.HP.AUT.R2048	EF.C.HP.AUT.E256
	SM-B	EF.C.HCI.AUT	EF.C.HCI.AUT.E256
	eGK G2	EF.C.CH.AUT.R2048	EF.C.CH.AUT.E256
C.ENC	~~HBA-VK~~	~~EF.C.HP.ENC~~	-
	HBA	EF.C.HP.ENC.2048	EF.C.HP.ENC.E256
	SM-B	EF.C.HCI.ENC.R2048	EF.C.HCI.ENC.E256
C.SIG	SM-B	EF.C.HCI.OSIG.R2048	EF.C.HCI.OSIG.E256

CertRef	Kartentyp	Objekt der Karte in Abhängigkeit vom kryptographischen Verfahren (Crypt)	
		RSA	ECC
C.QES	~~HBA-VK~~	~~EF.C.HP.QES~~	~~-~~
	HBA	EF.C.HP.QES.R2048	EF.C.HP.QES.E256

In Kapitel 4.1.9.4.3 "TUC_KON_034 „Zertifikatsinformationen extrahieren"" wird TIP1-A_4697 durch TIP1-A_4697-02 ersetzt:

TIP1-A_4697-02 - TUC_KON_034 „Zertifikatsinformationen extrahieren"
Der Konnektor MUSS den technischen Use Case TUC_KON_034 „Zertifikatsinformationen extrahieren" umsetzen.

Tabelle 27: TAB_KON_770 TUC_KON_034 „Zertifikatsinformationen extrahieren"

Element	Beschreibung
Name	TUC_KON_034 „Zertifikatsinformationen extrahieren"
Beschreibung	Dieser TUC beschreibt die Extraktion der fachlich zentralen Informationen aus bestimmten Zertifikaten einer gesteckten Karte eines Mandanten.
Auslöser	• Aufruf durch ein Fachmodul oder eine Basisanwendung des Konnektors
Vorbedingungen	Keine
Eingangsdaten	• cardSession – *optional/verpflichtend für den Zugriff auf eGK, HBA, SM-B oder gSMC-KT* • checkSMCK [Boolean] – *optional/verpflichtend für gSMC-K;* (Referenz auf eine/die gSMC-K, alternativ zu cardSession) • qes [Boolean] - *optional; default: false* – (Angabe, ob die QES-Identität oder die nonQES-Identität der Karte interessiert) • crypt - *optional; default = ~~RSA~~ECC* (kryptographischer Algorithmus, für welchen das Zertifikat ermittelt wird; Wertebereich: ECC, RSA)
Komponenten	Konnektor

Element	Beschreibung
Ausgangsdaten	• certType [C.CH.AUT | C.HP.AUT | C.HCI.AUT | C.HP.QES] (Zertifikatstyp) • certInfo (Zertifikatsinformationen, bestehend aus SerialNumber, Issuer, Subject, Rollen, registrationNumber und ggf. id-etsi-qcs-QcCompliance, siehe Standardablauf) • qcStatments – *optional/nur wenn certType = C.HP.QES* (QCStatements)
Nachbedingungen	Keine

Element	Beschreibung
Standardablauf	1. Je nach Kartentyp wird aus der Karte das passende Zertifikat über TUC_KON_216 "LeseZertifikat" {selektiertes Zertifikat} ausgelesen. Das Zertifikatsobjekt (fileIdentifier und folder)/Zertifikatsbezeichnung wird für die jeweilige Karte unter Berücksichtigung des kryptographischen Verfahrens crypt gemäß TAB_KON_858 bzw. TAB_KON_856 ermittelt. a. Bei qes = false: i. Für die eGK: C.CH.AUT ii. Für den HBAx: C.HP.AUT iii. Für SM-B: C.HCI.AUT iv. Für gSMC-K: C.NK.VPN b. Bei qes = true: i. Für den HBAx: C.HP.QES 2. Die Zertifikatsbezeichnung aus Schritt 1 („C.XXX.YYY.ZZZZ“) wird als Ausgangsdatum „certType“ zurückgegeben. 3. Zusätzlich werden aus dem Zertifikat folgende Informationen extrahiert und zurückgegeben: a. X509SerialNumber b. Issuer (DistinguishedName) nach RFC 2253 c. Subject (DistinguishedName) nach RFC 2253 d. Aus der Extension Admission: i. eine Liste von Rollen durch Aufruf von TUC_PKI_009 „Rollenermittlung” ii. registrationNumber (=Telematik-ID; falls vorhanden) e. id-etsi-qcs-QcCompliance (0.4.0.1862.1.1) in QCStatements, falls vorhanden f. Restriction, falls vorhanden g. validity
Varianten/Alternativen	Keine

Element	Beschreibung
Fehlerfälle	(→1) Wenn im Aufrufkontext (also erreichbar durch den Mandanten) zum angegebenen CardHandle keine Karte gefunden werden kann, bricht der TUC mit Fehlercode 4146 ab. (→1b) Ist bei Angabe von QES=true auf der Karte keine QES-Identität zu finden, bricht der TUC mit Fehlercode 4147 ab. Für die Kombination QES=true mit einer eGK bricht der TUC mit Fehlercode 4148 ab (QES-Zertifikate der eGK werden noch nicht unterstützt). (→1) Für eGK, HBA, SM-B gilt: Wenn crypt=ECC und Karte vom Typ <G2.1, bricht der TUC mit Warnung 4257 ab. (→1) Für gSMC-K gilt: Wenn crypt=ECC und TUC_KON_216 Warnung 4256 liefert, bricht der TUC mit Warnung 4257 ab. (→1) Wenn aus anderen Gründen die Extraktion der Zertifikatsinformationen fehlschlägt, bricht der TUC mit Fehlercode 4148 ab.
Nichtfunktionale Anforderungen	keine
Zugehörige Diagramme	keine

Tabelle 28: TAB_KON_602 Fehlercodes TUC_KON_034 „Zertifikatsinformationen extrahieren"

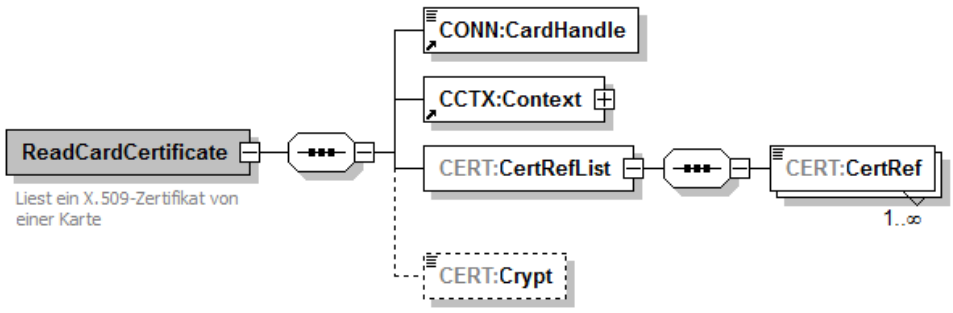
Fehlercode	ErrorType	Severity	Fehlertext
Neben den Fehlercodes der aufgerufenen technischen Use Cases können folgende weitere Fehlercodes auftreten:			
4146	Technical	Error	Kartenhandle existiert nicht
4147	Technical	Error	Zertifikat nicht vorhanden (z. B. kein QES-Zertifikat in SM-B)
4148	Technical	Error	Fehler beim Extrahieren von Zertifikatsinformationen
4257	Technical	Warning	<$Crypt>~~ECC~~-Zertifikate nicht vorhanden auf Karte: <cardHandle>

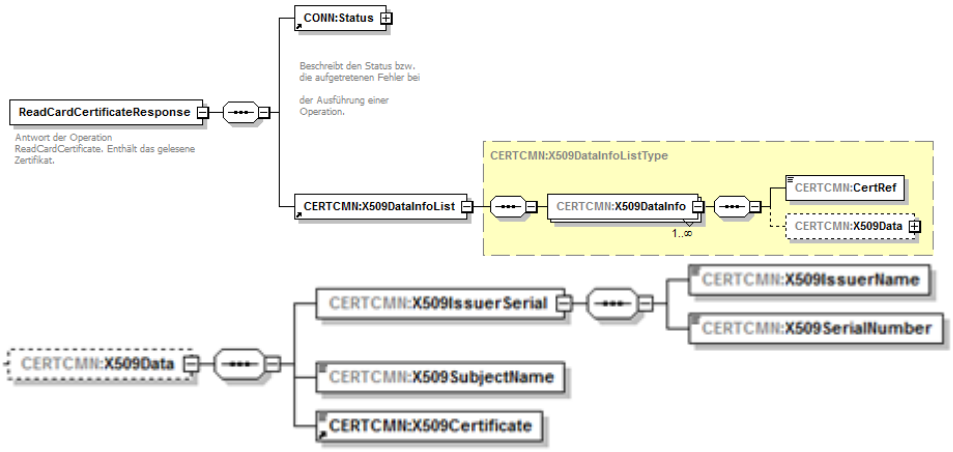
[<=]

In Kapitel 4.1.9.5.2 "ReadCardCertificate" wird Anforderung TIP1-A_4700 durch TIP1-A_4700-02 ersetzt:

TIP1-A_4700-02 - Operation ReadCardCertificate
Die Basisanwendung Zertifikatsdienst des Konnektors MUSS an der Clientschnittstelle eine Operation ReadCardCertificate wie in Tabelle TAB_KON_678 Operation ReadCardCertificate beschrieben anbieten.

