FESTLEGUNGEN DES BMF ZU DETAILFRAGEN DER REGISTRIERKASSENSICHERHEITSVERORDNUNG (RKSV)

BUNDESMINISTERIUM FÜR FINANZEN

A-SIT PLUS GMBH

VERSION 1.2 – 05.091 – 11.03.2016

Einleitung: Dieses Dokument wurde in Zusammenarbeit zwischen dem BMF und A-SIT Plus erstellt. Es enthält Festlegungen in technischen Detailfragen zur RKSV auf Prozessebene und Klarstellungen bzw. Ergänzungen im Bereich der Mustercode-Beispiele. Darüber hinaus werden in diesem Dokument häufig gestellte Fragen der Kassenhersteller und Sonderfälle aufgegriffen und geregelt. ~~Mit Version 1.1. werden Formate für Testfälle definiert und Basistestfälle für die Überprüfung der Korrektheit von Kassensystemen integriert.~~

Ausblick: In der nächsten Version des Dokuments werden die Testfälle für die Überprüfung der Kassenimplementierung erweitert sowie eine detaillierte Übersicht über RKSV-konforme Kassenkonfigurationen bereitgestellt.

Inhalt

Revisionen

Version	*Datum*	*Autor*	*Anmerkung*
1.0	16.02.2016	BMF, A-SIT Plus	Initiale Version
1.1	11.03.2016	BMF, A-SIT Plus	Alle Änderungen werden auch als „Diff-Dokument“ zur Verfügung gestellt, um einen schnellen Einblick auf Details zu ermöglichen. Es wurde in den Abschnitten, die die Signaturerstellung behandeln, fälschlicherweise die Bezeichnung „*Signatureinrichtung ausgefallen*“ statt „*Sicherheitseinrichtung ausgefallen*“ verwendet. Alle Referenzen auf *Signatureinrichtung* wurden durch *Sicherheitseinrichtung* ersetzt. In Passagen zu bestimmten Belegtypen wurde durch einen Fehler fälschlicherweise auf Informationen vom Startbeleg verwiesen. Die fehlerhaften Passagen wurden korrigiert. Diverse Typos bzw. Qualitätsmängel wurden behoben. Ergänzung einer Anmerkung zu der Verwendung der UTF-8 Kodierung der **Kassen-ID** bzw. **Belegnummer**. Hinzufügen der Kapitel die die automatisierten Tests von Kassensystemen beschreiben.
1.2	30.08.2016	BMF, A-SIT Plus	Alle Änderungen werden auch als „Diff-Dokument“ zur Verfügung gestellt, um einen schnellen Einblick auf Details zu ermöglichen. **Änderungen:** Ergänzung um den Prozess für die Berechnung der Prüfsumme des AES-Schlüssels für die manuelle Eingabe im FinanzOnline. Behebung der Fehler die über GitHub berichtet wurden: • Issue 48, Typos, falsche Referenz auf RKSV • Issue 52, Vertauschte Absätze für Emulation von ICM/CTR Modus bei Verschlüsselung. • Issue 54, DEP-Export BSP UTF-8 Escape-Sequenz \u003d durch = ersetzt. • Issue 56, Typo bei Beispiel für DEP-Export. • Issue 77, Korrektur der Beispiele für Stornobelege. Der Hinweis, dass die Aufbereitung des verschlüsselten Umsatzzählers notwendig ist, wurde entfernt. Dies ist beim Stornobeleg nicht notwendig, da der verschlüsselte Umsatzzähler nicht in den maschinenlesbaren Code übernommen wird. • Issue 118: Typo bei Beispiel für Verkettung

1 Allgemeine Informationen

Als Grundlage für ein korrektes Verständnis der im vorliegenden Dokument ausgeführten technischen Details werden in diesem einführenden Abschnitt relevante Begriffe definiert und grundlegende Eigenschaften festgelegt. Die in diesem Abschnitt enthaltenen Informationen basieren auf Definitionen der Registrierkassensicherheitsverordnung (RKSV), sind jedoch für ein besseres Verständnis in einem höheren Detailierungsgrad ausgeführt. Ziel ist es, ein gemeinsames Verständnis der für die Umsetzung einer Registrierkasse gemäß RKSV relevanten Komponenten und deren Beziehung zueinander zu gewährleisten.

Dieser Abschnitt ist in zwei Unterabschnitte gegliedert. Im ersten Unterabschnitt werden Definitionen für relevante Komponenten einer Registrierkasse gegeben. Basierend darauf wird im zweiten Unterabschnitt dargelegt, welche Relationen diese Komponenten zueinander haben.

1.1 Definitionen

Für die Umsetzung einer Registrierkasse gemäß RKSV sind die folgenden Komponenten relevant.

1.1.1 Registrierkasse

Die RKSV definiert den Begriff der Registrierkasse wie folgt:

„Registrierkasse (auch elektronische Registrierkasse): verallgemeinerte Form jedes elektronischen Datenverarbeitungssystems, das elektronische Aufzeichnungen zur Losungsermittlung und Dokumentation von einzelnen Barumsätzen erstellt, insbesondere elektronische Registrierkassen jeglicher Bauart, serverbasierte Aufzeichnungssysteme (auch zur Abwicklung von Online-Geschäften), Waagen mit Kassenfunktionen und Taxameter. Eine Registrierkasse kann mit Eingabestationen verbunden sein“

[RKSV~~, 2015~~]

Eine Registrierkasse wird im Wesentlichen durch ein Datenerfassungsprotokoll (DEP) definiert. Das heißt, jede Registrierkasse hat genau ein DEP. An eine Registrierkasse können beliebig viele Eingabestationen angeschlossen sein. Für die Signierung von Belegen gemäß RKSV steht einer Registrierkasse zumindest eine qualifizierte elektronische~~Sichere~~ Signaturerstellungseinheit zur Verfügung.

Formatiert: Deutsch (Deutschland)

Formatiert: Deutsch (Deutschland)

1.1.2 Kassen-ID

Die RKSV definiert den Begriff der Kassen-ID (Kassenidentifikationsnummer) wie folgt:

„Kassenidentifikationsnummer: über FinanzOnline gemeldetes Kennzeichen einer Registrierkasse, das auch die Unterscheidung verschiedener Registrierkassen mit gleicher Signaturerstellungseinheit ermöglicht“

[RKSV~~, 2015~~]

Die Kassen-ID muss innerhalb eines Unternehmens eindeutig sein. Das heißt, ein Unternehmer darf zwei oder mehr eigenen Registrierkassen nicht dieselbe Kassen-ID zuweisen. Sehr wohl dürfen jedoch zwei unterschiedliche Unternehmer dieselbe Kassen-ID verwenden. Prinzipiell gilt, dass die Kassen-ID durch den Unternehmer frei wählbar ist und jeder Registrierkasse genau eine Kassen-ID zugeordnet werden muss.

1.1.3 AES-Schlüssel

AES (Advanced Encryption Standard) bezeichnet ein symmetrisches Verschlüsselungsverfahren. Mit diesem Verfahren können beliebige Daten mithilfe eines Schlüssels verschlüsselt und wieder entschlüsselt werden. Eine Entschlüsselung der Daten ist nur möglich, wenn der für die Verschlüsselung verwendete Schlüssel bekannt ist. Aus den verschlüsselten Daten alleine kann weder auf die Klartextdaten noch auf den verwendeten Schlüssel geschlossen werden.

Die RKSV definiert die Verwendung von AES für die Verschlüsselung des Summenspeichers der Registrierkasse. Da der Summenspeicher der Registrierkasse Teil jedes erstellten Belegs ist,

muss der Summenspeicher vor Aufbringung am Beleg verschlüsselt werden, um die Vertraulichkeit des Summenspeichers zu gewährleisten. Der für diese Verschlüsselung verwendete Schlüssel wird in Folge als AES-Schlüssel bezeichnet.

Der AES-Schlüssel kann vom Unternehmer frei gewählt werden und muss im Zuge der Registrierung der Registrierkasse über FinanzOnline bekanntgegeben werden. Einer Registrierkasse ist genau ein AES-Schlüssel zugeordnet. Mehrere Registrierkassen können innerhalb eines Unternehmens auch denselben AES-Schlüssel verwenden. Ein Wechsel des AES-Schlüssels im laufenden Betrieb ist nicht möglich.

1.1.4 Datenerfassungsprotokoll

Die RKSV definiert den Begriff des Datenerfassungsprotokolls (DEP) wie folgt:

„Datenerfassungsprotokoll (DEP): eine im Speicher der Registrierkasse oder in einem externen Speicher mitlaufende Ereignisprotokolldatei, die in Echtzeit jeweils mit Belegerstellung vollständig, fortlaufend chronologisch die Barumsätze mit Beleginhalten dokumentiert“

[RKSV~~, 2015~~]

Jede Registrierkasse besitzt genau ein DEP. Das heißt die Entität Registrierkasse ist über die Existenz eines eigenen DEP definiert. Dies unterscheidet zum Beispiel Registrierkassen von an eine Registrierkasse angeschlossenen Eingabestationen, die selbst über kein DEP verfügen.

1.1.5 Summenspeicher

Die RKSV definiert den Begriff des Summenspeichers wie folgt:

„Summenspeicher: Speicher in der Registrierkasse, die Zwischen- oder einen aktuellen Endstand aufsummierter Beträge wiedergeben“

[RKSV~~, 2015~~]

Wie das DEP ist auch der Summenspeicher eindeutig einer Registrierkasse zugeordnet. Das heißt, jede Registrierkasse verfügt über genau einen Summenspeicher.

1.1.6 Belegnummer

Jeder von einer Registrierkasse erstellte Beleg muss über die Belegnummer des Barumsatzes eindeutig pro Kassen-ID und AES-Schlüssel unterscheidbar gemacht werden. Die Belegnummer eines Belegs ist innerhalb eines Unternehmens für Kassen-ID und AES-Schlüssel eindeutig.

Wird beispielsweise nach Stilllegung einer Registrierkasse A eine neue Registrierkasse B mit der gleichen Kassen-ID und dem gleichen AES-Schlüssel in Betrieb genommen, darf Registrierkasse B keine Belegnummern vergeben, die bereits zuvor von Registrierkasse A vergeben wurden. Ändert sich im Zuge der Inbetriebnahme von Registrierkasse B die Kassen-ID und/oder der AES-Schlüssel, dürfen Belegnummern, die bereits von Registrierkasse A vergeben wurden, auch von Registrierkasse B nochmal vergeben werden.

1.1.7 ~~Sichere~~ Signaturerstellungseinheit

Die RKSV definiert den Begriff der „Qualifizierten elektronischen~~Sicheren~~ Signaturerstellungseinheit“ wie folgt:

„Qualifizierte elektronische~~„Sichere~~ Signaturerstellungseinheit: konfigurierte Software oder Hardware, die
~~-~~zur Verarbeitung der Signaturerstellungsdaten verwendet wird und die den Sicherheitsanforderungen des Anhangs II der eIDAS-VO~~SigG sowie den dazu erlassenen Verordnungen~~ entspricht“~~ (§ 2 Z 5 SigG)“~~

[RKSV~~, 2015~~]

Formatiert: Block

In weiterer Folge wird der Begriff „Signaturerstellungseinheit“ sowohl für die „Qualifizierten elektronischen Signaturerstellungseinheiten“ als auch für die Systeme, die im geschlossenen Gesamtsystem zum Einsatz kommen, verwendet.

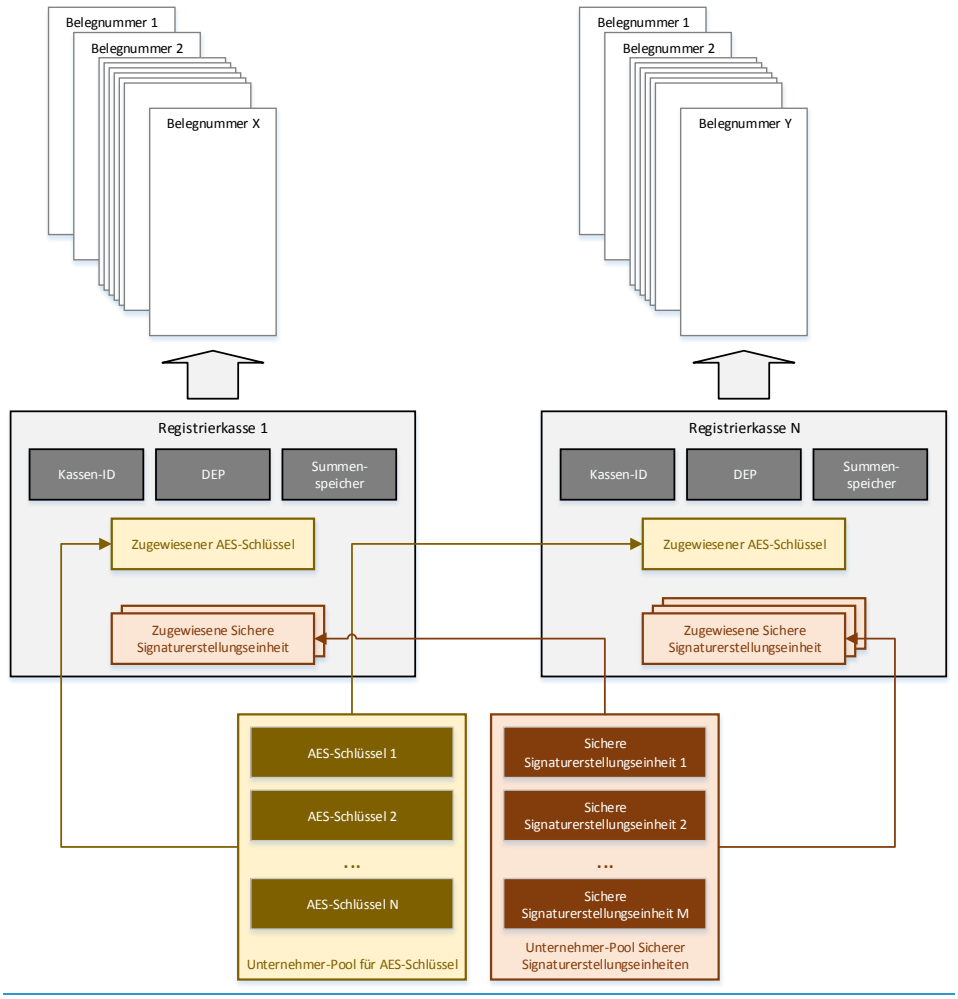
Die Signaturerstellungseinheit wird von der Registrierkasse verwendet, um Belege zu signieren. Die Kommunikation zwischen Registrierkasse und ~~Sicherer~~ Signaturerstellungseinheit hängt auch von der jeweiligen Implementierung ab. Im Speziellen können sich in Bezug auf die ~~Sichere~~ Signaturerstellungseinheit Unterschiede zwischen herkömmlichen Kassensystemen und Geschlossenen Gesamtsystemen gemäß RKSV ergeben.