Tabelle 29: TAB_KON_678 Operation ReadCardCertificate

<table>
<tr><td>Name</td><td colspan="2">ReadCardCertificate</td></tr>
<tr><td>Beschreibung</td><td colspan="2">Liest X.509-Zertifikate von einer Karte.</td></tr>
<tr><td>Aufruf-parameter</td><td colspan="2"> 
</td></tr>
<tr><td></td><td>Name</td><td>Beschreibung</td></tr>
<tr><td></td><td>CardHandle</td><td>Gibt die Karte an, von der das Zertifikat gelesen werden soll.
Es können Zertifikate von HBAx (HBA, HBA-VK), SM-B ausgelesen werden.
Die Operation ReadCardCertificate DARF das Lesen von Zertifikaten der eGK NICHT unterstützen.</td></tr>
<tr><td></td><td>Context</td><td>Aufrufkontext (Mandant)</td></tr>
</table>

Name	**ReadCardCertificate**	
	CertRefList	Gibt an, welche(s) Zertifikat(e) gelesen werden soll. Mögliche Werte für CertRef sind: C.AUT, C.ENC, C.SIG, C.QES
	Crypt	Optional; Default: ~~RSA~~ECC Gibt den kryptographischen Algorithmus vor, für den das Zertifikat ermittelt werden soll. Wertebereich: RSA, ECC • RSA: Zertifikat für RSA-2048 • ECC: Zertifikat für ECC-256
Rückgabe		
	Status	Enthält den Ausführungsstatus der Operation.
	CertRef	Dieses Element beinhaltet die Referenz des Zertifikats, welches bei der Anfrage übergeben wurde.
	X509Data	Inhalt des über die CertRef referenzierten Zertifikats. Ist das referenzierte Zertifikat nicht vorhanden, so wird dieses Element nicht vom Konnektor gefüllt.

Name	ReadCardCertificate		
		X509IssuerName	Enthält den Issuer-Name des Zertifikats. Bezüglich des Encodings sind die in [XMLDSig#4.4.4.4.1] angegebenen Regeln zu beachten (Escaping von Sonderzeichen etc.)
		X509SerialNumber	Enthält die serialNumber des Zertifikats.
		X509SubjectName	Enthält das Feld subject.CommonName. Bezüglich des Encodings sind die in [XML DSig#4.4.4.4.1] angegebenen Regeln zu beachten (Escaping von Sonderzeichen etc.)
		X509Certificate	Enthält das base64-codierte Zertifikat, dessen Binärstruktur wiederum ASN.1-codiert (gemäß [COMMON_PKI]) vorliegt.
Vorbedingungen	Keine		
Nachbedingungen	Keine		

Der Ablauf der Operation ReadCardCertificate ist in Tabelle TAB_KON_679 Ablauf ReadCardCertificate beschrieben:

Tabelle 30: TAB_KON_679 Ablauf ReadCardCertificate

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. Wurde als Zielkarte eine eGK adressiert, wird Fehlercode 4090 zurückgeliefert.

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 }

4.	getEF	Für jedes Paar von CertRef und CardHandle wird in Abhängigkeit des Parameters Crypt gemäß Tabelle TAB_KON_858 das zu lesende File (EF) bestimmt: Ist die übergebene Zertifikatsreferenz ungültig, wird Fehlercode 4149 zurückgegeben. Das Lesen von Zertifikaten der eGK ist aus Sicherheitsgründen für Clientsysteme nicht zulässig.
	TUC_KON_216 „LeseZertifikat“	Für jedes Paar von CardHandle und EF wird nun durch Aufruf von TUC_KON_216 „LeseZertifikat“ das Zertifikat ausgelesen. Falls TUC_KON_216 die Warnung 4256 zurückgibt, wird die Operation abgebrochen und Fehler 4258 zurückgegeben.
6.	Zertifikatsattribute extrahieren	Aus jedem Zertifikat werden die zu liefernden Attribute extrahiert. Die Ergebnisstruktur wird mit den erhaltenen Rückgabewerten gefüllt.

Tabelle 31: TAB_KON_604 Fehlercodes „ReadCardCertificate“

Fehlercode	ErrorType	Severity	Fehlertext
Neben den Fehlercodes der aufgerufenen technischen Use Cases können folgende weitere Fehlercodes auftreten:			
4000	Technical	Error	Syntaxfehler
4149	Technical	Error	Ungültige Zertifikatsreferenz
4090	Security	Error	Zugriff auf eGK nicht gestattet
4258	Technical	Error	<$Crypt>~~ECC~~Zertifikate nicht vorhanden auf Karte: <cardHandle>

[<=]

In Kapitel 4.1.9 wird Anforderung TIP_A_4691 durch TIP_A_4691-01 ersetzt:

Bei Karten, die ECC-Zertifikate tragen, wird nur noch das ECC-Zertifikat regelmäßig geprüft, da auch nur noch dieses verwendet wird.

TIP1-A_4691-01 - Ablauf der gSMC-K und der gesteckten Karten regelmäßig prüfen
Für die gSMC-K sowie für jede gesteckte Karte außer eGK MUSS der Konnektor im Intervall `CERT_EXPIRATION_CARD_CHECK_DAYS` genau einmal TUC_KON_033 aufrufen.
Der Konnektor MUSS die Gültigkeitsdauer der Zertifikate prüfen mittels Aufruf von:
für gSMC-K
TUC_KON_033{checkSMCK; doInformClients=Ja; crypt = ECC}
wenn kein ECC-Zertifikat personalisiert ist :
TUC_KON_033{checkSMCK; doInformClients=Ja; crypt = RSA}
für jede gesteckte G2.0 Karte außer eGK und außer gSMC-K
TUC_KON_033{cardSession; doInformClients=Ja; crypt = RSA}
für jede gesteckte ab G2.1 Karte außer eGK
TUC_KON_033{cardSession; doInformClients=Ja; crypt = ECC}
~~TUC_KON_033{cardSession; doInformClients=Ja; crypt = RSA}~~
[<=]

In Kapitel 4.1.9 wird Anforderung TIP_A_4695 durch TIP_A_4695-03 ersetzt:

TIP1-A_4695-03 - TUC_KON_033 „Zertifikatsablauf prüfen“
Der Konnektor MUSS den technischen Use Case TUC_KON_033 „Zertifikatsablauf prüfen” umsetzen.

Tabelle 32: TAB_KON_768 TUC_KON_033 „Zertifikatsablauf prüfen“

Element	Beschreibung
Name	TUC_KON_033 „Zertifikatsablauf prüfen“
Beschreibung	Dieser TUC prüft und meldet das zeitliche Ablaufen eines X.509-Zertifikats einer Karte.

Element	Beschreibung
Auslöser	• Aufruf durch andere TUCs des Konnektors oder • über die Managementschnittstelle
Vorbedingungen	Keine
Eingangsdaten	• cardSession – *optional*/für eGK, HBA, SM-B, gSMC-KT • checkSMCK [Boolean] – *optional/für gSMC-K;* (Referenz auf eine/die gSMC-K, alternativ zu cardSession) • doInformClients [Boolean] (Angabe, ob ein Event an die Clients gesendet werden soll) • crypt - *optional; default* = *ECC ~~RSA~~* (kryptographischer Algorithmus, für welchen das Zertifikat ermittelt wird; Wertebereich: ECC, RSA)
Komponenten	Konnektor
Ausgangsdaten	• expirationDate (Ablaufdatum des untersuchten Zertifikats)

Standardablauf	1. TUC_KON_216 „LeseZertifikat“ für:
	• Bei checkSMCK das Zertifikat der gSMC-K (C.NK.VPN) gemäß TAB_KON_856
	• bei CardSession die Zertifikate der identifizierten Karte.
	a. Für die eGK: C.CH.AUT
	b. Für den HBAx: C.HP.AUT
	c. Für SM-B: C.HCI.AUT
	• Das konkrete Zertifikatsobjekt der Karte gemäß TAB_KON_858 wird vom Eingangsparameter `crypt` abgeleitet.
	2. Das Ablaufdatum *expirationDate* wird aus dem Feld validity ausgelesen.
	3. Falls das Zertifikat abgelaufen ist, Systemereignis absetzen:
	• gSMC-K:
	TUC_KON_256 {
	topic = „CERT/CARD/EXPIRATION”;
	eventType = Op;
	severity = Warning;
	parameters = („CARD_TYPE=gSMC-K,
	ICCSN=$ICCSN,
	Konnektor=$MGM_KONN_HOSTNAME,
	ZertName=$Name des Zertifikatsobjekts gemäß TAB_KON_856,
	ExpirationDate=$validity");
	doLog = true;
	doDisp = $doInformClients }
	• Sonstige Karten (mit CARD(CardSession)):
	TUC_KON_256 {
	topic = „CERT/CARD/EXPIRATION”;
	eventType = Op;
	severity = Warning;
	parameters = („CARD_TYPE=$Type,
	ICCSN=$ICCSN,
	CARD_HANDLE=$CardHandle,
	CardHolderName=$CardHolderName,
	ZertName=$Name des Zertifikatsobjekts aus Schritt 1,
	ExpirationDate=$validity“);
	doLog=false;
	doDisp = $doInformClients }
	4. Alternativ bei Ablauf des Zertifikats innerhalb von CERT_EXPIRATION_WARN_DAYS Systemereignis absetzen:
	• gSMC-K:
	TUC_KON_256 {
	topic = „CERT/CARD/EXPIRATION”;
	eventType = Op;
	severity = Info;
	parameters = („CARD_TYPE=gSMC-K,
	ICCSN=$ICCSN,

Element	Beschreibung

	Konnektor=$MGM_KONN_HOSTNAME,
	ZertName=$Name des Zertifikatsobjekts gemäß TAB_KON_856,
	ExpirationDate=$validity,
	DAYS_LEFT=$validity-$Today“);
	doLog = false;
	doDisp = $doInformClients}
	• Sonstige mit CARD(CardSession)):
	TUC_KON_256 {
	topic = „CERT/CARD/EXPIRATION”;
	eventType = Op;
	severity = Info;
	parameters = („CARD_TYPE=$Type,
	ICCSN = $ICCSN,
	CARD_HANDLE = $CardHandle,
	CardHolderName = $CardHolderName,
	ZertName=$Name des Zertifikatsobjekts aus Schritt 1,
	ExpirationDate = $validity,
	DAYS_LEFT = $validity-$Today“);
	doLog = false;
	doSisp = $doInformClients}
	5. expirationDate wird zurückgegeben.

Varianten/ Alternativen	Keine
Fehlerfälle	(→1) Zur angegebenen CardSession keine Karte gefunden: Fehlercode 4131. (→1) Für eGK, HBA, SM-B gilt: Wenn crypt=ECC und Kartengeneration<G2.1, bricht der TUC mit Warnung 4257 ab. (→1) Für gSMC-K gilt: Wenn crypt=ECC und beim Aufruf von TUC_KON_216 wird die Warnung 4256 zurückgegeben, dann wird der TUC nach Schritt 1 beendet und die Warnung 4257 an den Aufrufer zurückgegeben. (→2) Extraktion des Ablaufsdatums fehlgeschlagen: Fehlercode 4132.
Nichtfunktionale Anforderungen	Keine
Zugehörige Diagramme	Keine

Tabelle 33: TAB_KON_600 Fehlercodes TUC_KON_033 „Zertifikatsablauf prüfen“

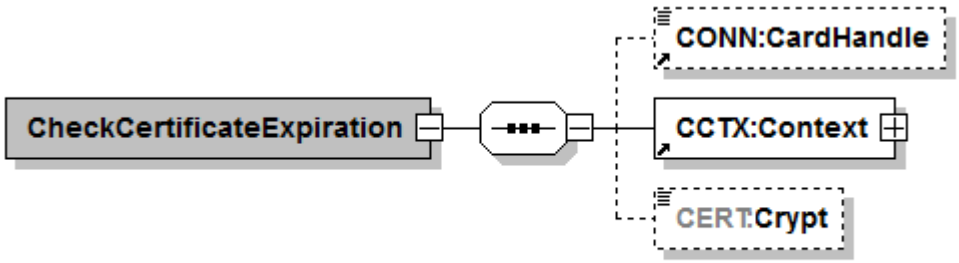
Fehlercode	ErrorType	Severity	Fehlertext
Neben den Fehlercodes der aufgerufenen technischen Use Cases können folgende weitere Fehlercodes auftreten:			
4131	Technical	Error	Zum angegebenen CardHandle keine Karte gefunden.
4132	Security	Error	Extraktion des Ablaufsdatums fehlgeschlagen
4257	Technical	Warning	ECC-Zertifikate nicht vorhanden auf Karte: <cardHandle>

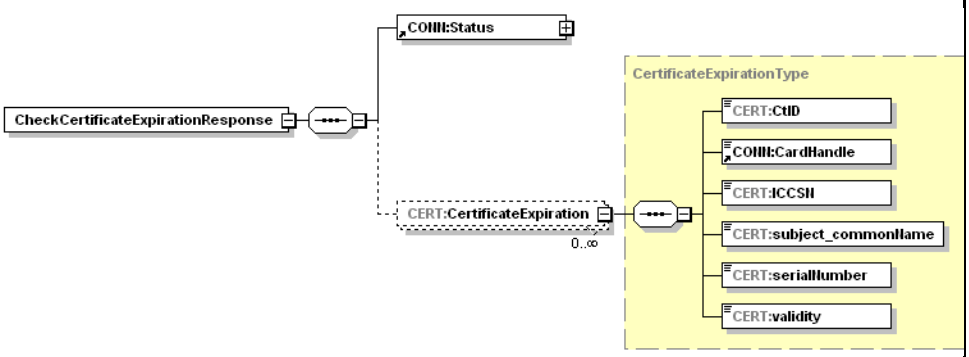
[<=]

In Kapitel 4.1.9 wird Anforderung TIP_A_4699-04 durch TIP_A_4699-05 ersetzt:

TIP1-A_4699-05 - Operation CheckCertificateExpiration
Die Basisanwendung Zertifikatsdienst des Konnektors MUSS an der Clientschnittstelle eine Operation CheckCertificateExpiration anbieten.

Tabelle 34: TAB_KON_676 Operation CheckCertificateExpiration

Name	`CheckCertificateExpiration`	
Beschreibung	Gibt das Datum des Ablaufs eines bestimmten Zertifikats oder gesammelt des Zertifikats der gSMC-K, der gSMC-KT's sowie aller gesteckten HBAx und SM-B des Mandanten zurück.	
Aufruf-parameter		
	Name	**Beschreibung**

Name	CheckCertificateExpiration	
	CardHandle	Optional. Identifiziert die Karte, deren Zertifikate geprüft werden sollen. Wird der Parameter nicht angegeben, so werden alle für den Konnektor erreichbaren Karten (inkl. gSMC-K und aller gSMC-KT's), die zum Mandanten passen, berücksichtigt. Die Operation CheckCertificateExpiration DARF das Lesen von Zertifikaten der eGK NICHT unterstützen.
	Context	MandantId, CsId, WorkplaceId verpflichtend; UserId optional
	Crypt	Optional; Default: ECC ~~RSA~~ Gibt den kryptopraphischen Algorithmus vor, für den das Zertifikat ermittelt werden soll. Wertebereich: RSA, ECC • RSA: Zertifikat für RSA-2048 • ECC: Zertifikat für ECC-256
Rückgabe		
	Status	Enthält den Ausführungsstatus der Operation.
	CertificateExpiration	Eine Liste von Tupeln aus (CtID, CardHandle, ICCSN, subject.CommonName, serialNumber, validity) der Zertifikate der Karten. Für die gSMC-K soll in CertificateExpiration/CtID und CertificateExpiration/CardHandle jeweils ein Leerstring zurückgegeben werden.
Vorbedingungen	Keine	
Nachbedingungen	Keine	

Der Ablauf der Operation CheckCertificateExpiration ist in Tabelle TAB_KON_677 beschrieben:

Tabelle 35: TAB_KON_677 Ablauf CheckCertificateExpiration

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. Wird die Operation für einen nicht unterstützten Kartentypen aufgerufen, so bricht die Operation mit Fehler 4058 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;
		allWorkplaces=true, wenn cardHandle nicht angegeben, ansonsten false
		}
		Tritt bei der Prüfung ein Fehler auf, bricht die Operation mit Fehlercode aus TUC_KON_000 ab.

3.	enumerateCardHandles	Wenn der Parameter CardHandle übergeben wurde, wird dieser als einziges Element in eine Liste gepackt. Wenn der Parameter CardHandle leer war, wird eine Liste der CardHandles aller für den Konnektor erreichbaren Karten (inkl. gSMC-K und gSMC-KT's), die zum Mandanten passen, erstellt.
Für jedes CardHandle der in Schritt 3 erzeugten Liste werden folgende Schritte ausgeführt, für die gSMC-Ks die Schritte 5 und 6: Falls Schritt 5 der TUC_KON_033 die Warnung 4257 zurückgibt, wird Schritt 6 nicht ausgeführt und die Schritte für das CardHandle der in Schritt 3 erzeugten Liste weiter ausgeführt. Die Warnung 4257 wird über alle CardHandle akkumuliert und <komma-separierte List von cardHandle> für den Fehlertext erzeugt.		

4.	TUC_KON_026 „Liefere CardSession“	Ermittle CardSession über TUC_KON_026 {
		mandatId =MandantId;
		clientSystemId = ClientSystemId;
		cardHandle = CardHandle;
		userId = UserId }

Nr.	Aufruf Technischer Use Case oder Interne Operation	Beschreibung

5.	TUC_KON_033 „Zertifikatsablauf prüfen“	Das Gültigkeitsdatum des Zertifikats wird geprüft mit
		TUC_KON_033 {
		cardSession;
		doInformClients = false;
		Crypt;
		}
		bzw. TUC_KON_033 {
		checkSMCK = true;
		doInformClients = false;
		Crypt;
		}

6.	TUC_KON_034 „Zertifikatsinformationen extrahieren”	Beim Aufruf des TUC_KON_034 ist der Parameter qes = false zu setzen. Aus den jeweiligen Rückgabewerten entsteht eine Liste aus Tupeln (CtID, CardHandle, ICCSN, subject.CommonName, serialNumber, validity). Diese wird von der Operation zurückgegeben.

Tabelle 36: TAB_KON_603 Fehlercodes „CheckCertificateExpiration“

Fehlercode	ErrorType	Severity	Fehlertext
Neben den Fehlercodes der aufgerufenen technischen Use Cases können folgende weitere Fehlercodes auftreten:			
4000	Technical	Error	Syntaxfehler
4058	Security	Error	Aufruf nicht zulässig
4257	Technical	Warning	<$Crypt>~~ECC~~-Zertifikate nicht vorhanden auf Karte: <komma-separierte List von cardHandle>

[<=]

In Kapitel 4.1.9 wird Anforderung TIP_A_4701-03 durch TIP_A_4701-04 ersetzt:

TIP1-A_4701-04 - TUC_KON_035 „Zertifikatsdienst initialisieren“
In der Bootup-Phase MUSS der Konnektor den Zertifikatsdienst durch Aufruf des TUC_KON_035 „Zertifikatsdienst initialisieren“ initialisieren.

Tabelle 37: TAB_KON_772 TUC_KON_035 „Zertifikatsdienst initialisieren“

Element	Beschreibung
Name	TUC_KON_035 „Zertifikatsdienst initialisieren“
Beschreibung	Der TUC beschreibt den gesamten Ablauf der Initialisierung des TrustStore im Rahmen der betrieblichen Prozesse: Prüfung der Aktualität, Integrität und Authentizität der Einträge im TrustStore.
Auslöser	• Bootup des Konnektors
Vorbedingungen	keine
Eingangsdaten	keine
Komponenten	Konnektor
Ausgangsdaten	• Status der Initialisierung des TrustStore
Nachbedingungen	Keine