Eine Registrierkasse kann beliebig viele ~~Sichere~~ Signaturerstellungseinheiten für die Belegsignierung verwenden. Eine ~~Sichere~~ Signaturerstellungseinheit kann auch von mehreren Registrierkassen eines Unternehmers verwendet werden. Eine gemeinsame Verwendung einer ~~Sicheren~~ Signaturerstellungseinheit durch mehrere Unternehmer ist nicht zulässig.

Formatiert: Deutsch (Deutschland)

1.2 Relationen

Im vorangegangenen Unterabschnitt wurden jene Komponenten, die für die Umsetzung einer Registrierkasse gemäß RKSV relevant sind, definiert und kurz beschrieben. Punktuell wurde auch bereits auf die Relationen der einzelnen Komponenten zueinander eingegangen. Diese Relationen sind in folgender Abbildung anhand zweier Registrierkassen nochmal exemplarisch dargestellt.


Belegnummer 1
Belegnummer 2
Belegnummer X
Belegnummer 1
Belegnummer 2
Belegnummer Y
Registrierkasse 1
Kassen-ID
DEP
Summen-speicher
Zugewiesener AES-Schlüssel
Zugewiesene Sichere Signaturerstellungseinheit
Registrierkasse N
Kassen-ID
DEP
Summen-speicher
Zugewiesener AES-Schlüssel
Zugewiesene Sichere Signaturerstellungseinheit
AES-Schlüssel 1
AES-Schlüssel 2
...
AES-Schlüssel N
Unternehmer-Pool für AES-Schlüssel
Sichere Signaturerstellungseinheit 1
Sichere Signaturerstellungseinheit 2
...
Sichere Signaturerstellungseinheit M
Unternehmer-Pool Sicherer Signaturerstellungseinheiten

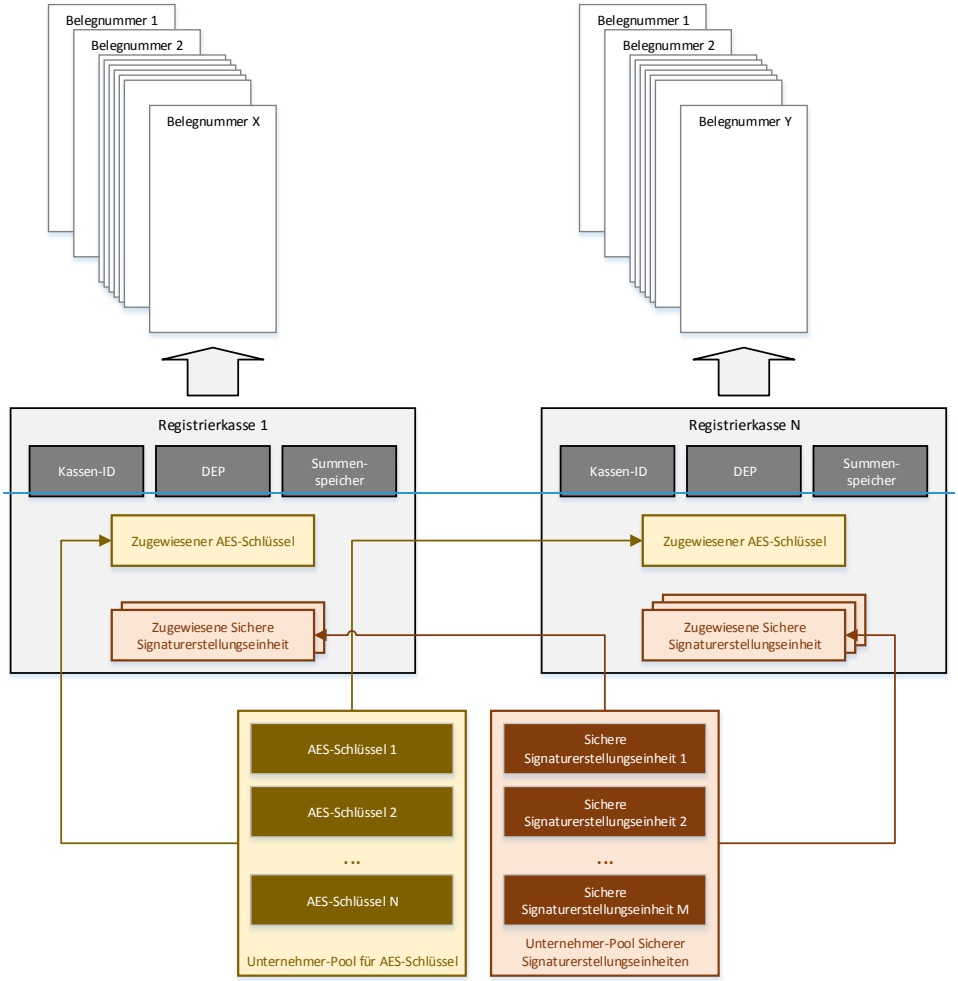


Abbildung 1. Beispiel-Setup für zwei Registrierkassen.

Abbildung 1 illustriert die Relationen der einzelnen für die Umsetzung einer RKSV-konformen Registrierkasse notwendigen Komponenten. Konkret werden aus Abbildung 1 folgende Sachverhalte ersichtlich:

- Konzeptionell betrachtet verfügt ein Unternehmer über N Registrierkassen. Neben diesen verfügt der Unternehmer über einen Pool von AES-Schlüsseln und einen Pool von ~~Sicheren~~ Signaturerstellungseinheiten. Aus diesen beiden Pools verwenden die N Registrierkassen des Unternehmers jeweils genau einen AES-Schlüssel und eine der verfügbaren ~~Sicheren~~ Signaturerstellungseinheiten. Das heißt, einer Registrierkasse wird genau ein verfügbarer AES-Schlüssel statisch zugewiesen. Dieser AES-Schlüssel wird während des Betriebs der Registrierkasse verwendet und kann nicht geändert werden. Die Verwendung der ~~Sicheren~~ Signaturerstellungseinheiten ist hingegen dynamisch, sodass die Registrierkasse im laufenden Betrieb für jeden Beleg neu entscheiden kann, welche registrierte ~~Sichere~~ Signaturerstellungseinheit verwendet werden soll.

- Jede Registrierkasse eines Unternehmers verfügt über eine eigene Kassen-ID, ein eigenes DEP und einen eigenen Summenspeicher. Für N Registrierkassen eines Unternehmers ergeben sich daher N unterschiedliche Kassen-IDs, DEPs und Summenspeicher.

- Jede Registrierkasse eines Unternehmers verfügt über genau einen AES-Schlüssel. Prinzipiell kann jeder Registrierkasse ein anderer AES-Schlüssel zugewiesen werden. Zwei oder mehr Registrierkassen können sich aber auch einen AES-Schlüssel teilen. Bei N Registrierkassen verfügt der Unternehmer daher über einen Pool von 1 bis N AES-Schlüssel.

- Jeder Registrierkasse steht zumindest eine ~~Sichere~~ Signaturerstellungseinheit zur Verfügung. Einer Registrierkasse können auch mehrere ~~Sichere~~ Signaturerstellungseinheiten zur Verfügung stehen. All diese ~~Sicheren~~ Signaturerstellungseinheiten müssen jedoch dem Unternehmer zugeordnet und in FinanzOnline registriert sein. Eine ~~Sichere~~ Signaturerstellungseinheit kann auch von mehreren Registrierkassen verwendet werden. Im Zuge der Belegerstellung kann die Registrierkasse eine beliebige ihr zur Verfügung stehende ~~Sichere~~ Signaturerstellungseinheiten für die Beleg-Signierung verwenden. Die Möglichkeit der dynamischen Wahl der zu verwendenden ~~Sicheren~~ Signaturerstellungseinheit ist notwendig, um beispielsweise den Zugriff auf zentrale ~~Sichere~~ Signaturerstellungseinheiten über Load-Balancing-Mechanismen zu ermöglichen, oder auch um Registrierkassen die Möglichkeit zu geben, im Fehlerfall dynamisch auf vorhandene Ersatzeinheiten zur ~~sicheren~~ Signaturerstellung zu wechseln.

- Anmerkung zu Load-Balancing: Bei der Verwendung von mehreren ~~Sicheren~~ Signaturerstellungseinheiten pro Kasse muss darauf geachtet werden, dass der Signaturvorgang und die Erstellung des maschinenlesbaren Codes abgeschlossen ist bevor der nächste Beleg erstellt wird, bzw. signiert werden kann. Der Notwendigkeit für die Einhaltung dieser Reihenfolge ist die kryptographische Verkettung der Belege.
  Im Zusammenhang mit Load Balancing kann also nicht die Belegerstellung mit beliebigen Signatureinheiten parallelisiert werden, es kann aber vor der Signaturerstellung aus der Menge der zur Verfügung stehenden Signatureinheiten eine nicht verwendete gewählt werden. Gerade beim Einsatz von mehreren Karten kann somit eine Verteilung der Last auf mehrere Signatureinheiten erzielt werden und vor allem dynamisch bei höherer Auslastung beliebige weitere Signatureinheiten zu dieser Menge hinzugefügt werden oder entfernt werden (entweder aufgrund einer niedrigeren Last oder durch den Ausfall von Signatureinrichtungen).

- Jede Registrierkasse erzeugt signierte Belege mit einer in Bezug auf Kassen-ID und AES-Schlüssel eindeutigen Belegnummer.

Abbildung 1 zeigt ein Beispiel-Setup für einen Unternehmer. Zwischen verschiedenen Unternehmern ergeben sich nur wenige Abhängigkeiten. Dennoch sei an dieser Stelle auf folgende Punkte hingewiesen:

- Zwei oder mehrere Unternehmer können sich weder eine Registrierkasse noch eine ~~Sichere~~ Signaturerstellungseinheit[1] teilen.

- Theoretisch können zwei oder mehrere Unternehmer zufällig den gleichen AES-Schlüssel wählen. Werden AES-Schlüssel zufällig generiert, ist dies in der Praxis jedoch äußerst unwahrscheinlich.

[1] Hier ist gemeint, dass zwei Unternehmen nicht das gleiche Signaturzertifikat verwenden können. Das Teilen einer Signaturerstellungseinheit ist im weiteren Sinne schon möglich, da auch zentrale HSM-Lösungen oder Remote-Signature Lösungen verwendet werden können. Diese können je nach Konfiguration von mehreren Unternehmen gleichzeitig eingesetzt werden.

- Kassen-IDs können von Unternehmern frei gewählt werden. Es kann und muss daher nicht davon ausgegangen werden, dass eine gewählte Kassen-ID, die innerhalb des Unternehmens eindeutig ist, auch unternehmerübergreifend eindeutig ist.

- Von Signaturerstellungseinheiten verwendete Signaturschlüssel und zugehörige Signatur-Zertifikate können als global eindeutig betrachtet werden.

- Belegnummern sind nur innerhalb eines Unternehmens pro Kassen-ID und AES-Schlüssel eindeutig. Verwenden zwei oder mehrere Unternehmer ein ähnliches Nummerierungs-Schema, kann nicht ausgeschlossen werden, dass Belege unterschiedlicher Unternehmer die gleiche Belegnummer aufweisen. Auch innerhalb eines Unternehmens können Belegnummern mehrfach vergeben werden, sofern sich Kassen-ID und/oder AES-Schlüssel der belegerstellenden Registrierkassen unterscheiden.

2 Aufbereitung der Daten für die Signaturerstellung

2.1 Daten für die Erstellung des maschinenlesbaren Codes

Eingabewerte

- Daten für Belegerstellung
  - Daten aus der Kasse
  - Rohbelegdaten

Prozessbeschreibung

Im Folgenden werden die Felder und deren Formate beschrieben, die für die Belegerstellung im Sinne der RKSV benötigt werden. Dazu sind folgende Anmerkungen relevant:

- **Längenbeschränkung**: Für einige der in der weiteren Folge definierten Felder ist keine Längenbeschränkung vorgesehen, dennoch muss berücksichtigt werden, dass der aufbereitete maschinenlesbare Code zu kurz wie möglich gehalten werden soll, um die QR/OCR Repräsentation am Beleg so einfach wie möglich zu halten. Lediglich bei der LINK-Repräsentation hat die Länge des maschinenlesbaren Codes keinen Einfluss für die Darstellung am Beleg.
- **Reihenfolge**: Die folgende Reihenfolge entspricht auch der Reihenfolge die für die Aufbereitung der zu signierenden Daten relevant ist.
- **Anmerkung zu UTF-8 Kodierung**: Diese Anmerkung betrifft vor allem die Elemente **Kassen-ID** und **Belegnummer**: Um hier keine wesentlichen Einschränkungen zu haben, wird UTF-8 für die Kodierung dieser Elemente verwendet. Es sollte dennoch beachtet werden, dass diese Daten – vor allem bei der manuellen Registrierung einer Kasse – vom Unternehmer eingegeben werden müssen. Es empfiehlt sich daher eine für den Unternehmer möglichst benutzerfreundliche Zeichenmenge zu verwenden, die z.B. UTF-8 Escape Sequenzen vermeidet.