Element	Beschreibung
Standardablauf	Für den übergebenen Status der Initialisierung des TrustStore werden folgende Schritte durchgeführt: 1. Durch eine DNS-Anfrage an den DNS-Forwarder zur Auflösung der SRV-RR mit dem Bezeichner "_ocsp._tcp.<DOMAIN_SRVZONE_TI>„ erhält der Konnektor Adressen des http-Forwarders des VPN-Zugangsdienststandortes. 2. Falls in den letzten 24 Stunden keine Aktualisierung der TSL und CRL im Truststore stattgefunden hat, aktualisiert der Konnektor die TSL durch den Aufruf von TUC_KON_032 „TSL aktualisieren“ und die CRL durch den Aufruf von TUC_KON_040 „CRL aktualisieren“. 3. Falls im Zeitraum von CERT_BNETZA_VL_UPDATE_INTERVAL keine Aktualisierung der BNetzA VL stattgefunden hat, aktualisiert der Konnektor die BNetzA VL durch den Aufruf von TUC_KON_031 „BNetzA-VL aktualisieren“. 4. Der Konnektor prüft die Gültigkeitsdauer der Zertifikate aller gesteckten Karten entsprechendTIP1-A_4691* ~~(inkl. gSMC-K) mittels Aufruf von:~~ für gSMC-K ~~TUC_KON_033{checkSMCK; doInformClients=Ja; crypt = ECC}~~ ~~TUC_KON_033{checkSMCK; doInformClients=Ja; crypt = RSA}~~ für jede gesteckte G2.0 Karte ~~TUC_KON_033{cardSession; doInformClients=Ja; crypt = RSA}~~ ~~für jede gesteckte ab G2.1 Karte~~ ~~TUC_KON_033{cardSession; doInformClients=Ja; crypt = ECC}~~ ~~TUC_KON_033{cardSession; doInformClients=Ja; crypt = RSA}~~ 5. Der Konnektor liest von der gSMC-K den öffentlichen Schlüssel des CVC-Root-Zertifikats und speichert diesen im TrustStore [gemSpec_gSMC-K_ObjSys#5.3.10].
Varianten/ Alternativen	Keine
Fehlerfälle	Keine
Nichtfunktionale Anforderungen	Keine
Zugehörige Diagramme	Keine

Tabelle 38: TAB_KON_605 Fehlercodes TUC_KON_035 „Zertifikatsdienst initialisieren“

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

[<=]

In Kapitel 4.1.9 wird Anforderung TIP_A_4704 durch TIP_A_4704-01 ersetzt:

TIP1-A_4704-01 - Zertifikatsablauf anzeigen
Der Administrator MUSS einen Prüflauf auf den innerhalb von CERT_EXPIRATION_WARN_DAYS Tagen bevorstehenden Ablauf von Zertifikaten aller für den Konnektor erreichbaren Karten (inkl. gSMC-K) an zentraler Stelle in der Managementschnittstelle auslösen können und das Ergebnis angezeigt bekommen.
Der Konnektor MUSS die Gültigkeitsdauer der Zertifikate aller gesteckten Karten gemäß TIP1-A_4691* prüfen ~~(inkl. gSMC-K) prüfen mittels Aufruf von:~~
~~für gSMC-K~~
~~TUC_KON_033{checkSMCK; doInformClients=Nein; crypt = ECC}~~
~~TUC_KON_033{checkSMCK; doInformClients=Nein; crypt = RSA}~~
~~für jede gesteckte G2.0 Karte außer gSMC-K~~
~~TUC_KON_033{cardSession; doInformClients=Nein; crypt = RSA}~~
~~für jede gesteckte ab G2.1 Karte außer gSMC-K~~
~~TUC_KON_033{cardSession; doInformClients=Nein; crypt = ECC}~~
~~TUC_KON_033{cardSession; doInformClients=Nein; crypt = RSA}~~
[<=]

2.1.1.11 Kapitel 4.1.11 - TLS-Dienst

In Kapitel 4.1.11 wird Anforderung TIP_A_4720-02 durch TIP_A_4720-03 ersetzt:

TIP1-A_4720-03 - TUC_KON_110 „Kartenbasierte TLS-Verbindung aufbauen“
Der Konnektor MUSS den technischen Use Case "Kartenbasierte TLS-Verbindung aufbauen" gemäß TUC_KON_110 umsetzen.

Tabelle 39: TAB_KON_773 – TUC_KON_110 „Kartenbasierte TLS-Verbindung aufbauen“

Element	Beschreibung
Name	TUC_KON_110 „Kartenbasierte TLS-Verbindung aufbauen“
Beschreibung	Der TUC_KON_110 „Kartenbasierte TLS-Verbindung aufbauen“ baut eine TLS-Verbindung zur angegebenen Zieladresse auf. Dabei kann für eine gegenseitige Authentisierung eine SM-B verwendet werden.
Auslöser	Aufruf durch ein Fachmodul
Vorbedingungen	Die für die Authentisierung adressierte Karte muss freigeschaltet sein

Element	Beschreibung
Eingangsdaten	• roleToMatch – *optional/verpflichtend, wenn Rollenprüfung durchgeführt werden soll* • cardSession – *optional/verpflichtend, wenn Clientauthentisierung durchgeführt werden soll* (CardSession einer SM-B) • targetUri (URI des Verbindungsziels)
Komponenten	Konnektor, eHealth-Kartenterminal, Karte, Server des Fachdienstes
Ausgangsdaten	• tlsConnectiontId (ConnectionIdentifier der TLS-Verbindung)

Standardablauf	Der Konnektor MUSS folgende Schritte durchlaufen:
	1. Auflösen des FQDN im targetUri per 'TUC_KON_361 „DNS Namen auflösen"
	2. TLS-Verbindung mit ermittelter Adresse aufbauen:
	Wähle die im Client-Hello präsentierten Ciphersuiten abhängig von dem verfügbaren Schlüsselmaterial. Wenn auf der über cardSession referenzierten Karte ein ECC-Zertifikat vorhanden ist, dürfen nur ECC-basierte Ciphersuiten angeboten werden.
	a) Prüfe Server-Zertifikat mittels TUC_KON_037 „Zertifikat prüfen" {
	certificate = C.FD.TLS-S;
	qualifiedCheck = not_required;
	offlineAllowNoCheck = true;
	policyList = oid_fd_tls_s;
	intendedKeyUsage= intendedKeyUsage(C.FD.TLS-S);
	intendedExtendedKeyUsage = id-kp-serverAuth;
	validationMode = OCSP}
	Das Server-Zertifikat MUSS C.FD.TLS-S sein
	b) Prüfe in a) zurückgegebene Rolle („ermittelte Rolle") == roleToMatch
	c) Wenn cardSession übergeben: Clientauthentisierung mittels ID.HCI.AUT. Der Konnektor darf nur dann das ECC-Zertifikat von der SMC-B auswählen, wenn auch das Serverzertifikat ein ECC-Zertifikat ist.
	3. tlsConnectiontId der erzeugten Verbindung zurückgeben

Varianten/ Alternativen	• Keine

Element	Beschreibung
Fehlerfälle	Fehler in den folgenden Schritten des Ablaufs führen zu einem Abbruch der Verarbeitung mit den ausgewiesenen Fehlercodes (→1) Der Name der Gegenstelle kann nicht aufgelöst werden (→2b) Rollenprüfung fehlgeschlagen: Fehlercode 4220 (→2) Server konnte nicht authentisiert werden: Fehlercode 4156 (→2) Clientauthentisierung fehlgeschlagen: Fehlercode 4157
Nichtfunktionale Anforderungen	keine
Zugehörige Diagramme	keine

Tabelle 40: TAB_KON_612 Fehlercodes TUC_KON_110 „Kartenbasierte TLS-Verbindung aufbauen"

Fehlercode	ErrorType	Severity	Fehlertext
Neben den Fehlercodes der aufgerufenen technischen Use Cases können folgende weitere Fehlercodes auftreten:			
4156	Security	Error	Server konnte bei TLS-Verbindungsaufbau nicht authentisiert werden
4157	Security	Error	Clientauthentisierung bei TLS-Verbindungsaufbau fehlgeschlagen
4220	Security	Error	Rollenprüfung bei TLS-Verbindungsaufbau fehlgeschlagen

[<=]

2.1.1.12 Kapitel 4.1.12 - LDAP-Proxy

In Kapitel 4.1.12 wird Anforderung TIP_A_5517-02 durch TIP_A_5517-03 ersetzt:

TIP1-A_5517-03 - Konnektor, TUC_KON_290 „LDAP-Verbindung aufbauen"
Der Konnektor MUSS den technischen Use Case TUC_KON_290 „LDAP-Verbindung aufbauen" gemäß TAB_KON_805 umsetzen.

Tabelle 41: TAB_KON_805 - TUC_KON_290 „LDAP-Verbindung aufbauen"

Element	Beschreibung
Name	TUC_KON_290 „LDAP-Verbindung aufbauen"
Beschreibung	Initiiert durch einen Verbindungsaufbau des LDAP-Clients zum Konnektor baut der Konnektor eine TLS-gesicherte Verbindung zum VZD auf.

Element	Beschreibung
Auslöser	LDAP (oder LDAPS wenn ANCL_TLS_MANDATORY=Enabled) Verbindungsaufbau von einem Fachmodul oder einem Clientsystem ist abgeschlossen. Bei Verwendung von LDAPS authentisiert sich der Konnektor beim LDAP-Client mit dem gemäß Kapitel 3.5 konfigurierten Serverzertifikat ~~der Identität ID.AK.AUT~~.
Vorbedingungen	• MGM_LU_ONLINE=Enabled
Eingangsdaten	keine
Komponenten	Konnektor, VZD
Ausgangsdaten	keine
Standardablauf	1. Der Konnektor ermittelt den FQDN und Port des VZD durch eine DNS-SD Namensauflösung gemäß [RFC6763] mit dem Bezeichner "_ldap._tcp.vzd.<DNS_TOP_LEVEL_DOMAIN_TI>." 2. Der Konnektor baut eine LDAPS-Verbindung zum VZD auf. Dabei wird das Serverzertifikat des Verzeichnisdienst C.ZD.TLS-S nach TUC_PKI_018 geprüft (PolicyList: oid_zd_tls_s (gemäß gemSpec_OID), intendedKeyUsage: intendedKeyUsage(C.ZD.TLS-S), ExtendedKeyUsages: serverAuth (1.3.6.1.5.5.7.3.1), Offlinemodus: nein, TOLERATE_OCSP_FAILURE: false , Prüfmodus: OCSP)
Varianten/Alternativen	keine
Fehlerfälle	
Nichtfunktionale Anforderungen	keine
Zugehörige Diagramme	keine