Felder und deren Werte:

- **Kassen-ID:**
  - **Beschreibung**: Hierbei handelt es sich um die Kassenidentifikationsnummer die für die Registrierung der Kasse in Finanzonline verwendet wird. Die Bezeichnung entspricht einer beliebigen Zeichenkette die UTF-8 kodiert wird.
  - **Referenz RKSV**: § 9 Abs. 2 Z 1
  - **Format**: JSON-Format *string,* UTF-8 kodiert
  - **Länge**: Keine Einschränkung
  - **Sonderfälle**: Keine. Es wird bei allen möglichen Varianten der Belegerstellung die tatsächliche Kassenidentifikationsnummer verwendet.
- **Belegnummer:**
  - **Beschreibung**: Hierbei handelt es sich um die Belegnummer des Belegs. Die Belegnummer entspricht einer alphanumerischen Zeichenkette die UTF-8 kodiert wird.
  - **Referenz RKSV**: § 9 Abs. 2 Z 2
  - **Format**: JSON-Format *string,* UTF-8 kodiert
  - **Länge**: Keine Einschränkung
  - **Sonderfälle**: Keine. Es muss immer eine Belegnummer vergeben werden für die garantiert ist, dass sie bei gleicher **Kassen-ID** und gleichem AES Schlüssel immer nur einmal verwendet wird.
- **Beleg-Datum-Uhrzeit:**
  - **Beschreibung**: Das Datum und die Uhrzeit wird im ISO 8601 Format ohne der Angabe der Zeitzone angegeben (`JJJJ-MM-TT'T'hh:mm:ss`, z. B. `2015-07-21T14:23:34`). Es wird immer von österreichischer Lokalzeit (CET/MEZ) ausgegangen.
  - **Referenz RKSV**: § 9 Abs. 2 Z 3
  - **Format**: JSON-Format *string,* UTF-8 kodiert
  - **Länge**: Aufgrund des Formats festgelegt.
  - **Sonderfälle**: Keine. Es wird bei allen möglichen Varianten der Belegerstellung der tatsächliche Zeitpunkt der Belegerstellung angegeben.
- **Betrag-Satz-Normal:**
  - **Beschreibung**: Summe der Brutto-Werte der Belegpositionen des zu

verarbeitenden Belegs mit dem Steuersatz „Normal“. Ist keine Position mit diesem Steuersatz im Beleg vorhanden, so wird der Wert `0,00` eingetragen.
    - **Referenz RKSV**: § 9 Abs. 2 Z 4
    - **Format**: JSON-Format *number* mit 2 Kommastellen. Dezimaltrennzeichen entspricht `,`
    - **Länge**: Aufgrund des Formats und des gegebenen Werts festgelegt.
    - **Sonderfälle**: Bei „Standardbelegen“, „Stornobelegen“ und „Trainingsbelegen“ entspricht der Wert dem tatsächlichem Wert. Bei „Nullbelegen“ und dem „Startbeleg“ wird der Wert per Definition auf `0,00` gesetzt.
- **Betrag-Satz-Ermaessigt-1:**
    - **Beschreibung**: Summe der Brutto-Werte der Belegpositionen des zu verarbeitenden Belegs mit dem Steuersatz „Ermaessigt-1“. Ist keine Position mit diesem Steuersatz im Beleg vorhanden, so wird der Wert `0,00` eingetragen.
    - **Referenz RKSV**: § 9 Abs. 2 Z 4 RKSV
    - **Format**: JSON-Format *number* mit 2 Kommastellen. Dezimaltrennzeichen entspricht `,`
    - **Länge**: Aufgrund des Formats und des gegebenen Werts festgelegt.
    - **Sonderfälle**: Bei „Standardbelegen“, „Stornobelegen“ und „Trainingsbelegen“ entspricht der Wert dem tatsächlichem Wert. Bei „Nullbelegen“ und dem „Startbeleg“ wird der Wert per Definition auf `0,00` gesetzt.
- **Betrag-Satz-Ermaessigt-2:**
    - **Beschreibung**: Summe der Brutto-Werte der Belegpositionen des zu verarbeitenden Belegs mit dem Steuersatz „Ermaessigt-2“. Ist keine Position mit diesem Steuersatz im Beleg vorhanden, so wird der Wert `0,00` eingetragen.
    - **Referenz RKSV**: § 9 Abs. 2 Z 4 RKSV
    - **Format**: JSON-Format *number* mit 2 Kommastellen. Dezimaltrennzeichen entspricht `,`
    - **Länge**: Aufgrund des Formats und des gegebenen Werts festgelegt.
    - **Sonderfälle**: Bei „Standardbelegen“, „Stornobelegen“ und „Trainingsbelegen“ entspricht der Wert dem tatsächlichem Wert. Bei „Nullbelegen“ und dem „Startbeleg“ wird der Wert per Definition auf `0,00` gesetzt.
- **Betrag-Satz-Null:**
    - **Beschreibung**: Summe der Brutto-Werte der Belegpositionen des zu verarbeitenden Belegs mit dem Steuersatz „Null“. Ist keine Position mit diesem Steuersatz im Beleg vorhanden, so wird der Wert `0,00` eingetragen.
    - **Referenz RKSV**: § 9 Abs. 2 Z 4 RKSV
    - **Format**: JSON-Format *number* mit 2 Kommastellen. Dezimaltrennzeichen entspricht `,`
    - **Länge**: Aufgrund des Formats und des gegebenen Werts festgelegt.
    - **Sonderfälle**: Bei „Standardbelegen“, „Stornobelegen“ und „Trainingsbelegen“ entspricht der Wert dem tatsächlichem Wert. Bei „Nullbelegen“ und dem „Startbeleg“ wird der Wert per Definition auf `0,00` gesetzt.
- **Betrag-Satz-Besonders:**
    - **Beschreibung**: Summe der Brutto-Werte der Belegpositionen des zu verarbeitenden Belegs mit dem Steuersatz „Besonders“. Ist keine Position mit diesem Steuersatz im Beleg vorhanden, so wird der Wert `0,00` eingetragen.
    - **Referenz RKSV**: § 9 Abs. 2 Z 4 RKSV
    - **Format**: JSON-Format *number* mit 2 Kommastellen. Dezimaltrennzeichen entspricht `,`
    - **Länge**: Aufgrund des Formats und des gegebenen Werts festgelegt.
    - **Sonderfälle**: Bei „Standardbelegen“, „Stornobelegen“ und „Trainingsbelegen“ entspricht der Wert dem tatsächlichem Wert. Bei „Nullbelegen“ und dem „Startbeleg“ wird der Wert per Definition auf `0,00` gesetzt.
- **Stand-Umsatz-Zaehler-AES256-ICM:**
    - **Beschreibung**: Hierbei handelt es sich um den verschlüsselten Umsatzzähler der jeweiligen Kasse. Für die Verarbeitung dieses Felds wird auf Prozess 2.5

verwiesen.
    - **Referenz RKSV**: § 9 Abs. 2 Z 5 RKSV
    - **Format**: JSON-Format *string*. BASE64-kodierter Wert des verschlüsselten Gesamtumsatzes.
    - **Länge**: Aufgrund des Auswahl des Parameters „Länge des Umsatzzählers“ in Prozess 2.5.2 festgelegt.
    - **Sonderfälle**:
        - **Startbeleg (siehe 2.2.2)**: Bei Startbelegen wird der Umsatzzähler auf den Wert `0` gesetzt und verschlüsselt.
        - **Stornobelege (siehe 2.2.4)**: Bei Stornobelegen wird anstelle des verschlüsselten Umsatzzählers der BASE64-kodierte Wert der Zeichenkette `STO` eingetragen. Dies entspricht der Zeichenkette `U1RP`.
        - **Trainingsbelege (siehe 2.2.5)**: Bei Trainingsbelegen wird anstelle des verschlüsselten Umsatzzählers der BASE64-kodierte Wert der Zeichenkette `TRA` eingetragen. Dies entspricht der Zeichenkette `VFJB`.
- **Zertifikat-Seriennummer:**
    - **Beschreibung**:
        - **Offene Systeme**: In diesem Fall wird die Seriennummer des Zertifikats eingetragen. Es wird dazu die Hexadezimal-Kodierung[2] der Seriennummer verwendet. Der Präfix `0x` wird nicht angegeben. Beispiel: `a431623bedf` oder `A431623BEDF`
        - **Geschlossene Systeme**: Bei geschlossenen Gesamtsystemen wird der Ordnungsbegriff des Unternehmens – ergänzt um die Kennung des jeweiligen offenen Schlüssels – eingetragen. Der Ordnungsbegriff des Unternehmens wird wie folgt aufbereitet: `{S|U|G}:ID`. Diese Kennung wird mit dem Trennzeichen `-` durch die Identifikationsnummer des jeweiligen Schlüssels ergänzt. Dies ergibt `{S|U|G}:ID-KID`. Dabei entspricht `S` dem Ordnungsbegriff Steuernummer, `U` dem Ordnungsbegriff UID und `G` dem Ordnungsbegriff GLN. `ID` ist die jeweilige Identifikationsnummer. `KID` ist die Identifikationsnummer des Schlüssels. Im Detail ergibt sich daraus:
            - **Steuernummer**
                - **Länge:** immer 9 Stellen
                - **Typ:** numerisch (keine Trennzeichen, Sonderzeichen)
                - **Beispiel**: `S:081953820`
            - **UID**
                - **Länge**: max. 14 Stellen (abhängig vom Land)
                - **Typ**: alphanumerisch (Großbuchstaben)
                - **Beispiel**: `U:ATU57780814`
            - **GLN**
                - **Länge**: immer 13 Stellen
                - **Typ**: numerisch
                - **Beispiel**: `G:9110005102096`
            - **KID**: Die Identifikationsnummer des jeweiligen Schlüssels entspricht einer möglichst kurz gehaltenen Zeichenkette. Gültige Zeichen sind Kleinbuchstaben (`a-z`, kein ü,ö etc.), Großbuchstaben (`A-Z`, kein ö,ü etc.), Ziffern (`0-9`). Es ergeben sich somit 26+26+10=62 Möglichkeiten pro Zeichen. Mit 3 Zeichen können damit bereits 238.328 Schlüssel pro Unternehmen identifiziert werden. Man wird daher in der Realität in vielen Fällen bereits mit 1-2 Zeichen auskommen.
                - **Beispiel**: Gegeben kodierter Ordnungsbegriff des

[2] Hexadezimal wurde gewählt, da viele Bibliotheken, die für die Verarbeitung von Zertifikaten verwendet werden, diese Kodierung wählen.

Unternehmens (Steuernummer): `S:081953820` und Identifikationsnummer des Schlüssels gleich `A1b`. Damit ergibt sich für den Beleg die Zeichenkette: `S:081953820-A1b`

- **Referenz RKSV:** § 9 Abs. 2 Z 6 RKSV
- **Format:** JSON-Format *string*, UTF-8 kodiert
- **Länge:** Abhängig vom Zertifikat des ZDAs bzw. des Ordnungsbegriffs des Unternehmens.
- **Sonderfälle:** Bei geschlossenen Gesamtsystemen wird der jeweilige Ordnungsbegriff des Unternehmens – ergänzt um die Identifikationsnummer des Schlüssels – anstelle der Seriennummer des Zertifikats verwendet.

- **Sig-Voriger-Beleg**:
  - **Beschreibung**: Verkettungswert der die Bindung zum vorigen Beleg herstellt. Für die Berechnung dieses Werts wird auf Prozess 2.4 verwiesen.
  - **Referenz RKSV:** § 9 Abs. 2 Z 7 RKSV
  - **Format:** JSON-Format *string*, UTF-8 kodiert
  - **Länge:** Vorgegeben durch die im Registrierkassenalgorithmuskennzeichen definierten Algorithmen.
  - **Sonderfälle:** Beim Startbeleg wird der Verkettungswert über die **Kassen-ID** gebildet.

Beispiele

Relevante Beispiele werden in den nachfolgenden Prozessen gezeigt.

Referenzen – Muster-Code

Relevante Referenzen zum Muster-Code werden in den jeweiligen Detailprozessen genannt.

Referenzen – RKSV

Z 4 zur Anlage der RKSV

2.2 Verarbeitung von Belegen

- 2.2.1 Verarbeitung eines Standardbelegs
- 2.2.2 Verarbeitung eines Startbelegs
- 2.2.3 Verarbeitung von Nullbelegen (z.B. Jahresbeleg, Monatsbeleg ...)
- 2.2.4 Verarbeitung von Stornobelegen
- 2.2.5 Verarbeitung von Trainingsbelegen

2.2.1 Verarbeitung eines Standardbelegs

Eingabewerte

Eingabewerte für Felder laut Prozess 2.1, die aus Belegdaten aufbereitet werden:
• **Beleg-Satz-Normal**
• **Beleg-Satz-Ermaessigt-1**
• **Beleg-Satz-Ermaessigt-2**
• **Beleg-Satz-Null**
• **Beleg-Satz-Besonders**
 	
Eingabewerte die aus der Kasse extrahiert werden:
• Felder laut Prozess 2.1:
  ○ **Kassen-ID**
  ○ **Belegnummer**
  ○ **Beleg-Datum-Uhrzeit**
  ○ **Zertifikat-Seriennummer**
• Weitere Daten:
  ○ **Umsatzzähler der Kasse**
  ○ **AES-Schlüssel der Kasse**

Prozessbeschreibung
Dieser Prozess ist für die Erstellung von Standardbelegen relevant. Werte für Felder laut Prozess 2.1, die berechnet werden müssen: • **Stand-Umsatz-Zaehler-AES256-ICM** • **Sig-Voriger-Beleg** Es werden nun folgende Prozesse durchgeführt: • **Aufbereiten/Aktualisieren/Verschlüsseln des Umsatzzählers:** siehe Prozess 2.5. Als Ergebnis der dort beschriebenen Detailprozesse erhält man den Wert des Felds **Stand-Umsatz-Zaehler-AES256-ICM.** • **Berechnung des Verkettungswerts:** Für den Verkettungswert wird der vorhergehende Beleg herangezogen. Siehe Prozess 2.4.2: Als Ergebnis erhält man den Wert des Felds **Sig-Voriger-Beleg**.
Beispiele
Beispiele für die unterschiedliche Behandlung der Belegtypen werden in den jeweiligen Detailprozessen gezeigt.
Referenzen – Muster-Code

• Klasse**:**
  ○ `at.asitplus.regkassen.core.DemoCashBox`
  ○ https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/DemoCashBox.java
• Methode**:** *createStoreAndSignReceiptPackage*

Referenzen – RKSV
Relevante Referenzen werden bei den jeweiligen Detailprozessen genannt.
Ausgabewerte
• Daten für die Erstellung der Signatur (Eingabewert für Prozess 2.3)

2.2.2 Verarbeitung eines Startbelegs

Eingabewerte

Eingabewerte für Felder laut Prozess 2.1, die beim Startbeleg einen festgelegten Wert haben:

- **Beleg-Satz-Normal:** `0,00`
- **Beleg-Satz-Ermaessigt-1:** `0,00`
- **Beleg-Satz-Ermaessigt-2:** `0,00`
- **Beleg-Satz-Null:** `0,00`
- **Beleg-Satz-Besonders:** `0,00`

Eingabewerte die aus der Kasse extrahiert werden:

- Felder laut Prozess 2.1:
  - **Kassen-ID**
  - **Belegnummer**
  - **Beleg-Datum-Uhrzeit**
  - **Zertifikat-Seriennummer**
- Weitere Daten:
  - **Umsatzzähler der Kasse:** Dieser ist beim Startbeleg per Definition `0`.
  - **AES-Schlüssel der Kasse**

Prozessbeschreibung

Der Startbeleg stellt den zu erstellenden ersten Beleg einer Kasse dar. Es gibt daher keinen vorigen Beleg, der für die Berechnung des Verkettungswerts herangezogen werden kann. Stattdessen wird das Feld **Kassen-ID** für die Berechnung verwendet.