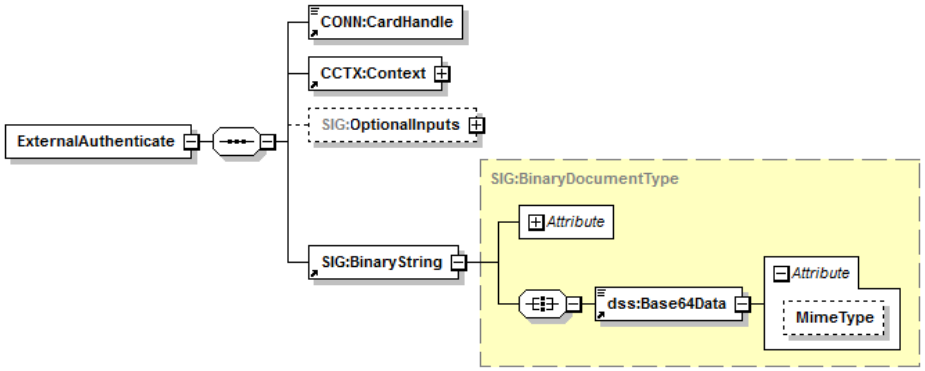
[<=]

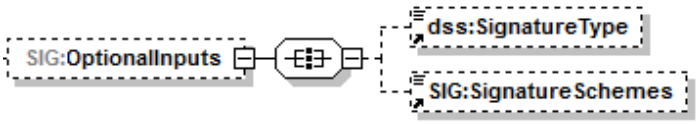
2.1.1.13 Kapitel 4.1.13 - Authentifizierungsdienst

In Kapitel 4.1.13.4.1 wird Anforderung TIP1-A_5439 durch TIP1-A_5439-02 ersetzt:

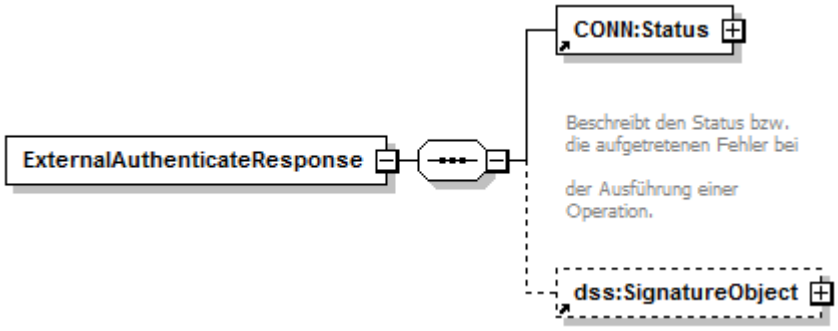
TIP1-A_5439-02 - Operation ExternalAuthenticate
Der Authentifizierungsdienst des Konnektors MUSS an der Clientschnittstelle eine Operation ExternalAuthenticate anbieten**.**

Tabelle 42: TAB_KON_781 Operation ExternalAuthenticate

<table>
<tr><td>Name</td><td colspan="2">ExternalAuthenticate</td></tr>
<tr><td>Beschreibung</td><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. Das AUT-Zertifikat der SM-B und das AUT-Zertifikat des HBAx werden unterstützt.</td></tr>
<tr><td>Aufrufparameter</td><td colspan="2"> 

<table>
<tr><td>Name</td><td>Beschreibung</td></tr>
<tr><td>CONN:
CardHandle</td><td>Identifiziert die zu verwendende Signaturkarte.
Die Operation unterstützt HBAx und SM-B.</td></tr>
<tr><td>CCTX:
Context</td><td>Aufrufkontext für HBAx:
MandantId, ClientSystemId, WorkplaceId, UserId verpflichtend
Aufrufkontext für SM-B:
MandantId, ClientSystemId, WorkplaceId verpflichtend; UserId nicht ausgewertet</td></tr>
<tr><td>SIG:
Optional
Inputs</td><td>Enthält optionale Eingangsparameter:
 
</td></tr>
</table></td></tr>
</table>

Name	ExternalAuthenticate	
	SIG: Binary String	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. Für G2.1-Karten (und höher) werden folgende Längen unterstützt: • 256 Bit: SHA-256 (OID 2.16.840.1.101.3.4.2.1 Für G2.0-Karten 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 beginnt der Konnektor die Ausführung der Methode EMSA-PKCS1-v1_5-ENCODE nach [RFC3447], Abschnitt 9.2, mit Schritt 2, Erstellung des DigestInfo-Datenfeldes. • Im Falle des Signaturverfahrens RSASSA-PSS beginnt der Konnektor die Ausführung der Methode EMSA-PSS-ENCODE nach [RFC3447], Abschnitt 9.1.1, mit Schritt 3. • 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

Name	ExternalAuthenticate	
	dss: SignatureType	Der Übergabeparameter wird nicht ausgewertet, statt dessen wird das Signaturverfahren über die Kartenversion bestimmt. ~~Durch dieses in [OASIS-DSS] (Abschnitt 3.5.1) beschriebene Element wird der Typ der zu erzeugenden Signatur bestimmt. Als Signaturtyp wird unterstützt :~~ • Für G2.0-Karten: **PKCS#1-Signatur** ~~Durch Übergabe der URI~~ urn:ietf:rfc:3447 ~~wird eine~~ Es wird eine PKCS#1 (Version 2.1) Signatur gemäß [RFC3447] erzeugt, die als dss:Base64Signature mit ~~der oben genannten~~ URI urn:ietf:rfc:3447 zurückgeliefert wird. • Für G2.1-Karten, HSM-B und höher: **ECDSA-Signatur** ~~Durch Übergabe der URI~~ urn:bsi:tr:03111:ecdsa ~~wird~~ Es wird eine ECDSA-Signatur gemäß [BSI-TR-03111]#4.2.1 erzeugt, die als dss:Base64Signature mit der oben genannten URI urn:bsi:tr:03111:ecdsa zurückgeliefert wird. ~~Andere SignatureType Angaben führen zu einer Fehlermeldung 4111 (Ungültiger Signaturtyp oder Signaturvariante).~~
	SIG: Signature Schemes	Für G2.1-Karten (und höher) wird dieses Element nicht ausgewertet. Für G2.0-Karten: 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		
	CONN: Status	Enthält den Status der ausgeführten Operation.

Name	ExternalAuthenticate	
	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. Die Signatur wird binär gemäß [BSI-TR-03111]#5.2.2 in der ASN.1 Struktur ECDSA-Sig-Value zurückgegeben. 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	

Der Ablauf der Operation ExternalAuthenticate ist in Tabelle TAB_KON_782 beschrieben:

Tabelle 43: TAB_KON_782 Ablauf Operation ExternalAuthenticate

Nr.	Aufruf Technischer Use Case oder Interne Operation	Beschreibung
1.	checkArguments	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 {$context.mandantId; $context.clientsystemId; $context.workplaceId; $context.userId; $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 { MandantId, CsId, CardHandle, UserId }

Nr.	Aufruf Technischer Use Case oder Interne Operation	Beschreibung

4.	TUC_KON_218 „Signiere“	Signaturberechnung durch Aufruf des TUC_KON_218 {
		PinRef = PIN.CH bzw. PIN.SMC;
		KeyRef = PrK.HP.AUT bzw. PrK.HCI.AUT;
		AlgorithmusID = signPKCS1_V1_5 oder signPSS oder signECDSA;
		DTBS = Binärstring
		}

Tabelle 44: TAB_KON_783 Übersicht Fehler Operation ExternalAuthenticate

Fehlercode	ErrorType	Severity	Fehlertext
Neben den Fehlercodes der aufgerufenen TUCs können folgende weitere Fehlercodes auftreten:			
4000	Technical	Error	Syntaxfehler
4058	Security	Error	Aufruf nicht zulässig

Die folgende Tabelle führt die zulässigen privaten Schlüssel für die Operation ExternalAuthenticate auf:

Tabelle 45: TAB_KON_784 Privater Schlüssel je Karte für ExternalAuthenticate

Karte	Schlüssel
SM-B	PrK.HCI.AUT in DF.ESIGN
HBAx	PrK.HP.AUT in DF.ESIGN

[<=]

2.1.1.14 Kapitel 4.2.4 - VPN-Client

2.1.1.15 Kapitel 4.3 - Konnektormanagement

~~In Kapitel 4.3.7 wird A_23120 "Beschränkung der Anfragen zur Re-Registrierung (ECC-Migration)~~

~~"gestrichen.~~

In Kapitel 4.3.7 wird A_23122 "Konfigurationsschalter für automatische Re-Registrierung (ECC-Migration)

"gestrichen.

TIP1-A_4825-01 wird durch TIP1-A_4825-02 ersetzt:

TIP1-A_4825-02 - Konnektor zur Nutzung (wiederholt) freischalten
Der Administrator MUSS den Konnektor über folgenden Mechanismus zur Nutzung freischalten bzw. eine vorhandene Freischaltung mit einer neuen SM-B aktualisieren können (Voraussetzung ist eine korrekte Konfiguration aller für einen Online-Zugang erforderlicher Parameter):

1. Der Konnektor MUSS eine registerKonnektorRequest-Struktur gemäß ProvisioningService.xsd [gemSpec_VPN_ZugD] erstellen und mit den entsprechenden Parametern befüllen (aktuelles Datum/Uhrzeit, C.NK.VPN (wenn vorhanden MUSS dass ECC-Zertifikat verwendet werden), MGM_ ZGDP_CONTRACTID). Der Konnektor MUSS die Request-Nachricht mittels der ausgewählten SM-B (ID.HCI.OSIG von MGM_ZGDP_SMCB; Bei SMC-B ab G2.1 muss das ECC-Zertifikat verwendet werden, sonst das RSA-Zertifikat) im Element registerKonnektorRequest/Signature signieren und das SM-B-Zertifikat im Element X509Data ablegen.
   Ist der nötige Sicherheitszustand für den privaten Schlüssel der SM-B nicht gesetzt, MUSS der Konnektor zur PIN-Verifikation an dem Kartenterminal auffordern, in dem die SM-B steckt.
2. Der Konnektor ermittelt die URI des Registrierungsservers (MGM_ZGDP_REGSERVER) durch eine DNS-Anfrage nach dem SRV und TXT Resource Record „_regserver._tcp.<DNS_DOMAIN_VPN_ZUGD_INT>„
3. Der Konnektor ruft unter Verwendung der erzeugten Request-Nachricht die in [gemSpec_VPN_ZugD#Tab_ZD_registerKonnektor] definierte Operation I_Registration_Service::registerKonnektor mit der Zieladresse MGM_ZGDP_REGSERVER auf.
4. Der Konnektor zeigt dem Administrator den Inhalt von registerKonnektorResponse/AdditionalInformation und /Status an
5. Der Response der Operation wird verarbeitet:
   a. Setze MGM_TI_ACCESS_GRANTED auf
      - Enabled, wenn /RegistrationStatus = „Registriert“
      - Disabled, wenn /RegistrationStatus = „Nicht registriert“
   b. Persistiere diese Zustandsinformation zusammen mit dem VPN:ContractStatus
   c. Verteile das folgende interne Ereignis über TUC_KON_256 {
      topic = ”MGM/TI_ACCESS_GRANTED“;
      eventType = Op;
      severity = Info;
      parameters = „Active=$MGM_TI_ACCESS_GRANTED“;
      doDisp = false }

Tritt während der Verarbeitungskette ein Fehler auf, so bricht die weitere Verarbeitung ab und der Administrator MUSS darüber geeignet informiert werden (u.a. Klartextanzeige des vom Registrierungsdienst gemeldeten Fehlers).
Wenn eine Reregistrierung mit einer neuen SMC-B fehlschlägt (Request wird mit einem SOAP-Error beantwortet) dann ist der Konnektor nicht registriert (MGM_TI_ACCESS_GRANTED = Disabled).
[<=]

In Kapitel 4.3.7 wird A_23150 durch A_23150-01 ersetzt:

A_23150-01 - Konnektor (wiederholt) manuell registrieren
Der Konnektor MUSS dem Administrator über den Mechanismus in TUC_KON_411 ermöglichen, den Konnektor zu registrieren bzw. eine vorhandene Registrierung zu aktualisieren. Dabei MUSS der Administrator das für die Registrierung zu verwendende NK-Zertifikat (RSA oder ECC) und die zu verwendende SMC-B auswählen können. Sofern dem Konnektor ein ECC-basiertes NK-Zertifikat zur Verfügung steht, DARF ein RSA-basiertes NK-Zertifikat hierbei NICHT zur Auswahl stehen.
[<=]

In Kapitel 4.3.8 wird A_23149 "Konfigurationsschalter für Verwendung von ECC bei TLS (ECC-Migration)" entfernt

In Kapitel 4.3.8 wird A_21758-06 durch A_21758-07 ersetzt.

- Die Vorbedingung "Laufzeitverlängerung: Keine" wird gestrichen, da es dem TUC nur um die Durchführung der Registrierung geht. Es gibt keine übergreifende Regelung, die fordern würde, dass eine Neuregistrierung mit ECC nach eine LZV nicht durchgeführt werden darf. Falls es solche Regelungen geben wird, dann werden diese im Zuge der LZV (Zertifikatserstellung/Herausgabe) umgesetzt

A_21758-07 - TUC_KON_411 „Konnektor mit neuem NK-Zertifikat registrieren“
Der Konnektor MUSS den technischen Use Case TUC_KON_411 "Konnektor mit neuem NK-Zertifikat registrieren" umsetzen.

Tabelle 46: TAB_KON_932 – TUC_KON_411 „Konnektor mit neuem NK-Zertifikat registrieren“

Element	Beschreibung
Name	TUC_KON_411 "Konnektor mit neuem NK-Zertifikat registrieren"
Beschreibung	Dieser TUC führt eine Neuregistrierung mit einem neuen (ECC) NK-Zertifikat durch.
Auslöser	A_22332, A_21745, Administrator
Vorbedingungen	• ECC-Migration: Die gSMC-K ist gemäß A_18928 dual-personalisiert. • ~~Laufzeitverlängerung: Keine~~
Eingangsdaten	Keine
Komponenten	Konnektor, VPN-ZugD
Ausgangsdaten	Keine

Element	Beschreibung

Standardablauf	1. Der Konnektor ermittelt die URI des Registrierungsservers (MGM_ZGDP_REGSERVER) durch eine DNS-Anfrage nach dem SRV und TXT Resource Record „_regserver._tcp.<DNS_DOMAIN_VPN_ZUGD_INT>".
	2. Der Konnektor MUSS eine registerKonnektorRequest-Struktur gemäß ProvisioningService.xsd [gemSpec_VPN_ZugD] erstellen und mit den entsprechenden Parametern befüllen (aktuelles Datum/Uhrzeit, neues C.NK.VPN-Zertifikat, MGM_ ZGDP_CONTRACTID). Der Konnektor MUSS die Request-Nachricht mittels einer verfügbaren SM-B (ID.HCI.OSIG) im Element registerKonnektorRequest/Signature signieren und das SM-B-Zertifikat im Element X509Data ablegen. (Bei SMC-B ab G2.1 muss das ECC-Zertifikat verwendet werden, sonst das RSA-Zertifikat. MGM_ ZGDP_SMCB ist zu bevorzugen, es kann aber auch eine andere SM-B verwendet werden).
	3. Der Konnektor ruft unter Verwendung der erzeugten Request-Nachricht die in [gemSpec_VPN_ZugD#Tab_ZD_registerKonnektor] definierte Operation I_Registration_Service::registerKonnektor mit der Zieladresse MGM_ZGDP_REGSERVER auf.
	Der Response der Operation wird verarbeitet:
	a. Setze MGM_TI_ACCESS_GRANTED auf
	- Enabled, wenn /RegistrationStatus = „Registriert"
	- Disabled, wenn /RegistrationStatus = „Nicht registriert"
	b. Persistiere diese Zustandsinformation zusammen mit dem VPN:ContractStatus
	c. Verteile das folgende Ereignis über TUC_KON_256
	{
	topic = "MGM/TI_ACCESS_GRANTED";
	eventType = Op;
	severity = Info;
	parameters = „Active=$MGM_TI_ACCESS_GRANTED";
	doLog = true;
	doDisp = true }

Varianten/Alternativen	Manuelle Registrierung: (->2) Der Administrator soll die zu verwendende SM-B auswählen können.

Element	Beschreibung

Fehlerfälle	(→ 2) Es konnte keine freigeschaltete SM-B ausgewählt werden:
	Fail=No_Smcb
	(->2,3) Im Fehlerfall
	TUC_KON_256 {
	topic = „SMC_K/REGISTER/ERROR“;
	eventType = Op;
	severity = Error;
	parameters = „$Parameters“;
	doLog = true;
	doDisp = true }
	Die Registrierung soll herstellerspezifisch erneut mehrmals versucht werden.
	Bei allen Fehlerfällen, die zum Abbruch führen:
	TUC_KON_256 {
	topic = „SMC_K/REGISTER/ERROR“;
	eventType = Op;
	severity = Error;
	parameters = „$Parameters“;
	doLog = true;
	doDisp = true }

Nichtfunktionale Anforderungen	Keine
Zugehörige Diagramme	Keine

Tabelle 47: Tab_Kon_933 Fehlercodes TUC_KON_411 "Konnektor mit neuem NK-Zertifikat registrieren"

Fehlercode	ErrorType	Severity	Fehlertext
Neben den Fehlercodes der aufgerufenen technischen Use Cases können folgende weitere Fehlercodes auftreten:			
herstellerspezifisch			

[<=]

Aus Kapitel 4.3.8 der gemSpec_Kon wird A_21781 entfernt.

Für A_21781-01 werden die Zuordnungen zu Prüfverfahren des Konnektors entfernt.

2.1.1.16 Änderungen in 5.5 Weitere Dokumente

[Quelle]	Herausgeber (Erscheinungsdatum): Titel
[ALGCAT]	SOG-IS Crypto Evaluation Scheme - Agreed Cryptographic Mechanisms, ~~Version 1.2, Januar 2020.~~ Version 1.3, Februar 2023. https://www.sogis.eu/documents/cc/crypto/SOGIS-Agreed-Cryptographic-Mechanisms-1.3.pdf

2.1.2 Änderungen an api-telematik

2.1.2.1 operatingData.xsd

Die Schnittstellenbeschreibung aus [api-telematik] wird gemäß Pull Request https://github.com/gematik/api-telematik/pull/21 angepasst.

Die darin enthaltenen Änderungen müssen vom Konnektor umgesetzt werden.

2.1.3 Änderungen in gemSpec_FM_VSDM

In Kapitel 7.2 wird Anforderung A_25723 unter A_23158 neu eingefügt:

A_25723 - FM_VSDM: Zertifikatsauswahl
Das Fachmodul VSDM MUSS das AUT-Zertifikat der eGK abhängig von der Kartengeneration wählen.

- eGK G2.0: Das Fachmodul greift auf das RSA-Zertifikat zu
- ab eGK G2.1: Das Fachmodul greift auf das ECC-Zertifikat zu. Ein Fehlen des RSA-Zertifikats darf nicht zu einem Fehler führen.