Werte für Felder laut Prozess 2.1, die berechnet werden müssen:

- **Stand-Umsatz-Zaehler-AES256-ICM**
- **Sig-Voriger-Beleg**

Es werden nun folgende Prozesse durchgeführt:

- **Aufbereiten/Aktualisieren/Verschlüsseln des Umsatzzählers:** Der Umsatzzähler ist beim Starbeleg per Definition `0`. Siehe Prozess 2.5: Als Ergebnis der dort beschriebenen Prozesse erhält man den Wert des Felds **Stand-Umsatz-Zaehler-AES256-ICM**.
- **Berechnung des Verkettungswerts:** Für den Verkettungswert wird das Feld **Kassen-ID** verwendet. Siehe Prozess 2.4.1: Als Ergebnis erhält man den Wert des Felds **Sig-Voriger-Beleg**.

Beispiele

Beispiele für die unterschiedliche Behandlung der Belegtypen werden in den jeweiligen Detailprozessen gegeben.

Referenzen – Muster-Code

- Klasse**:**
  - `at.asitplus.regkassen.core.DemoCashBox`
  - https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/DemoCashBox.java
- Methode**:** *createStoreAndSignReceiptPackage*

Referenzen – RKSV

Relevante Referenzen werden bei den jeweiligen Detailprozessen genannt.

Ausgabewerte

- Daten für die Erstellung der Signatur (Eingabewert für Prozess 2.3)

2.2.3 Verarbeitung von Nullbelegen (z.B. Jahresbeleg, Monatsbeleg ...)

Eingabewerte

Eingabewerte für Felder laut Prozess 2.1, die beim Nullbeleg einen festgelegten Wert haben:

- **Beleg-Satz-Normal:** `0,00`
- **Beleg-Satz-Ermaessigt-1:** `0,00`
- **Beleg-Satz-Ermaessigt-2:** `0,00`
- **Beleg-Satz-Null:** `0,00`
- **Beleg-Satz-Besonders:** `0,00`

Eingabewerte die aus der Kasse extrahiert werden:

- Felder laut Prozess 2.1:
  - **Kassen-ID**
  - **Belegnummer**
  - **Beleg-Datum-Uhrzeit**
  - **Zertifikat-Seriennummer**
- Weitere Daten:
  - **Umsatzzähler der Kasse**
  - **AES-Schlüssel der Kasse**

Prozessbeschreibung

Die folgenden Belege entsprechen Nullbelegen:

- Jahresbelege bzw. alle weiteren Belege die einen bestimmten Zeitraum beschreiben und dabei keine Buchungen vornehmen (Monatsbelege etc.).
- Schlussbelege, die bei der Außerbetriebnahme einer Kasse erstellt werden.
- Belege die nach der Wiederinbetriebnahme einer beschädigten Sicherheitseinrichtung erstellt werden.

Werte für Felder laut Prozess 2.1, die berechnet werden müssen:

- **Stand-Umsatz-Zaehler-AES256-ICM**
- **Sig-Voriger-Beleg**

Es werden nun folgende Prozesse durchgeführt:

- **Aufbereiten/Aktualisieren des Umsatzzählers:** Der Umsatzzähler der Kasse wird bei der Erstellung eines Nullbelegs nicht verändert.
- **Verschlüsseln des Umsatzzählers:** siehe Prozesse 2.5.2, 2.5.3, 2.5.4, 2.5.5. Als Ergebnis der dort beschriebenen Prozesse erhält man den Wert des Felds **Stand-Umsatz-Zaehler-AES256-ICM.**
- **Berechnung des Verkettungswerts:** Für den Verkettungswert wird der vorhergehende Beleg herangezogen. Siehe Prozess 2.4.2: Als Ergebnis erhält man den Wert des Felds **Sig-Voriger-Beleg**.

Beispiele

Beispiele für die unterschiedliche Behandlung der Belegtypen werden in den jeweiligen Detailprozessen gegeben.

Referenzen – Muster-Code

- Klasse**:**
  - `at.asitplus.regkassen.core.DemoCashBox`
  - https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/DemoCashBox.java
- Methode**:** *createStoreAndSignReceiptPackage*

Referenzen – RKSV

Relevante Referenzen werden bei den jeweiligen Detailprozessen genannt.

Ausgabewerte

- Daten für die Erstellung der Signatur (Eingabewert für Prozess 2.3)

2.2.4 Verarbeitung von Stornobelegen

Eingabewerte

Eingabewerte für Felder laut Prozess 2.1, die aus Belegdaten aufbereitet werden:

- **Beleg-Satz-Normal**
- **Beleg-Satz-Ermaessigt-1**
- **Beleg-Satz-Ermaessigt-2**
- **Beleg-Satz-Null**
- **Beleg-Satz-Besonders**

Eingabewerte die aus der Kasse extrahiert werden:

- Felder laut Prozess 2.1:
  - **Kassen-ID**
  - **Belegnummer**
  - **Beleg-Datum-Uhrzeit**
  - **Zertifikat-Seriennummer**
- Weitere Daten:
  - **Umsatzzähler der Kasse**

Prozessbeschreibung

Ein Stornobeleg wird erstellt, wenn einzelne oder mehrere Belegpositionen, die vorher im Rahmen der Erstellung eines Standardbelegs falsch boniert wurden, storniert werden müssen. Stornobelege dürfen nur stornierte Belegpositionen enthalten, da andernfalls die Bezeichnung im maschinenlesbaren Code (`STO` im Umsatzzähler) nicht eindeutig wäre. In den meisten Fällen werden Stornobelege negative Werte erhalten (diese Stornos bewirken eine Verkleinerung des Umsatzzählers). Allerdings müssen auch negative Beträge einer Gutschrift storniert werden können. In diesem Fall wäre der stornierte Wert positiv (diese Stornos bewirken eine Vergrößerung des Umsatzzählers). Die Beträge der einzelnen Positionen des Stornobeleges sind – gleich wie beim Standardbeleg – nach Steuersätzen getrennt für die Signaturerstellung und für die Ausgabe auf dem Beleg aufzubereiten.

Werte für Felder laut Prozess 2.1, die berechnet werden müssen:

- **Sig-Voriger-Beleg**

Es werden nun folgende Prozesse durchgeführt:

- **Aufbereiten/Aktualisieren des Umsatzzählers:** Dies erfolgt – gleich wie bei der Verarbeitung eines Standardbelegs – anhand von Prozess 2.5.1.
- **Verschlüsselung des Umsatzzählers:** Beim Stornobeleg wird anstelle des verschlüsselten Umsatzzählers die BASE64-Kodierung der Zeichenkette `STO`, welche der Zeichenkette `U1RP` entspricht, im Feld **Stand-Umsatz-Zaehler-AES256-ICM** ablegt.
- **Berechnung des Verkettungswerts:** Für den Verkettungswert wird der vorhergehende Beleg herangezogen. Siehe Prozess 2.4.2. Als Ergebnis erhält man den Wert des Felds **Sig-Voriger-Beleg**.

Beispiele

Beispiele für die unterschiedliche Behandlung der Belegtypen werden in den jeweiligen Detailprozessen gegeben.

Referenzen – Muster-Code

- Klasse**:**
  - `at.asitplus.regkassen.core.DemoCashBox`
  - https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/DemoCashBox.java
- Methode**:** *createStoreAndSignReceiptPackage*

Referenzen – RKSV

Relevante Referenzen werden bei den jeweiligen Detailprozessen genannt.

Ausgabewerte

- Daten für die Erstellung der Signatur (Eingabewert für Prozess 2.3)

2.2.5 Verarbeitung von Trainingsbelegen

Eingabewerte

Eingabewerte für Felder laut Prozess 2.1, die aus Belegdaten aufbereitet werden:
• **Beleg-Satz-Normal**
• **Beleg-Satz-Ermaessigt-1**
• **Beleg-Satz-Ermaessigt-2**
• **Beleg-Satz-Null**
• **Beleg-Satz-Besonders**
 	
Eingabewerte die aus der Kasse extrahiert werden:
• Felder laut Prozess 2.1:
  ○ **Kassen-ID**
  ○ **Belegnummer**
  ○ **Beleg-Datum-Uhrzeit**
  ○ **Zertifikat-Seriennummer**

Prozessbeschreibung
Diese Belege können zu Trainings- oder Testzwecken beliebig erstellt werden ohne den Umsatzzähler der Kasse zu beeinflussen. Werte für Felder laut Prozess 2.1, die berechnet werden müssen: • **Sig-Voriger-Beleg** Es werden nun folgende Prozesse durchgeführt: • **Aufbereiten/Aktualisieren des Umsatzzählers:** Der Umsatzzähler der Kasse wird von Trainingsbelegen nicht beeinflusst. • **Verschlüsselung des Umsatzzählers:** Beim Trainingsbeleg wird anstelle des verschlüsselten Umsatzzählers die BASE64-Kodierung der Zeichenkette `TRA`, welche der Zeichenkette `VFJB` entspricht im Feld **Stand-Umsatz-Zaehler-AES256-ICM** ablegt. • **Berechnung des Verkettungswerts:** Für den Verkettungswert wir der vorhergehende Beleg herangezogen. Siehe Prozess 2.4.2, als Ergebnis erhält man den Wert des Felds **Sig-Voriger-Beleg**.
Beispiele
Beispiele für die unterschiedliche Behandlung der Belegtypen werden in den jeweiligen Detailprozessen gegeben.
Referenzen – Muster-Code

• Klasse**:**
  ○ `at.asitplus.regkassen.core.DemoCashBox`
  ○ https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/DemoCashBox.java
• Methode**:** *createStoreAndSignReceiptPackage*

Referenzen – RKSV
Relevante Referenzen werden bei den jeweiligen Detailprozessen genannt.
Ausgabewerte
• Daten für die Erstellung der Signatur (Eingabewert für Prozess 2.3)

2.3 Aufbereiten der zu signierenden Daten

Eingabewerte

- Daten für die Erstellung der Signatur (Ausgabewert je nach verarbeitetem Beleg: siehe Prozess 2.2)

Prozessbeschreibung

Die im vorigen Prozess aufbereiteten Belegdaten stellen die Basis für die Erstellung der zu signierenden Daten dar, die dann anhand des JWS-Standards signiert werden.

Für die Aufbereitung der zu signierenden Daten wird wie folgt vorgegangen:

- **Zusammenfügen der Belegdaten**: Die Werte, der im vorigen Schritt definierten Belegdaten, werden über das Zeichen `_` zusammengefügt (Die Zeilenumbrüche wurden zur besseren Lesbarkeit eingefügt):

```
Kassen-ID
_Belegnummer
_Beleg-Datum-Uhrzeit
_Betrag-Satz-Normal
_Betrag-Satz-Ermaessigt-1
_Betrag-Satz-Ermaessigt-2
_Betrag-Satz-Null
_Betrag-Satz-Besonders
_Stand-Umsatz-Zaehler-AES256-ICM
_Zertifikat-Seriennummer
_Sig-Voriger-Beleg
```

- **Hinzufügen des Registrierkassenalgorithmuskennzeichen als Präfix**: Die resultierende Zeichenkette wird anschließend mit dem Präfix `_RKA_` ergänzt. `RKA` stellt in dieser Beschreibung einen Platzhalter für das Registrierkassenalgorithmuskennzeichen dar.

Beispiel – Standardbeleg (siehe Prozess 2.2.1)

Aufbereitete Daten:

- **Kassen-ID:** `DEMO-CASH-BOX524`
- **Belegnummer:** `366585AB`
- **Beleg-Datum-Uhrzeit:** `2015-12-17T11:23:43`
- **Betrag-Satz-Normal:** `10,50`
- **Betrag-Satz-Ermaessigt-1:** `0,00`
- **Betrag-Satz-Ermaessigt-2**: `0,00`
- **Betrag-Satz-Null:** `0,78`
- **Betrag-Satz-Besonders:** `0,00`
- **Stand-Umsatz-Zaehler-AES256-ICM:** Für das Beispiel wurde ein 8-Bytes langer Umsatzzähler verwendet. Ein Beispiel für einen möglichen Umsatzzähler in BASE64 Kodierung wäre (abhängig von Schlüssel und Umsatzwert): `Q+dTEnSc`
- **Zertifikat-Seriennummer:** Bsp. Seriennummer eines Zertifikats in Hexadezimaldarstellung `245abcde`
- **Sig-Voriger-Beleg**: Es wird ein Beispiel für einen möglichen Verkettungswert gegeben. Dabei entsprechen die BASE64-Kodierung und Länge einem echten Verkettungswert. Der tatsächliche Wert hängt vom vorigen Beleg ab, der für dieses Beispiel nicht angegeben wird. Bsp. Wert: `OJ16FcqeA7s`

Aufbereitung der zu signierenden Daten – Zusammenfügen der Belegdaten:
(Für die bessere Lesbarkeit wurden Zeilenumbrüche eingefügt):

```
DEMO-CASH-BOX524
_366585AB
_2015-12-17T11:23:43
_10,50
_0,00
```

Formatiert: Muster: Transparent (Hintergrund 1), Nicht Hervorheben

Formatiert: Muster: Transparent (Hintergrund 1)

```
_0,00
_0,78
_0,00
_Q+dTEnSc
_245abcde
_OJ16FcqeA7s
```

Formatiert: Schriftart: Courier, Nicht Hervorheben

Aufbereitung der zu signierenden Daten –
Hinzufügen des Registrierkassenalgorithmuskennzeichen als Präfix:
Zu der im vorigen Schritt erstellten Zeichenkette wird das jeweilige Registrierkassenalgorithmuskennzeichen hinzugefügt. In diesem Beispiel wird `R1-AT75` verwendet. `R1` entspricht dem zum Zeitpunkt der Erstellung dieses Dokuments einzig verfügbaren Kennzeichen. `AT75` steht für einen fiktiven österreichischen ZDA. Daraus ergibt sich (für die bessere Lesbarkeit wurden Zeilenumbrüche eingefügt):

```
_R1-AT75
_DEMO-CASH-BOX524
_366585AB
_2015-12-17T11:23:43
_10,50
_0,00
_0,00
_0,78
_0,00
_Q+dTEnSc
_245abcde
_OJ16FcqeA7s
```

Formatiert: Muster: Transparent (Hintergrund 1)