[<=]

Funktionale Eignung: Test

2.1.4 Änderungen in gemSpec_FM_NFDM

Bei der Anwendung NFDM ist kein Zertifikatszugriff spezifiziert. Dennoch muss die Funktionsfähigkeit mit ECC-only-Karten abgesichert werden: neue Anforderung

A_25877 - FM_NFDM Funktion bei nicht personalisierten RSA-Zertifikaten
Das Fachmodul NFDM MUSS seine Operationen mit einer G2.1 eGK fehlerfrei ausführen, auch wenn auf dieser keine RSA-Zertifikate personalisiert sind.**[<=]**

Funktionale Eignung: Test

2.1.5 Änderungen in gemSpec_FM_AMTS

neue Anforderung

A_25878 - FM_AMTS Verwendung von ECC-Zertifikaten bei G2.1eGK
Das Fachmodul AMTS MUSS bei einer G2.1 eGK auf das ECC-Zertifikat von C.CH.AUT zugreifen. Ein fehlendes oder ungültiges RSA-Zertifikat darf nicht zu einem Fehler führen.**[<=]**

Funktionale Eignung: Test

2.1.6 Änderungen in gemSpec_Krypt

In Kapitel 3.14 wird Anforderung GS-A_5207 durch GS-A_5207-01 ersetzt (Aufnahme von ECDSA Signatur für ECC Zertifikate analog A_17090-01):

GS-A_5207-01 - Signaturverfahren beim initialen Pairing zwischen Konnektor und eHealth-Kartenterminal
Alle Produkttypen ~~, die~~ MÜSSEN in Bezug auf das verwendete Signaturverfahren beim initialen Pairing zwischen Konnektor und eHealth-Kartenterminal folgende Vorgaben umsetzen:

1. Falls für die aktuelle TLS-Verbindung mit dem Konnektor eine RSA-basierte Ciphersuite verwendet wird, so MUSS für die Signatur des Shared-Secret (ShS.AUT.KT vgl. [gemSpec_KT#2.5.2.1, 3.7.2.1])~~erzeugen oder prüfen, und~~ RSASSA-PSS [PKCS#1] und SHA-256 verwendet werden.
2. ~~auf Basis von RSA die TLS-Verbindung betreiben, die für das aktuell durchzuführende Pairing notwendig ist,~~ Falls für die aktuelle TLS-Verbindung mit dem Konnektor eine ECDSA-basierte Ciphersuite verwendet wird, so MUSS für die Signatur des Shared-Secret ECDSA [BSI-TR-03111] und SHA-256 verwendet werden

~~MÜSSEN für die Signatur des Shared-Secret und dessen Signaturprüfung RSASSA-PSS [PKCS#1] verwenden.~~
[<=]

Anforderung GS-A_5208 und nachfolgender Freitext wird gelöscht (galt nur für PTV3) und durch A_17209 ersetzt.

GS-A_5208 - Signaturverfahren für externe Authentisierung
Der Konnektor MUSS an der Schnittstelle für die externe Authentisierung die Signaturverfahren RSASSA-PKCS1-v1_5 [PKCS#1] und RSASSA-PSS [PKCS#1] anbieten.**[<=]**

~~Erläuterung: Der Konnektor erlaubt (bei entsprechender Berechtigung) die direkte Nutzung der privaten Schlüssel MF/ DF.ESIGN/ PrK.HP.AUT.* auf einem HBA oder MF/ DF.ESIGN/ PrK.HCI.AUT.* auf einer SMC-B durch ein Primärsystem. Dies wird fast immer für eine clientenseitige TLS-Authentisierung gegenüber einem TLS-Server (außerhalb der TI) verwendet. Dafür werden über die Schnittstelle RSASSA-PKCS1-v1_5- Signaturen von den entsprechenden Karten erzeugt und über den Konnektor an ein Primärsystem übergeben. Für unbenannte Anwendungen müssen auch RSASSA-PSS-Signaturen erzeugbar sein. Diese Signaturen sind nicht als Dokumentensignaturen verwendbar, der Verwendungszweck ist in den zu den privaten Schlüsseln gehörigen Zertifikaten kodiert (ExtendedKeyUsage: keyPurposeId = id-kp-clientAuth).~~

~~Hinweis: GS-A_5208 ist nicht dem PTV4-Konnektor zugewiesen, sondern die erweiterte Anforderungen ☐ML-92911 - Missing cross-reference .~~

A_17209 - Signaturverfahren für externe Authentisierung (ECC-Migration)
Der Konnektor MUSS an der Schnittstelle für die externe Authentisierung die Signaturverfahren RSASSA-PKCS1-v1_5 [PKCS#1], RSASSA-PSS [PKCS#1] und ECDSA [BSI-TR-03111] anbieten.**[<=]**

Die Zuordnung von A_22458 - TLS-Algorithmus passend zum Pairing wird vom PTV6 entfernt

A_22458 - TLS-Algorithmus passend zum Pairing
Der Konnektor MUSS beim TLS-Verbindungsaufbau (TLS-Handshake) zu einem eHealth-Kartenterminal ausschließlich Ciphersuiten mit solchen Authentisierungsalgorithmen (entweder RSA oder ECDSA) anbieten, die zum Algorithmus des gespeicherten KT-Zertifikats passen, wenn zu diesem Kartenterminal bereits Pairinginformationen (Shared Secret in Kombination mit dem Zertifikat des Kartenterminals) gespeichert sind.**[<=]**

A_17210 wird aus der Spezifikation entfernt:

(Alle Aktiven ZugD sind auf PTV1.8.7 und unterstützen damit A_17125)

A_17210 - Konnektor, IKE-Schlüsselaushandlung Fallback (ECC-Migration)
Ein Konnektor MUSS, falls beim IKE-Verbindungsaufbau klar wird, dass der IKE-Responder (VPN-Konzentrator, VPN-Zugangsdienst) (noch) keine ECC-Verfahren unterstützt (INVALID_KE_PAYLOAD-Nachricht), auf die Vorgaben aus GS-A_4382-* "zurückfallen".
[<=]

2.1.7 Änderungen in gemSpec_PKI

Der Freitext unter A_17820 wird wie folgt geändert (A_17784 ist nicht mehr in der Spec/Steckbrief des Konnektors:

Hinweis: Die Nutzung von Cross-Zertifikaten für den Wechsel des Vertrauensraums ist für den Konnektor besonders geregelt (s. gemSpec_Kon#A_17837-* ~~und A_17784-*~~).

2.1.8 Änderungen in gemILF_PS

Alle folgenden Änderungen für gemILF_PS betreffend:

Textabschnitte, welche eine bestimmte PTV betreffen und im Format "*<PTVx> </PTVx>*" beschrieben sind werden entfernt und durch die Formulierung passend zur jeweils gültigen PTV ersetzt. Der PS-Hersteller muss sich daher aus den jeweils gültigen PTV-Steckbriefen die jeweils normativ gültige Version des gemILF_PS heraussuchen und entsprechend Konformität sicherstellen.

Neues Kapitel

4.1.1.6.3 Schlüsselqualität der Clientzertifikate

Der Konnektor überwacht, ob die Qualität der Clientzertifikate den Mindestanforderungen entspricht und setzt diese bei der Erzeugung oder dem Import von Clientzertifikaten durch. Wenn durch Updates oder die Übernahme bestehender Konfigurationen Zertifikate mit RSA-2048-Schlüsseln konfiguriert sind, so wird dieses durch den Betriebszustand EC_TLS_Client_Certificate_Security (Sec; Info) angezeigt und kann über das Event OPERATIONAL_STATE/EC_TLS_Client_Certificate_Security dem Clientsystem

gemeldet werden. Um den Betriebszustand zu beseitigen müssen neue Clientzertifikate entsprechend TAB_KON_866 konfiguriert werden.

Neues Kapitel

4.2.1.6 Information zu ungültigen Karten

Aktuelle Konnektorversionen führen nach dem Stecken eines HBA oder einer SMC-B eine Zertifikatsprüfung durch und senden im Fehlerfall das Event CERT/CARD/STATUS an Clientsysteme mit passender Subscription. Dadurch kann der Anwender sofort informiert werden, wenn eine gesteckte Karte noch nicht vollständig in Betrieb genommen wurde (CERTSTATUS=unknown) oder widerrufen wurde (CERTSTATUS=revoked).

A_25850 - Anzeige ungültiger LE-Karten
Das Primärsystem MUSS das Event CERT/CARD/STATUS abonnieren und den Anwender über das Stecken einer ungültigen Karte informieren**[<=]**

Da Karten der Generation 2.0 nicht die ab 01.01.2026 vorgeschriebenen Schlüssel enthalten prüft der Konnektor nach dem Stecken eines HBA oder einer SMC-B die Kartengeneration und erzeugt für jede G2 Karten einen Betriebszustand EC_G2_CARD_USED($pseudonym) mit den Parametern Kartentyp und Ablaufdatum. In den Logfiles des Konnektors findet sich auch die ICCSN und der CARDHOLDER zu diesem Pseudonym.

A_25851 - Information zu genutzten G2 karten
Das Primärsystem MUSS die Betriebszustände EC_G2_HBA_USED
und EC_G2_SMCB_USED auswerten oder die zugehörigen Events abonnieren und den Anwender über die Notwendigkeit des Kartentausches für diese Karten informieren.**[<=]**

InKapitel 4.4 "<PTV2> Signaturerstellung und Verschlüsselung":

- Die Überschrift des Kapitels wird in "~~<PTV2>~~ Signaturerstellung und Verschlüsselung" geändert
- Inhaltlich wird folgendes geändert:
  - HBA-Vorläuferkarten sind nicht mehr im Feld vorhanden. Daher können die entsprechende Einträge gelöscht werden.
  - Verweise auf Schnittstellenbeschreibungen sowie Beispiele verweisen nun auf GitHub

Der Konnektor stellt generische Schnittstellen für QES-Basisdienste zur Verfügung (`SignatureService`, `EncryptionService`, `CertificateService`, `AuthSignatureService`),~~sowie Schnittstellen für die tokenbasierte Authentisierung~~.
Diese Schnitt-stellen können vom Primärsystem in einer Vielzahl von Szenarien genutzt werden:

- Signatur und Signaturprüfung mit Identitäten von SMC-B~~,~~ oder HBA ~~und HBA-Vorläuferkarten~~;
- Ver- und Entschlüsselung von Dokumenten und Daten mit SMC-B~~,~~ oder HBA ~~und HBA-Vorläuferkarten~~;
- Authentisierung mit SMC-B~~,~~ oder HBA ~~und HBA-Vorläuferkarten~~;

- Smartcard-Zertifikatsabfragen und Prüfung von Zertifikaten.

Beispiel-Dateien für die Nutzung der Signaturschnittstelle am Konnektor sind über~~das Fachportal der gematik im Kontext der Schemadateien der Signaturschnittstelle zugänglich.~~ folgende Ressourcen zugänglich:

- https://github.com/gematik/api-telematik (siehe PTV Steckbriefe für die jeweils konkret verbindliche Version)
- https://github.com/gematik/examples-TelematikInterfaces/tree/master (insbesondere Unterordner ./conn/SignatureService)

...