Beispiel – Startbeleg (siehe Prozess 2.2.2)

Aufbereitete Daten:

- **Kassen-ID:** `DEMO-CASH-BOX524`
- **Belegnummer:** `366585AB`
- **Beleg-Datum-Uhrzeit:** `2015-12-17T11:23:43`
- **Betrag-Satz-Normal:** beim Startbeleg per Definition `0,00`
- **Betrag-Satz-Ermaessigt-1:** beim Startbeleg per Definition `0,00`
- **Betrag-Satz-Ermaessigt-2**: beim Startbeleg per Definition `0,00`
- **Betrag-Satz-Null:** beim Startbeleg per Definition `0,00`
- **Betrag-Satz-Besonders:** beim Startbeleg per Definition `0,00`
- **Stand-Umsatz-Zaehler-AES256-ICM:** Für das Beispiel wurde ein 8-Bytes langer Umsatzzähler verwendet, der aufgrund des Startbelegs auf `0` gesetzt wurde. Ein Beispiel für einen möglichen Umsatzzähler in BASE64 Kodierung wäre (abhängig von Schlüssel und Umsatzwert): `5/4fWv5/uhI=`
- **Zertifikat-Seriennummer:** Bsp. Seriennummer eines Zertifikats in Hexadezimaldarstellung `245abcde`
- **Sig-Voriger-Beleg**: Der Verkettungswert wird beim Startbeleg anhand der **Kassen-ID** berechnet. Für die in diesem Beispiel verwendeten **Kassen-ID** mit dem Wert `DEMO-CASH-BOX524` entspricht dies dem Verkettungswert: `lDUkNhEeJKY=`

Aufbereitung der zu signierenden Daten – Zusammenfügen der Belegdaten:
(Für die bessere Lesbarkeit wurden Zeilenumbrüche eingefügt):

```
DEMO-CASH-BOX524
_366585AB
_2015-12-17T11:23:43
_0,00
_0,00
_0,00
```

Formatiert: Muster: Transparent (Hintergrund 1)

```
_0,00
_0,00
_5/4fWv5/uhI=
_245abcde
_lDUkNhEeJKY=
```

Aufbereitung der zu signierenden Daten –
Hinzufügen des Registrierkassenalgorithmuskennzeichen als Präfix:
Zu der im vorigen Schritt erstellten Zeichenkette wird das jeweilige Registrierkassenalgorithmuskennzeichen hinzugefügt. In diesem Beispiel wird `R1-AT75` verwendet. `R1` entspricht dem zum Zeitpunkt der Erstellung dieses Dokuments einzig verfügbaren Kennzeichen. `AT75` steht für einen fiktiven österreichischen ZDA. Daraus ergibt sich (für die bessere Lesbarkeit wurden Zeilenumbrüche eingefügt):

```
_R1-AT75
_DEMO-CASH-BOX524
_366585AB
_2015-12-17T11:23:43
_0,00
_0,00
_0,00
_0,00
_0,00
_5/4fWv5/uhI=
_245abcde
_lDUkNhEeJKY=
```

Formatiert: Muster: Transparent (Hintergrund 1)

Formatiert: Schriftart: Courier

Beispiel – Nullbeleg (siehe Prozess 2.2.3)

Aufbereitete Daten:

- **Kassen-ID:** `DEMO-CASH-BOX524`
- **Belegnummer:** `366585AB`
- **Beleg-Datum-Uhrzeit:** `2015-12-17T11:23:43`
- **Betrag-Satz-Normal:** beim Nullbeleg per Definition `0,00`
- **Betrag-Satz-Ermaessigt-1:** beim Nullbeleg per Definition `0,00`
- **Betrag-Satz-Ermaessigt-2**: beim Nullbeleg per Definition `0,00`
- **Betrag-Satz-Null:** beim Nullbeleg per Definition `0,00`
- **Betrag-Satz-Besonders:** beim Nullbeleg per Definition `0,00`
- **Stand-Umsatz-Zaehler-AES256-ICM:** Für das Beispiel wurde ein 8-Bytes langer Umsatzzähler verwendet. Ein Beispiel für einen möglichen Umsatzzähler in BASE64 Kodierung wäre (abhängig von Schlüssel und Umsatzwert): `Q+dTEnSc`
- **Zertifikat-Seriennummer:** Bsp. Seriennummer eines Zertifikats in Hexadezimaldarstellung `245abcde`
- **Sig-Voriger-Beleg**: Es wird ein Beispiel für einen möglichen Verkettungswert gegeben. Dabei entsprechen die BASE64-Kodierung und Länge einem echten Verkettungswert. Der tatsächliche Wert hängt vom vorigen Beleg ab, der für dieses Beispiel nicht angegeben wird. Bsp. Wert: `OJ16FcqeA7s`

Aufbereitung der zu signierenden Daten – Zusammenfügen der Belegdaten:
(Für die bessere Lesbarkeit wurden Zeilenumbrüche eingefügt):

```
DEMO-CASH-BOX524
_366585AB
_2015-12-17T11:23:43
_0,00
_0,00
_0,00
_0,00
```

Formatiert: Muster: Transparent (Hintergrund 1)

```
_0,00
_Q+dTEnSc
_245abcde
_OJ16FcqeA7s
```

Formatiert: Schriftart: Courier, Nicht Hervorheben

Aufbereitung der zu signierenden Daten –
Hinzufügen des Registrierkassenalgorithmuskennzeichen als Präfix:
Zu der im vorigen Schritt erstellten Zeichenkette wird das jeweilige Registrierkassenalgorithmuskennzeichen hinzugefügt. In diesem Beispiel wird `R1-AT75` verwendet. `R1` entspricht dem zum Zeitpunkt der Erstellung dieses Dokuments einzig verfügbaren Kennzeichen. `AT75` steht für einen fiktiven österreichischen ZDA. Daraus ergibt sich (für die bessere Lesbarkeit wurden Zeilenumbrüche eingefügt):

```
_R1-AT75
_DEMO-CASH-BOX524
_366585AB
_2015-12-17T11:23:43
_0,00
_0,00
_0,00
_0,00
_0,00
_Q+dTEnSc
_245abcde
_OJ16FcqeA7s
```

Formatiert: Muster: Transparent (Hintergrund 1)

Beispiel – Stornobeleg (siehe Prozess 2.2.4)

Aufbereitete Daten:

- **Kassen-ID:** `DEMO-CASH-BOX524`
- **Belegnummer:** `366585AB`
- **Beleg-Datum-Uhrzeit:** `2015-12-17T11:23:43`
- **Betrag-Satz-Normal:** `-5,00`
- **Betrag-Satz-Ermaessigt-1:** `0,00`
- **Betrag-Satz-Ermaessigt-2**: `0,00`
- **Betrag-Satz-Null:** `0,00`
- **Betrag-Satz-Besonders:** `0,00`
- **Stand-Umsatz-Zaehler-AES256-ICM:** Beim Stornobeleg wird der Umsatzzähler in der Kasse adaptiert, der verschlüsselter Wert wird aber nicht im Beleg gespeichert. Stattdessen wird der Wert `U1RP` (Ergebnis BASE64-Kodierung `STO`) im Feld gespeichert.
- **Zertifikat-Seriennummer:** Bsp. Seriennummer eines Zertifikats in Hexadezimaldarstellung `245abcde`
- **Sig-Voriger-Beleg**: Es wird ein Beispiel für einen möglichen Verkettungswert gegeben. Dabei entsprechen die BASE64-Kodierung und Länge einem echten Verkettungswert. Der tatsächliche Wert hängt vom vorigen Beleg ab, der für dieses Beispiel nicht angegeben wird. Bsp. Wert: `OJ16FcqeA7s`

Aufbereitung der zu signierenden Daten – Zusammenfügen der Belegdaten:
(Für die bessere Lesbarkeit wurden Zeilenumbrüche eingefügt):

```
DEMO-CASH-BOX524
_366585AB
_2015-12-17T11:23:43
_-5,00
_0,00
_0,00
_0,00
_0,00
```

Formatiert: Muster: Transparent (Hintergrund 1)

```
_U1RP
_245abcde
_OJ16FcqeA7s
```

Formatiert: Schriftart: Courier, Englisch (USA), Nicht Hervorheben

Aufbereitung der zu signierenden Daten –
Hinzufügen des Registrierkassenalgorithmuskennzeichen als Präfix:
Zu der im vorigen Schritt erstellten Zeichenkette wird das jeweilige Registrierkassenalgorithmuskennzeichen hinzugefügt. In diesem Beispiel wird `R1-AT75` verwendet. `R1` entspricht dem zum Zeitpunkt der Erstellung dieses Dokuments einzig verfügbaren Kennzeichen. `AT75` steht für einen fiktiven österreichischen ZDA. Daraus ergibt sich (für die bessere Lesbarkeit wurden Zeilenumbrüche eingefügt):

```
_R1-AT75
_DEMO-CASH-BOX524
_366585AB
_2015-12-17T11:23:43
_-5,00
_0,00
_0,00
_0,00
_0,00
_U1RP
_245abcde
_OJ16FcqeA7s
```

Formatiert: Muster: Transparent (Hintergrund 1)

Formatiert: Schriftart: Courier

Beispiel – Trainingsbeleg (siehe Prozess 2.2.5)

Aufbereitete Daten:

- **Kassen-ID:** `DEMO-CASH-BOX524`
- **Belegnummer:** `366585AB`
- **Beleg-Datum-Uhrzeit:** `2015-12-17T11:23:43`
- **Betrag-Satz-Normal:** `5,00`
- **Betrag-Satz-Ermaessigt-1:** `0,00`
- **Betrag-Satz-Ermaessigt-2**: `9,00`
- **Betrag-Satz-Null:** `13,30`
- **Betrag-Satz-Besonders:** `0,00`
- **Stand-Umsatz-Zaehler-AES256-ICM:** Beim Trainingsbeleg wird weder der Umsatzzähler der Kasse adaptiert, noch ein verschlüsselter Wert im Beleg abgelegt. Stattdessen wird der Wert `VFJB` (Ergebnis BASE64-Kodierung `TRA`) im Feld gespeichert.
- **Zertifikat-Seriennummer:** Bsp. Seriennummer eines Zertifikats in Hexadezimaldarstellung `245abcde`
- **Sig-Voriger-Beleg**: Es wird ein Beispiel für einen möglichen Verkettungswert gegeben. Dabei entsprechen die BASE64-Kodierung und Länge einem echten Verkettungswert. Der tatsächliche Wert hängt vom vorigen Beleg ab, der für dieses Beispiel nicht angegeben wird. Bsp. Wert: `OJ16FcqeA7s`

Aufbereitung der zu signierenden Daten – Zusammenfügen der Belegdaten:
(Für die bessere Lesbarkeit wurden Zeilenumbrüche eingefügt):

```
DEMO-CASH-BOX524
_366585AB
_2015-12-17T11:23:43
_5,00
_0,00
_9,00
_13,30
_0,00
_VFJB
```

Formatiert: Muster: Transparent (Hintergrund 1)

Formatiert: Englisch (USA), Nicht Hervorheben

```
_245abcde
_OJ16FcqeA7s
```

Aufbereitung der zu signierenden Daten –
Hinzufügen des Registrierkassenalgorithmuskennzeichen als Präfix:
Zu der im vorigen Schritt erstellten Zeichenkette wird das jeweilige Registrierkassenalgorithmuskennzeichen hinzugefügt. In diesem Beispiel wird `R1-AT75` verwendet. `R1` entspricht dem zum Zeitpunkt der Erstellung dieses Dokuments einzig verfügbaren Kennzeichen. `AT75` steht für einen fiktiven österreichischen ZDA. Daraus ergibt sich (für die bessere Lesbarkeit wurden Zeilenumbrüche eingefügt):

```
_R1-AT75
_DEMO-CASH-BOX524
_366585AB
_2015-12-17T11:23:43
_5,00
_0,00
_9,00
_13,30
_0,00
_VFJB
_245abcde
_OJ16FcqeA7s
```

Formatiert: Muster: Transparent (Hintergrund 1)

Formatiert: Nicht Hervorheben

Formatiert: Schriftart: Courier

Referenzen – Muster-Code – Aufbereitung der zu signierenden Daten

- Klasse:
  - `at.asitplus.regkassen.core.base.receiptdata.ReceiptRepresentationForSignature`
  - https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/base/receiptdata/ReceiptRepresentationForSignature.java
- Methode: *getDataToBeSigned*

Referenzen – RKSV

- Z 5 zur Anlage der RKSV

Ausgabewerte

- Aufbereitete Daten für die Erstellung der Signatur anhand des JWS-Standards (Eingabewert für Prozess 3).

2.4 Detailprozesse für die Verkettung von Belegen

- 2.4.1 Verkettung beim Startbeleg
- 2.4.2 Verkettung bei allen Belegen außer dem Startbeleg

2.4.1 Verkettung beim Startbeleg

Eingabewerte

- Kassen-ID

Prozessbeschreibung

Belege sind untereinander kryptographisch verkettet, um das spätere Einfügen oder Entfernen einzelner Belege zu verhindern.

Da beim Startbeleg noch kein vorhergehender Beleg vorhanden ist, der für den Verkettungswert herangezogen werden kann, wird die **Kassen-ID** verwendet, aus der der initiale Verkettungswert berechnet wird. Dabei wird wie folgt vorgegangen:

- **Eingabewert für Hashalgorithmus**: Dies ist die UTF8-kodierte Zeichenkette des Felds **Kassen-ID.**
- **Anwenden des Hashalgorithmus**: Es wird die im Registrierkassenalgorithmuskennzeichen angegebene Hash-Funktionen für die Berechnung des Hash-Werts angewendet.
- **Extraktion des Verkettungswerts**: Aus dem Ergebnis des Hashalgorithmus werden `M` Bytes startend mit Byte `0` extrahiert. `M` ist ebenfalls im Registrierkassenalgorithmuskennzeichen definiert.
- **Kodierung des Verkettungswerts**: Der extrahierte Verkettungswert wird BASE64-kodiert und im Feld **Sig-Voriger-Beleg** des zu erstellenden maschinenlesbaren Codes abgelegt.

Beispiel – Berechnen des Verkettungswerts

Der Verkettungswert für den Startbeleg wird bei einer Kasse mit der **Kassen-ID** `A12345~~5~~7` wie folgt berechnet:

- **Eingabewert für Hashalgorithmus**: Die Kassen-ID – in diesem Fall `A12347` ist im UTF8-Format gespeichert.
- **Anwenden des Hashalgorithmus/Extraktion des Verkettungswerts**: Die **Kassen-ID** `A12347` ist der Eingabewert für den Hashalgorithmus SHA-256 (im Falle von `RK1`). Aus dem Ergebnis werden 8 Bytes/64 Bits extrahiert (im Falle von `RK1`).
- **Kodierung des Verkettungswerts:** Des extrahierte Wert wird BASE64-Kodiert. Dies entspricht in diesem Beispiel dem Wert `OeSKQjO4zKI=`, der im Feld **Sig-Voriger-Beleg** abgelegt wird.

Referenzen – Muster-Code – Berechnen des Verkettungswerts

- Klasse**:**
  - `at.asitplus.regkassen.core.DemoCashBox`
  - https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/DemoCashBox.java
- Methode**:** *calculateChainValue*

Referenzen – RKSV

- Z 4 zur Anlage der RKSV, Definition des Felds **Sig-Voriger-Beleg**
- Z 2 zur Anlage der RKSV, Definition des "**Hashalgorithmus für die Verkettung der Belege und Berechnung des IVs, Anzahl N der extrahierten Bytes"**

Ausgabewerte

- Verkettungswert für die Ablage im Feld „**Sig-Voriger-Beleg**"

2.4.2 Verkettung bei allen Belegen außer dem Startbeleg

Eingabewerte

- Kompakte JWS-Repräsentation des vorigen Belegs

Prozessbeschreibung

Bei allen Belegtypen – außer dem Startbeleg – wird für die Berechnung des Verkettungswerts wie folgt vorgegangen:

- **Eingabewert für Hashalgorithmus**: Als Eingabewert wird die kompakte Repräsentation der JWS-Signatur des vorhergehenden Beleges herangezogen (siehe Ausgabewert der Prozesse 3.1 oder 3.2).
- **Weitere Prozessschritte**: Die weiteren Prozessschritte entsprechen den korrespondierenden Schritten bei der Erstellung des Startbelegs (siehe Prozess 2.4.1).

Beispiele

Für die Berechnung des Verkettungswerts wird der folgende (zuletzt erstellte Beleg) herangezogen:

- **Maschinenlesbarer Code**: (Zeilenumbrüche aufgrund von Lesbarkeit):


```
_R1-AT0_DEMO-CASH-BOX524
_366587
_2015-12-17T11:23:44_34,77
_59,64
_38,13
_0,00
_0,00
_8MG8C1Kr7HA=
_20f2ed172daa09e5
_xTfZvkBSTr4=
_GeWps9kci+fUqKLymS1pHlIbv0L8Oek+v6TDmZj9Ffucb8yvSijqZ8LcBalV9lADM
XQ8U3itViKkd/i1Ba22BA==
```


- **Eingabewert für Hashalgorithmus – Kompakte Repräsentation der JWS-Signatur:** (Zeilenumbrüche aufgrund von Lesbarkeit):


```
eyJhbGciOiJFUzI1NiJ9.
X1IxLUFUMF9ERU1PLUNBU0gtQk9YNTI0XzM2NjU4N18yMDE1LTEyLTE3VDExOjIzOj
Q0XzM0LDc3XzU5LDY0XzM4LDEzXzAsMDBfMCwwMF84TUc4QzFLcjdIQT1fMjBmMmVk
MTcyZGFhMDllNV94VGZadmtCU1RyND0.
GeWps9kci-fUqKLymS1pHlIbv0L8Oek-v6TDmZj9Ffucb8yvSijqZ8LcBalV9lADMX
Q8U3itViKkd_i1Ba22BA
```


- **Anwenden des Hashalgorithmus/Extraktion des Verkettungswerts**: Die kompakte Repräsentation der JWS-Signatur ist der Eingabewert für den Hashalgorithmus SHA-256 (im Falle von `RK1`). Aus dem Ergebnis werden 8 Bytes/64 Bits extrahiert (im Falle von `RK1`).
- **Kodierung des Verkettungswerts:** Des extrahierte Wert wird BASE64-Kodiert. Dies entspricht in diesem Beispiel dem Wert `5HjRCx+XIz4=`, der im Feld **Sig-Voriger-Beleg** abgelegt wird.

Referenzen – Muster-Code – Berechnen des Verkettungswerts

- Klasse:
  - `at.asitplus.regkassen.core.DemoCashBox`
  - https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/DemoCashBox.java
- Methode: *calculateChainValue*

Referenzen – RKSV

- Z 4 zur Anlage der RKSV, Definition des Felds **Sig-Voriger-Beleg**
- Z 2 zur Anlage der RKSV, Definition des "**Hashalgorithmus für die Verkettung der Belege und Berechnung des IVs, Anzahl N der extrahierten Bytes"**

Ausgabewerte

- Verkettungswert für die Ablage im Feld „**Sig-Voriger-Beleg**"

Formatierte Tabelle

Formatiert: Muster: Transparent (Hintergrund 1)

Formatiert: Muster: Transparent (Hintergrund 1)

2.5 Detailprozesse für die Verarbeitung des Umsatzzählers

Für die Verschlüsselung des Umsatzzählers wird der AES-Algorithmus im ICM/CTR Modus mit einer Schlüssellänge von 256 Bit verwendet. Der Einsatz des ICM/CTR Modus erlaubt es das verschlüsselte Ergebnis in einer kürzeren Darstellung abzulegen als durch die in den anderen Modi vorgegebenen Block-Länge von 128 Bit. Der ICM/CTR Modus kann durch die Verwendung des ECB oder CFB Modus emuliert werden. Dies ist vor allem dann relevant wenn die verwendete Architektur (Bibliothek) nicht über den ICM/CTR Modus verfügt.

Der Verschlüsselungsprozess ist in den folgenden Prozessen dargestellt und wird wie folgt unterteilt:

- 2.5.1 Aufbereiten/Aktualisieren des Umsatzzählers
- 2.5.2 Aufbereiten der Daten für den Verschlüsselungsprozess
- 2.5.3 Verschlüsselung mit ICM/CTR Modus
- 2.5.4 Verschlüsselung durch Emulation des ICM/CTR Modus über ECB/CFB Modus
- 2.5.5 Aufbereitung des verschlüsselten Umsatzzählers

2.5.1 Aufbereiten/Aktualisieren des Umsatzzählers

Eingabewerte

- Aktueller Umsatzzähler der jeweiligen Kasse
- Werte der jeweiligen summierten Beträge der Steuersätze des vorhandenen Belegs

Prozessbeschreibung

- **Format des Umsatzzählers**: Der Umsatzzähler wird in €-Cent dargestellt.
- **Aktualisierung des Umsatzzählers**: Die Beträge der Steuersätze des zu erstellenden Belegs werden in €-Cent Beträge konvertiert, und aufsummiert. Das Ergebnis, in weiterer Folge als „*Summe-Steuer-Sätze*“ bezeichnet, beeinflusst abhängig vom Belegtyp den Umsatzzähler wie folgt:
  - **Standardbeleg/Stornobeleg**: Der Wert „*Summe-Steuer-Sätze*“ wird zum Umsatzzähler der Kasse addiert.
  - **Alle anderen Belegtypen**: Bei allen anderen Belegtypen (Startbeleg, Nullbeleg, Trainingsbeleg) wird der Umsatzzähler der Kasse nicht beeinflusst.
- **Setzen des neuen Umsatzzählers in der Kasse**: Der Umsatzzähler der Kasse wird mit dem neuen Wert aktualisiert.

Beispiel – Standardbeleg – typische Buchungen

Die summierten Beträge der Steuersätze der Bruttowerte der Positionen eines zu erstellenden Belegs sind wie folgt gegeben:

- **Beleg-Satz-Normal:** 24,54 € (entspricht dem Wert `24,54`)
- **Beleg-Satz-Ermaessigt-1:** 0 € (entspricht dem Wert `0,00`)
- **Beleg-Satz-Ermaessigt-2:** 3.7 € (entspricht dem Wert `3,70`)
- **Beleg-Satz-Null:** 5,0 € (entspricht dem Wert `5,00`)
- **Beleg-Satz-Besonders:** 0 € (entspricht dem Wert `0,00`)

Der Umsatzzähler der Kasse zum Zeitpunkt der Belegerstellung: `5340,45` €, dies entspricht dem Wert `534045` in €-Cent.

- **Summieren der Beträge der Steuersätze**: Die einzelnen Beträge der Steuersätze werden
  - summiert `24,54 + 0,00 + 3,70 + 5,00 + 0,00 = 33,24`
    und in €-Cent umgewandelt: `3324` oder
  - zuerst in €-Cent umgewandelt und dann summiert:
    `2454 + 0 + 370 + 500 + 0 = 3324`
- **Adaptieren des Umsatzzählers der Kasse**: Der im vorigen Schritt berechnete Wert wird zum Umsatzzähler der Kasse addiert: `534045 + 3324 = 537369`. Dies entspricht einem aktualisierten Umsatzzähler mit dem Wert `537369` €-Cent bzw. `5373,69` €. Der aktualisierte Umsatzzähler wird als Eingabewert für den Verschlüsselungsprozess, der im Zuge der Belegerstellung durchgeführt wird, verwendet.

Beispiel – Standardbeleg – Buchungen mit Rabatt/Gutschrift

Die summierten Beträge der Steuersätze der Bruttowerte der Positionen eines zu erstellenden Belegs sind wie folgt gegeben:

- **Beleg-Satz-Normal:** 24,54 € (entspricht dem Wert `24,54`)
- **Beleg-Satz-Ermaessigt-1:** 0 € (entspricht dem Wert `0,00`)
- **Beleg-Satz-Ermaessigt-2:** 3.7 € (entspricht dem Wert `3,70`)
- **Beleg-Satz-Null:** -5,0 € (entspricht dem Wert `-5,00`), in diesem Beispiel ist der Wert negativ, da es sich um eine Gutschrift/Rabatt handelt. Es handelt sich um keinen Stornowert.
- **Beleg-Satz-Besonders:** 0 € (entspricht dem Wert `0,00`)

Der Umsatzzähler der Kasse zum Zeitpunkt der Belegerstellung: `5340,45` €, dies entspricht dem Wert `534045` in €-Cent.

- **Summieren der Beträge der Steuersätze**: Die einzelnen Beträge der Steuersätze werden
  - summiert `24,54 + 0,00 + 3,70 + (-5,00) + 0,00 = 23,24`

und in €-Cent umgewandelt: `2324` oder
  - zuerst in €-Cent umgewandelt und dann summiert: `2454 + 0 + 370 + (-500) + 0 = 2324`
- **Adaptieren des Umsatzzählers der Kasse**: Der im vorigen Schritt berechnete Wert wird zum Umsatzzähler der Kasse addiert: `534045 + 2324 = 536369`. Dies entspricht einem aktualisierten Umsatzzähler mit dem Wert `536369` €-Cent bzw. `5363,69` €. Der aktualisierte Umsatzzähler wird als Eingabewert für den Verschlüsselungsprozess, der im Zuge der Belegerstellung durchgeführt wird, verwendet.

Beispiel – Stornobeleg – Storno einer Buchung

In diesem Beispiel wird eine vorher getätigte Buchung storniert. Ein Stornobeleg darf nicht mit Standardbuchungen vermischt werden, es können aber mehrere Buchungen mit unterschiedlichen Steuersätzen gleichzeitig in einem Beleg storniert werden. Die summierten Beträge der Steuersätze der Bruttowerte der Positionen eines zu erstellenden Belegs sind wie folgt gegeben:

- **Beleg-Satz-Normal: -**24,54 € (entspricht dem Wert `-24,54`)
- **Beleg-Satz-Ermaessigt-1:** -2 € (entspricht dem Wert `-2,00`)
- **Beleg-Satz-Ermaessigt-2:** 0 € (entspricht dem Wert `0,00`)
- **Beleg-Satz-Null:** 0 € (entspricht dem Wert `0,00`)
- **Beleg-Satz-Besonders:** 0 € (entspricht dem Wert `0,00`)

Der Umsatzzähler der Kasse zum Zeitpunkt der Belegerstellung: `5340,45` €, dies entspricht dem Wert `534045` in €-Cent.

- **Summieren der Beträge der Steuersätze**: Die einzelnen Beträge der Steuersätze werden
  - summiert `-24,54 + -2,00 + 0,00 + 0,00 + 0,00 = -26,54` und in €-Cent umgewandelt: `-2654` oder
  - zuerst in €-Cent umgewandelt und dann summiert: `-2454 + -200 + 0 + 0 + 0 = -2654`
- **Adaptieren des Umsatzzählers der Kasse**: Der im vorigen Schritt berechnete Wert wird zum Umsatzzähler der Kasse addiert: `534045 + (-2654) = 531391`. Dies entspricht einem aktualisierten Umsatzzähler mit dem Wert `531391` €-Cent bzw. `5313,91` €. Der ~~aktualisierte~~ Umsatzzähler wird in der Kasse adaptiert. Die Aufbereitung~~als Eingabewert~~ für den Verschlüsselungsprozess ist nicht notwendig, da~~der im Zuge~~ der verschlüsselte Umsatzzähler bei einem Stornobeleg nicht in den maschinenlesbaren Code übernommen~~Belegerstellung durchgeführt~~ wird~~ verwendet~~.

Beispiel – Stornobeleg – Storno einer Gutschrift/eines Rabatts

In diesem Beispiel wird eine vorher negative Buchung (aufgrund z.B. einer Gutschrift/Rabatt) storniert. Daher ergibt sich hier für den jeweiligen Betrag des Steuersatzes ein positiver Wert trotz Storno. Die Identifizierung des Belegs als Stornobelegs ist dennoch über die Kennzeichnung `STO` möglich. Gleich wie bei den anderen Stornobelegen muss garantiert sein, dass keine Vermischung mit Standardbuchungen erfolgt. Das Stornieren von mehreren Buchungen mit unterschiedlichen Steuersätzen gleichzeitig ist hingegen möglich. Die summierten Beträge der Steuersätze der Bruttowerte der Positionen eines zu erstellenden Belegs sind wie folgt gegeben:

- **Beleg-Satz-Normal:** 24,54 € (entspricht dem Wert `24,54`)
- **Beleg-Satz-Ermaessigt-1:** 0 € (entspricht dem Wert `0,00`)
- **Beleg-Satz-Ermaessigt-2:** 0 € (entspricht dem Wert `0,00`)
- **Beleg-Satz-Null:** 0 € (entspricht dem Wert `0,00`)
- **Beleg-Satz-Besonders:** 0 € (entspricht dem Wert `0,00`)

Der Umsatzzähler der Kasse zum Zeitpunkt der Belegerstellung: `5340,45` €, dies entspricht dem Wert `534045` in €-Cent.