~~<PTV4>~~Um die Interoperabilität der Dokumentenvalidierung zu gewährleisten, muss für das Format PDF/A die PDF/A-ID als XML-Element in den Metadaten des Dokuments stehen, etwa in der Form

```
<pdfaid:part
xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/">1</pdfaid:part>
```

~~</PTV4>~~

~~<PTV4>~~Nach der Einführung von elliptischen Kurven auf TI-Smartcards der Generation G2.1und dem bevorstehendem Ende der Nutzbarkeit der RSA-Zertifikate werden vom Konnektor bevorzugt ECC-Zertifikate verwendet. Spezifisches dazu ist für Verschlüsselung in [gemSpec_Kon#TIP1-A_4621-*] bzw. für Signatur in [gemSpec_Kon# TIP1_A_5010-*] geregelt.

Das Default-Verhalten an der Konnektorschnittstelle hängt von dessen PTV des ab:

* vor PTV6 werden RSA-Schlüssel verwendet

* ab PTV6 werden ECC-Schlüssel verwendet. Bei G2.0-Smartcard wird auf RSA Schlüssel zurückgefallen.

~~ist so beschaffen, dass ohne explizite Steuerung der Optionen RSA oder ECC durch das PS der Konnektor unter Auswertung der verfügbaren Karten nach Möglichkeit ECC-die geeigneten Zertifikate auswählt.~~

Vor PTV6 kann das PS kann das Default-Verhalten des Konnektors durch Nutzung des Parameters CRYPT übersteuern, wenn ein geeignetes ECC-Zertifikat auf der verwendeten TI-Smartcard vorhanden ist. Um dies a-priori herauszufinden, ist das PS ~~Wenn ein PS das Default-Verhalten des Konnektors durch Nutzung der Auswahloption übersteuern möchte, ist es~~ darauf angewiesen, den Typ der verwendeten Karte zu ermitteln. Im Rückgabewert von getCards ist an der VersionInfo in CARD:CardVersion/CARD:ObjektSystemVersion erkennbar, ob eine Karte der Generation G2.1 oder höher mit einem ECC-Zertifikat vorliegt. Jede Smartcard mit einer Objektsystemversion >= 4.4.0 (Major.Minor.Revision-Versionsnummer) enthält ECC-Zertifikate.Ab PTV6 wird der Parameter CRYPT nicht mehr ausgewertet~~</PTV4>~~

~~An PTV3-Konnektoren werden auch bei Karten der Generation G2.1 deren RSA-Zertifikate verwendet.~~

In Kapitel 4.4.1 "Erstellen digitaler Signaturen" wird der Freitext unter A_13524 wie folgt geändert:

- Vermerke zur HBA-VK werden entfernt
- Tabelle 17 wird entfernt. Es wird stattdessen auf SignDocument in gemSpec_Kon verwiesen, weil dort bereits alle Kryptoverfahren in Abhängigkeit der Karte beschrieben sind.

...

~~In Bezug auf die QES-Stapelsignatur unterscheiden sich HBAs von HBA-Vorläuferkarten:~~

- ~~Die HBA-Vorläuferkarten können mittels Konnektor nicht für Stapelsignaturen verwendet werden.~~
- Für HBAs steuert der Konnektor die Eingabe der Signatur-PIN am Kartenterminal. Wenn ein Signaturstapel mehr Dokumente enthält, als im Signaturzertifikat angegeben, wird der Signaturstapel vom Konnektor geteilt. Der Konnektor fordert in diesem Fall für jeden Teilstapel eine PIN-Eingabe an.

...

~~<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 48: 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. Ab PTV6 wird, sofern von der verwendeten Signaturkarte unterstützt, mit ECC signiert. Das genaue zu erwartende Verhalten in Abhängigkeit der Signaturkarte ist [gemSpec_Kon# TIP1_A_5010-*] zu entnehmen.

~~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.~~

~~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>~~

...

In Kapitel 4.4.1 "Erstellen digitaler Signaturen" wird der Freitext unter A_13527 wie folgt geändert:

...

Das Einbetten von OCSP-Antworten wird vom Konnektor ~~nur für~~ sowohl für die QES als auch für die nonQES unterstützt (nicht jedoch bei PDF-Signaturen).~~Für die nonQES ist das Einbetten von OCSP-Antworten nicht möglich.~~

Einzubettende OCSP-Antworten werden nach der Erstellung der Signaturen eingeholt und eingebettet. Dies hat möglicherweise Einfluss auf das Handling von Stapelsignaturen durch die Primärsysteme.

~~<PTV5+>~~
~~Ab der Konnektor Produkttypversion PTV 5.61.0 gibt es eine neue Version des `SignatureService` V7.5.6 mit zwei semantischen Änderungen:~~

- ~~Es wird das Einbetten von OCSP-Antworten jetzt auch bei nonQES-Signaturen unterstützt.~~
- ~~Es ändert sich der Konnektor-interne Ablauf bei der Erstellung von QES und nonQES-Signaturen: Einzubettende OCSP-Antworten werden jetzt nach der Erstellung der Signaturen eingeholt und eingebettet. Dies hat möglicherweise Einfluss auf das bisherige Handling von Stapelsignaturen durch die Primärsysteme.~~

~~Das Einbetten von OCSP-Antworten wird vom Konnektor sowohl für die QES als auch für die nonQES unterstützt (nicht jedoch bei PDF-Signaturen).~~

~~<\PTV5+>~~

...

Am Ende von Kapitel 4.4.3 "Verifizieren Digitaler Signaturen" wird folgende Anforderung ergänzt:

Der Konnektor prüft, ob die Kryptographie der Signatur gemäß europäischer Richtlinie vertrauenswürdig ist. Wenn diese Prüfung Fehlschlägt, muss diese Information an den Nutzer weitergegeben werden. Ab dem 01.01.2026 verlieren die verbreiteten RSA-2048-Schlüssel ihren Vertrauensstatus:

A_25844 - Anzeigen der Kryptographieprüfung
Das Primärsystem MUSS die
Meldung SIG:VerifyDocumentResponse/SIG:OptionalOutputs/vr:VerificationReport/vr:IndividualReport/dss:Result/dss:ResultMessage dem Anwender anzeigen, unabhängig davon, ob ein VerificationReport angefordert wurde.**[<=]**

In Kapitel 4.4.5.1 "Verschlüsseln" wird der Freitext unter A_13536 wie folgt geändert:

- Tabelle 25 und beschreibungen zum crypt Parameter werden entfernt. Es wird stattdessen auf SignDocument in gemSpec_Kon verwiesen, weil dort bereits alle Kryptoverfahren in Abhängigkeit der Karte beschrieben sind.
-

...

~~<PTV4>~~

~~Nach der Einführung von elliptischen Kurven auf TI-Smartcards der Generation G2.1 ist es optional möglich, bei `EncryptDocument` die Verwendung von ECC- und RSA-Zertifikaten durch den optionalen Parameter `Crypt` zu steuern.~~

Tabelle 49: Tab_ILF_PS_Steuerung_Verschlüsselungsalgorithmus

Parameter Crypt	Smartcard Objektsystemversion < 4.4.0 oder HBA-V (Kartengeneration noch nicht G2.1)	SmartcardObjektsystemversion >= 4.4.0 (ab Kartengeneration G2.1)
wird nicht verwendet	RSA-Verschlüsselung	RSA-Verschlüsselung
"ECC"	keine Verschlüsselung, Fehlermeldung	ECC-Verschlüsselung
"RSA"	RSA-Verschlüsselung	RSAECC-Verschlüsselung
"RSA_ECC"	RSA-Verschlüsselung	RSA- und ECC-Verschlüsselung, wenn beide Typen von Verschlüsselungszertifikaten auf der Smartcard vorhanden sind

[gemSpec_Konn#TAB_KON_747 KeyReference für Encrypt-/DecryptDocument] listet die ausgewählten Encrypt-Zertifikate je nach Kartentyp auf.

Sämtliche Konnektoren können mit elliptischen Kurven verschlüsseln. In Abhängigkeit von den vorhandenen Schlüsseln wird ab PTV6 bevorzugt und im Default-Verhalten mit ECC verschlüsselt. Das genaue zu erwartende Verhalten in Abhängigkeit der TI-Smartcard ist [gemSpec_Kon#TIP1-A_4621-*] und [gemSpec_Konn#TAB_KON_747 KeyReference für Encrypt-/DecryptDocument] zu entnehmen.

Das PS soll den Parameter Crypt nicht verwenden oder mit dem Wert "RSA" belegen, falls das hybrid verschlüsselte Dokument zur Entschlüsselung durch einen Konnektor vorgesehen ist, der noch nicht ECC verarbeiten kann ist, d.h. noch nicht PTV4 entspricht.

Falls unbekannt ist, ob der Konnektor, der beim Entschlüsseln eingesetzt wird, ECC unterstützt, soll beim Verschlüsseln der Parameter Crypt auf "RSA_ECC" gesetzt werden, so dass zwei Chiffrate entstehen (RSA-Chiffrat und ECC-Chiffrat).

</PTV4>

A_21781-01 wird ins Kapitel 4.1.4.3 gemILF_PS aufgenommen.

Für A_21781-01 wurden die Zuordnungen zu Prüfverfahren des Konnektors entfernt.

A_21781-01 - Nutzerhinweis bezüglich Fehler bei Re-Registrierung

Der Hersteller des Konnektors MUSS in seinem Handbuch den Nutzer/Administrator darauf hinweisen, dass das Ereignis mit dem Topic=SMC_K/REGISTER/ERROR dringend durch das Primärsystem abonniert werden sollte.

Das Primärsystem MUSS das Ereignis mit dem Topic=SMC_K/REGISTER/ERROR abonnieren.

Beim Auftreten des Fehlers mit parameters = „Fail=No_Smcb" muss das Primärsystem an seiner Nutzeroberfläche anzeigen in der Leistungserbringerumgebung dafür gesorgt

~~werden~~, dass keine freigeschaltete SMC-B verfügbar ist, die der Konnektor beim nächsten Re-Registrierungsversuch verwenden kann.
Bei allen anderen Ereignissen mit dem Topic=SMC_K/REGISTER/ERROR muss das Primärsystem an seiner Nutzeroberfläche darüber informieren, dass eine Re-Registrierung fehlgeschlagen und ein DVO-Einsatz nötig ist, bevor der Konnektor den nächsten Re-Registrierungsversuch startet.
[<=]