- **Summieren der Beträge der Steuersätze**: Die einzelnen Beträge der Steuersätze werden summiert
  - `24,54 + 0,00 + 0,00 + 0,00 + 0,00 = 24,54` und in €-Cent

umgewandelt: `2454` oder
    - zuerst in €-Cent umgewandelt und dann summiert: `2454 + 0 + 0 + 0 + 0 = 2454`
- **Adaptieren des Umsatzzählers der Kasse**: Der im vorigen Schritt berechnete Wert wird beim Stornobeleg vom Umsatzzähler der Kasse addiert: `534045 + 2454 = 536499`. Dies entspricht einem aktualisierten Umsatzzähler mit dem Wert `536499` €-Cent bzw. `5364,99` €. Der aktualisierte Umsatzzähler wird als Eingabewert für den Verschlüsselungsprozess der im Zuge der Belegerstellung durchgeführt wird verwendet.

Beispiel – Stornobeleg – Storno einer Buchung bzw. Rabatt/Gutschrift mit negativem Umsatzzähler

In der Realität wird dieses Beispiel nur in seltenen Fällen auftreten. Dennoch zeigt es, dass in bestimmten Fällen der Umsatzzähler der Kasse negativ sein kann. Der Fall tritt z.B. ein wenn mit einer kürzlich initialisierten Kasse ein Storno einer früheren Buchung (andere Kasse, oder Neuinitialisierung der Kasse aufgrund eines Ausfalls) durchgeführt wird. Der Fall kann auch dann eintreten, wenn eine Gutschrift/ein Rabatt mit einer kürzlich initialisierten Kasse verbucht wird.
Die summierten Beträge der Steuersätze der Bruttowerte der Positionen eines zu erstellenden Belegs sind wie folgt gegeben:

- **Beleg-Satz-Normal: -**24,54 € (entspricht dem Wert `-24,54`)
- **Beleg-Satz-Ermaessigt-1:** 0 € (entspricht dem Wert `0,00`)
- **Beleg-Satz-Ermaessigt-2:** 0 € (entspricht dem Wert `0,00`)
- **Beleg-Satz-Null:** 0 € (entspricht dem Wert `0,00`)
- **Beleg-Satz-Besonders:** 0 € (entspricht dem Wert `0,00`)

Der Umsatzzähler der Kasse zum Zeitpunkt der Belegerstellung: `10,45` €, dies entspricht dem Wert `1045` in €-Cent.

- **Summieren der Beträge der Steuersätze**: Die einzelnen Beträge der Steuersätze werden summiert
    - `-24,54 + 0,00 + 0,00 + 0,00 + 0,00 = -24,54` und in €-Cent umgewandelt: `2454` oder
    - zuerst in €-Cent umgewandelt und dann summiert: `-2454 + 0 + 0 + 0 + 0 = -2454`
- **Adaptieren des Umsatzzählers der Kasse**: Der im vorigen Schritt berechnete Wert wird beim Stornobeleg vom Umsatzzähler der Kasse addiert: `1045 + (-2454) = -1409`. Dies entspricht einem aktualisierten Umsatzzähler mit dem Wert `-1409` €-Cent bzw. `-14,09` €. Der ~~aktualisierte~~ Umsatzzähler wird in der Kasse adaptiert. Die Aufbereitung~~als Eingabewert~~ für den Verschlüsselungsprozess ist nicht notwendig, da~~, der im Zuge~~ der verschlüsselte Umsatzzähler bei einem Stornobeleg nicht in den maschinenlesbaren Code übernommen~~Belegerstellung durchgeführt~~ wird~~, verwendet~~.

Referenzen – Muster-Code – Aktualisierung des Umsatzzählers

- Klasse**:**
    - `at.asitplus.regkassen.core.DemoCashBox`
    - https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/DemoCashBox.java
- Methoden**:** *createStoreAndSignReceiptPackage, updateTurnOverCounter*

Ausgabewerte

- Aktualisierter Umsatzzähler
    - wird in der Kasse gespeichert
    - wird als Eingabewert für die Berechnung des verschlüsselten Umsatzzählers verwendet

2.5.2 Aufbereiten der Daten für den Verschlüsselungsprozess

Eingabewerte

Eingabewerte die aus der Kasse extrahiert werden:

- Felder anhand von Prozess 2.1:
  - **Kassen-ID**
  - **Belegnummer**
- Weitere Daten:
  - **Umsatzzähler der Kasse**
  - **AES-Schlüssel der Kasse**

Prozessbeschreibung

Für das Durchführen des Verschlüsselungsprozesses müssen der Umsatzzähler und die Parameter für den AES-Algorithmus aufbereitet werden.

- **Kodierung des Umsatzzählers**: Die Block-Größe von AES-256 entspricht einem Byte-Array der Länge 16 (128 Bits). Für die Kodierung des Umsatzzählers im Klartext wird dabei ein Byte-Array der Länge 16 erstellt. Jedes Element des Byte-Arrays wird mit `0` initialisiert. Der Umsatzzähler mit der Byte-Anzahl `N` wird startend mit Byte `0` im BIG-ENDIAN Format als Zweier-Komplement Darstellung („signed") gespeichert. `N` entspricht der Anzahl der Bytes die für die Kodierung des Umsatzzählers notwendig sind. Es müssen mindestens 5 Bytes/40 Bits für den Umsatzzähler verwendet werden.
- **Initialisierungsvektor für AES (IV)**: Der Initialisierungsvektor (IV) für den Verschlüsselungsalgorithmus ist ein Byte-Array mit der Länge 16. Für die Berechnung des IVs werden die UTF-8 kodierte Kassenidentifikationsnummer (Wert des Feldes **Kassen-ID**" laut Prozess 2.1) und die UTF-8-kodierte Belegnummer (Wert des Feldes „**Belegnummer**" anhand von Prozess 2.1) in dieser Reihenfolge zusammengefügt. Das Ergebnis ist eine UTF-8 kodierte Zeichenkette, die als Eingabewert für die im Registrierkassenalgorithmuskennzeichen definierten Hashalgorithmus verwendet wird. Das Ergebnis der Hash-Funktion ist der Hash-Wert abgebildet in einem Byte-Array. Die Bytes `0-15` werden daraus extrahiert und als IV verwendet.
- **Festlegen der Länge des verschlüsselten Umsatzzählers**: Die minimale Länge eines Umsatzzählers entspricht 5 Bytes. Die Block-Größe des AES-Algorithmus entspricht 16 Bytes/128 Bits. Für die Darstellung in der QR/OCR Repräsentation auf dem Beleg soll die Länge des maschinenlesbaren Codes so kurz wie möglich sein. Da für die Darstellung des Umsatzes einer Kasse ein Bruchteil dieser Länge ausreicht, wird in der Spezifikation der AES ICM/CTR Modus verwendet, da es in diesem Modus möglich ist Teile des Ergebnisses des Verschlüsselungsprozesses zu extrahieren ohne dabei Informationen zu verlieren. Der Parameter „*Länge des verschlüsselten Umsatzzählers*" definiert wie viele Bytes für die Kodierung des Umsatzes benötigt werden. Die minimale Länge ist mit 5 Bytes definiert. Bei der gewählten Zweier-Komplement Darstellung und der Repräsentation des Umsatzzählers als €-Cent-Werte entspricht der maximal abbildbare Umsatz bei 5 Bytes noch immer über Fünf Milliarden € (2^40/2/100 = 5.497.558.138,9 €). Dieser Maximalwert sollte prinzipiell für alle Kassen ausreichend sein. Besteht dennoch die Notwendigkeit größere Umsatzzähler zu verwenden, kann auch eine größere Anzahl an Bytes extrahiert werden.

Anmerkungen:

- **ACHTUNG – Hinweis zur Sicherheit**: Es muss garantiert sein, dass für jede Verschlüsselungsoperation, die mit einem gegebenen AES-Schlüssel durchgeführt wird, niemals der gleiche IV verwendet wird. Dies bedeutet im Wesentlichen, dass für eine **Kassen-ID** jede Belegnummer nur einmal verwendet werden darf.

Referenzen – Muster-Code – Kodierung des Umsatzzählers und Aufbereiten des Initialisierungsvektors (IV)

- Klasse**:**
  - `at.asitplus.regkassen.core.DemoCashBox`
  - https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/DemoCashBox.java

• Methoden**:** *createStoreAndSignReceiptPackage, encryptTurnOverCounter*
Referenzen – RKSV
• Z 4 zur Anlage der RKSV, Definition des Felds **Stand-Umsatz-Zaehler-AES256-ICM** • Z 2 zur Anlage der RKSV, Definition des "**Hashalgorithmus für die Verkettung der Belege und Berechnung des IVs, Anzahl N der extrahierten Bytes"**
Ausgabewerte
• Kodierter Umsatzzähler für den weiteren Verschlüsselungsvorgang • Initialisierungsvektor (IV) für die Anwendung des Verschlüsselungsprozesses

2.5.3 Verschlüsselung mit ICM/CTR Modus

Eingabewerte

- Zu verschlüsselnder Umsatzzähler in €-Cent

Prozessbeschreibung

In diesem Fall wird davon ausgegangen, dass die in der jeweiligen Architektur verwendete Bibliothek für das Anwenden des AES-Algorithmus den ICM/CTR Modus unterstützt.

- **Initialisierung des AES-Algorithmus**: In diesem Fall werden der Bibliothek die folgenden Daten übergeben:
  - **IV**: Der im vorigen Schritt aufbereitete Initialisierungsvektor (16 Bytes/128 Bits Länge).
  - **Klartextdaten**: Diese entsprechen den im vorigen Schritt kodierten Umsatzzähler (16 Bytes/128 Bits Länge).
  - **AES-Schlüssel**: Hierbei handelt es sich um den **AES-Schlüssel der Kasse**.
  - **Padding**: Es wird kein Padding-Schema verwendet.
- **Verschlüsselung des Umsatzzählers**: Die im vorigen Schritt aufbereiteten *Klartextdaten* werden mit dem initialisierten AES-Algorithmus im ICM/CTR mit Hilfe der jeweiligen Bibliothek verschlüsselt. Das Ergebnis ist ein 16-Bytes/128 Bits langer Wert.

Referenzen – Muster-Code – Verschlüsselung des Umsatzzählers über ICM/CTR Modus

- Klasse**:**
  - `at.asitplus.regkassen.common~~core.base~~.util.CryptoUtil`
  - https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-common/src/main/java/at/asitplus/regkassen/common/util/CryptoUtil.java
  - ~~https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/base/util/CryptoUtil.java~~
- Methoden**:** *encryptCTR, decryptCTR*

Referenzen – RKSV

- Z 8 zur Anlage der RKSV
- Z 9 zur Anlage der RKSV
- Z 10 zur Anlage der RKSV

Ausgabewerte

- Verschlüsselter Umsatzzähler

Formatiert: Schriftfarbe: Grau-85 %

Formatiert: Schriftfarbe: Grau-85 %

2.5.4 *Verschlüsselung durch Emulation des ICM/CTR Modus über ECB/CFB Modus*

Eingabewerte

- Zu verschlüsselnder Umsatzzähler in €-Cent

Prozessbeschreibung

In diesem Fall wird davon ausgegangen, dass die in der jeweiligen Architektur verwendete Bibliothek für das Anwenden des AES-Algorithmus den ICM/CTR Modus NICHT unterstützt. Das ICM/CTR Verfahren kann aber über den CFB- oder ECB-Modus emuliert werden.
Die Daten (IV, Klartextdaten, AES-Schlüssel) werden gleich wie im ICM/CTR Modus aufbereitet.

Vorgehen für ECB-Modus:

- **Initialisierung des AES-Algorithmus**: Der AES Algorithmus wird mit dem ECB Modus ohne Padding-Schema und dem gegebenen AES-Schlüssel initialisiert. In diesem Modus verwendet der AES-Algorithmus keinen Initialisierungsvektor.
- **Verschlüsselung des im vorigen Prozess definierten IVs**: Der im Prozess 2.5.2 erstellte Initialisierungsvektor wird mit dem initialisierten AES-Algorithmus verschlüsselt. Das Ergebnis ist ein 16-Bytes/128 Bits langer Wert. Dieser Wert stellt den Schlüssel für die weitere Verschlüsselung des Umsatzzählers dar.
- ~~**Verschlüsseln des Umsatzzählers**: Die im vorigen Schritt aufbereiteten Klartextdaten werden mit dem initialisierten AES-Algorithmus im CFB-Modus mit Hilfe der jeweiligen Bibliothek verschlüsselt. Das Ergebnis ist ein 16-Bytes/128 Bits langer Wert.~~

~~***Vorgehen für CFB-Modus:***~~

- ~~**Initialisierung des AES-Algorithmus**: Der AES-Algorithmus wird mit dem CFB-Modus ohne Padding-Schema und dem gegeben AES-Schlüssel initialisiert. Als Initialisierungsvektor wird der im Prozess 2.5.2 erstellte IV verwendet.~~
- **Verschlüsseln des Umsatzzählers**: Die im vorigen Prozess aufbereiteten Klartextdaten (der kodierte Umsatzzähler) werden mit dem im vorigen Schritt berechneten Schüssel über die XOR-Operation verknüpft. Das Ergebnis entspricht dem verschlüsselten Umsatzzähler – ein 16-Bytes/128 Bits langer Wert.

Vorgehen für CFB-Modus:

- **Initialisierung des AES-Algorithmus**: Der AES Algorithmus wird mit dem CFB Modus ohne Padding-Schema und dem gegeben AES-Schlüssel initialisiert. Als Initialisierungsvektor wird der im Prozess 2.5.2 erstellte IV verwendet.
- **Verschlüsseln des Umsatzzählers**: Die im vorigen Schritt aufbereiteten Klartextdaten werden mit dem initialisierten AES-Algorithmus im CFB-Modus mit Hilfe der jeweiligen Bibliothek verschlüsselt. Das Ergebnis ist ein 16-Bytes/128 Bits langer Wert.

Referenzen – Muster-Code – Verschlüsselung des Umsatzzählers über ECB Modus

- Klasse:
  - `at.asitplus.regkassen.common~~core.base~~.util.CryptoUtil`
  - https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-common/src/main/java/at/asitplus/regkassen/common/util/CryptoUtil.java
  - ~~https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/base/util/CryptoUtil.java~~
- Methoden: *encryptECB, decryptECB*

Referenzen – Muster-Code – Verschlüsselung des Umsatzzählers über CFB Modus

- Klasse:
  - `at.asitplus.regkassen.common~~core.base~~.util.CryptoUtil`
  - https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-common/src/main/java/at/asitplus/regkassen/common/util/CryptoUtil.java
  - ~~https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-~~

Formatiert: Schriftart: Nicht Kursiv

Feldfunktion geändert

Formatiert: Schriftart: Nicht Kursiv

Formatiert: Schriftart: Nicht Kursiv

Formatiert: Schriftfarbe: Grau-85 %

Formatiert: Schriftfarbe: Grau-85 %

Formatiert: Schriftfarbe: Grau-85 %

Formatiert: Schriftfarbe: Grau-85 %

~~core/src/main/java/at/asitplus/regkassen/core/base/util/CryptoUtil.java~~ • Methoden**:** *encryptCFB, decryptCFB*
Referenzen – RKSV
• Z 8 zur Anlage der RKSV • Z 9 zur Anlage der RKSV • Z 10 zur Anlage der RKSV
Ausgabewerte
• Verschlüsselter Umsatzzähler

2.5.5 Aufbereitung des verschlüsselten Umsatzzählers

Eingabewerte

- Verschlüsselter Umsatzzähler

Prozessbeschreibung

Unabhängig vom vorher durchgeführten Prozess (ICM/CTR, CFB, ECB) entspricht das Ergebnis einem 16 Bytes/128 Bits langem Wert (der verschlüsselte Umsatzzähler). Um die Länge des in weiterer Folge aufbereiteten maschinenlesbaren Codes möglichst gering zu halten, werden vom verschlüsselten Umsatzzähler nur die Anzahl der im vorigen Prozess definierten „*Länge des verschlüsselten Umsatzzählers*“ (siehe Prozess [2]) extrahiert.

Dabei wird wie folgt vorgegangen: Das Resultat des Verschlüsselungsprozesses (Prozess 2.5.3 oder Prozess 2.5.4) entspricht dem verschlüsselten Umsatzzähler (16-Bytes/128 Bits langer Wert). Startend mit Byte `0` werden `N` Bytes aus dem Array extrahiert und BASE64-kodiert. `N` entspricht dabei der im vorigen Prozess definierten *Länge des verschlüsselten Umsatzzählers*. Der Minimalwert für `N` ist 5. Dies entspricht 5 Bytes/40 Bits, die aus dem verschlüsselten Umsatzzähler extrahiert werden.

Das Ergebnis ist der „*Verkürzter verschlüsselter Umsatzzähler für die Ablage im Feld „**Stand-Umsatz-Zaehler-AES256-ICM**~~maschinenlesbaren Code~~*“.

Referenzen – Muster-Code – Verschlüsselung des Umsatzzählers über ICM/CTR Modus

- Klasse:
  - `at.asitplus.regkassen.common~~core.base~~.util.CryptoUtil`
  - https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-common/src/main/java/at/asitplus/regkassen/common/util/CryptoUtil.java
  - ~~https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/base/util/CryptoUtil.java~~
- Methoden: *encryptCTR, decryptCTR*

Referenzen – Muster-Code – Verschlüsselung des Umsatzzählers über ECB Modus

- Klasse:
  - `at.asitplus.regkassen.common~~core.base~~.util.CryptoUtil`
  - https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-common/src/main/java/at/asitplus/regkassen/common/util/CryptoUtil.java
  - ~~https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/base/util/CryptoUtil.java~~
- Methoden: *encryptECB, decryptECB*

Referenzen – Muster-Code – Verschlüsselung des Umsatzzählers über CFB Modus

- Klasse:
  - `at.asitplus.regkassen.common~~core.base~~.util.CryptoUtil`
  - https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-common/src/main/java/at/asitplus/regkassen/common/util/CryptoUtil.java
  - ~~https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/base/util/CryptoUtil.java~~
- Methoden: *encryptCFB, decryptCFB*

Referenzen – RKSV

- Z 8 zur Anlage der RKSV
- Z 9 zur Anlage der RKSV
- Z 10 zur Anlage der RKSV

Ausgabewerte

- Verkürzter verschlüsselter Umsatzzähler für die Ablage im Feld „**Stand-Umsatz-Zaehler-**

Formatiert: Schriftfarbe: Grau-85 %
Formatiert: Schriftfarbe: Grau-85 %
Formatiert: Schriftfarbe: Grau-85 %
Formatiert: Schriftfarbe: Grau-85 %
Formatiert: Schriftfarbe: Grau-85 %
Formatiert: Schriftfarbe: Grau-85 %

AES256-ICM

3 Signaturerstellung

- 3.1 Erstellung der JWS-Signatur (Sicherheitseinrichtung funktionsfähig)
- 3.2 Erstellung der JWS-Signatur (Sicherheitseinrichtung ausgefallen)

3.1 Erstellung der JWS-Signatur (Sicherheitseinrichtung funktionsfähig)

Eingabewerte

- Aufbereitete Belegdaten (Ausgabewert aus Prozess 2.3)

Prozessbeschreibung

Die JWS-Signatur wird entweder manuell oder mit einer JWS-fähigen Bibliothek erstellt. Dabei sind drei Parameter für die Signaturerstellung relevant:

- **Klartextdaten (zu signierenden Daten)**: Entsprechen dem Eingabewert *„Aufbereitete Belegdaten“*
- **Algorithmus zur Hash-Berechnung/Signaturerstellung**

Als Ergebnis der JWS-Signaturerstellung erhält man die kompakte Darstellung der JWS-Signatur. In dieser Darstellung werden der Header, die Klartextdaten und der Signaturwert über das Trennzeichen `.` verknüpft: `header.payload.signature`:

- **Header**: Dabei handelt es sich um den im JWS-Standard definierten Header, der die Informationen zum verwenden Algorithmus enthält. Diese Daten werden im JSON-Format dargestellt und BASE64-URL kodiert an der ersten Stelle der kompakten Repräsentation der JWS-Signatur abgelegt.
- **Payload:** Dabei handelt es sich um die signierten Daten (entsprechend den *„Aufbereiteten Belegdaten“* (siehe Prozess 2.3). Diese Daten werden BASE64-URL kodiert und als Payload an der 2. Stelle der kompakten Repräsentation der JWS-Signatur abgelegt.
- **Signature:** Dabei handelt es sich um den Signaturwert, der das Ergebnis der angewandten Hash- und Signaturalgorithmen BASE64-URL kodiert in der kompakten Repräsentation der JWS-Signatur an der dritten Stelle abgelegt wird. Der Eingabewert für die JWS-Signaturerstellung ist `header.payload`

Anmerkungen:

Signaturwert:

Wird die Signatur manuell erstellt (ohne Verwendung einer JWS-Bibliothek) muss darauf geachtet werden, dass der Signaturwert korrekt formatiert ist. So verwendet z.B. Java ASN.1 für die Kodierung und die DER Darstellung für die Repräsentation des Signaturwerts. Der JWS-Standard[3] verlangt aber die einfache Konkatenierung der beiden Teilelemente des Signaturwertes `R` und `S`: `R|S`. Im Muster-Code ist dies in den unten angegebenen Beispielen ersichtlich.

JWS-Header:

Der JWS-Header enthält Details zu den verwendeten Algorithmen. Beim Registrierkassenalgorithmuskennzeichen `R1` wird der Signaturalgorithmus ECDSA P-256 und der Hash-Algorithmus SHA-256 verwendet. Diese Information wird als JSON-Wert im Header abgelegt und entspricht der Zeichenkette `{"alg":"ES256"}`. Die JWS-Spezifikation definiert aber nicht wie die JSON-Notation im Detail formatiert ist. Es könnte ebenso der Fall sein, dass Leerzeichen in dieser Zeichenkette enthalten sind (z.B. `{"alg" : "ES256"}`). Der JWS-Header entspricht der BASE64-URL-kodierten Zeichenkette dieser JSON-Notation und ist Bestandteil der zu signierenden Daten. Da im maschinenlesbaren Code der JWS-Header nicht abgelegt wird, und dieser Wert bei der Prüfung aus den Algorithmen im Registrierkassenalgorithmuskennzeichen rekonstruiert werden muss, muss darauf geachtet werden einen normierten Header zu verwenden. Für den typischen Fall, in dem der Header in der folgenden Notation geschrieben wird `{"alg":"ES256"}`, entspricht dies dem BASE64-URL kodierten Wert `eyJhbGciOiJFUzI1NiJ9`. Bei den evaluierten Bibliotheken wird auch dieser Wert verwendet. Sollte die Signatur manuell erstellt werden, dann muss dieser Wert als JWS-Header und damit für die Signaturerstellung verwendet werden.

Beispiel

Aufbereitete Belegdaten für die Erstellung der Signatur:

[3] A.3 Example JWS using ECDSA P-256 SHA-256, A.3.1. Encoding, JSON Web Signature (JWS)

(Für die Lesbarkeit wurden Zeilenumbrüche eingefügt).

```
_R1-AT0
_DEMO-CASH-BOX524_366588
_2015-12-17T11:23:44
_0,00
_0,00
_0,00
_26,05
_0,00
_m8LGyyY4UAA=
_20f2ed172daa09e5
_5HjRCx+XIz4=
```

Formatiert: Muster: Transparent (Hintergrund 1)

JWS Setup:
Im Falle von `R1` wird der im JSW-Standard definierte Algorithmus `ES256` verwendet. Dieses Kennzeichen identifiziert die folgenden Algorithmen:

- **Hashalgorithmus**: Der Hashalgorithmus wird anhand der im jeweiligen Registrierkassenalgorithmuskennzeichen gewählt. Im Falle von `R1` ist das der SHA-256 Algorithmus.
- **Signaturalgorithmus**: Der Signaturalgorithmus wird anhand der im jeweiligen Registrierkassenalgorithmuskennzeichen gewählt. Im Falle von `R1` ist das der ECDSA P-256.

Ergebnis der JWS-Signatur Signaturerstellung:
Kompakte Repräsentation der JWS-Signatur eines Belegs (zur besseren Darstellung sind die drei Komponenten in getrennten Zeilen dargestellt):


```
eyJhbGciOiJFUzI1NiJ9.
X1IxLUFUMF9ERU1PLUNBU0gtQk9YNTI0XzM2NjU5Nl8yMDE1LTEyLTE3VDExOjIzOjQ0XzAs
MDBfMCwwMF8zLDY0Xy0yLDYwXzEsNzlfVkZKQl80N2JlNzM3Y2IxZjZkMWYxX1p2TnhKdzZh
MUE0PQ.
J7YC28zquHfHzMpx02TqElbXOTSgXQu5JAA9Xu1Xzzu5p8eUYT-
sgmyhzRps5nYyEp5Yh8ATIa9130zmuiACHw
```


Formatiert: Muster: Transparent (Hintergrund 1)

Referenzen – Muster-Code – JWS – Nimbus JOSE + JWT

Dieses Beispiel zeigt wie die JWS-Signatur mit der Nimbus Bibliothek[4] erstellt werden kann.

- Klasse:
  - `at.asitplus.regkassen.core.modules.signature.jws.ComNimbusdsJwsModule`
  - https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/modules/signature/jws/ComNimbusdsJwsModule.java

Referenzen – Muster-Code – JWS – jose.4.j

Dieses Beispiel zeigt wie die JWS-Signatur mit der jose.4.j Bibliothek[5] erstellt werden kann.

- Klasse:
  - `at.asitplus.regkassen.core.modules.signature.jws.OrgBitbucketBcJwsModule`
  - https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/modules/signature/jws/OrgBitbucketBcJwsModule.java

Referenzen – Muster-Code – JWS – Simple JWS Module

[4] http://connect2id.com/products/nimbus-jose-jwt
[5] https://bitbucket.org/b_c/jose4j/wiki/Home

Dieses Beispiel zeigt wie die JWS-Signatur ohne die Verfügbarkeit einer JWS-fähigen Bibliothek aufbereitet werden kann.
• Klasse:
  ○ `at.asitplus.regkassen.core.modules.signature.jws.ManualJWSModule`
  ○ https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/modules/signature/jws/ManualJWSModule.java

Referenzen – Muster-Code – Signatur via PKCS11

• Klasse:
  ○ `at.asitplus.regkassen.core.modules.signature.rawsignatureprovider.PKCS11SignatureModule`
  ○ https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/modules/signature/rawsignatureprovider/PKSC11SystemSignatureModule.java

Referenzen – Muster-Code – Signatur via APDUs
Code ist noch nicht vorhanden.
Referenzen – RKSV
• Z 5 zur Anlage der RKSV • Z 6 zur Anlage der RKSV
Ausgabewerte
• Signierter Beleg in der kompakten Repräsentation der JWS-Signatur (Eingabewert für Prozess 4.1).

3.2 Erstellung der JWS-Signatur (Sicherheitseinrichtung ausgefallen)

Eingabewerte

- Aufbereitete Belegdaten (Ausgabewert aus Prozess 2.3)

Prozessbeschreibung

Beim Ausfall der Sicherheitseinrichtung kann die JWS-Signatur nicht erstellt werden. Um dennoch in der Lage zu sein, Belege zu erstellen, die der RKSV entsprechen und im DEP abgelegt werden können, wird für diese Fälle eine zur kompakten Repräsentation der JWS-Signatur kompatible Darstellung gewählt: `header.payload.sicherheitseinrichtung-ausgefallen`. Die ersten beiden Elemente `header` und `payload` entsprechen dabei der kompakten JWS-Repräsentation.

- **Header**: siehe Prozess 3.1
- **Payload:** siehe Prozess 3.1
- **sicherheitseinrichtung-ausgefallen:** Statt dem Signaturwert, der in diesem Fall nicht berechnet werden kann, wird die Zeichenkette `Sicherheitseinrichtung ausgefallen` BASE64-URL kodiert und als drittes Element anstelle des Signaturwertes in der kompakten Darstellung der JWS-Signatur verwendet. In der BASE64-URL Kodierung entspricht dies dem Wert `U2ljaGVyaGVpdHNlaW5yaWNodHVuZyBhdXNnZWZhbGxlbg.`[6]

Beispiel

Aufbereitete Belegdaten für die Erstellung der Signatur:
(Für die Lesbarkeit wurden Zeilenumbrüche eingefügt)

```
_R1-AT0
_DEMO-CASH-BOX524
_366588
_2015-12-17T11:23:44
_0,00
_0,00
_0,00
_26,05
_0,00
_m8LGyyY4UAA=
_20f2ed172daa09e5
_5HjRCx+XIz4=
```

Ergebnis der JWS-Signatur Signaturerstellung:
Das dritte Element entspricht der BASE64-URL Kodierung der Zeichenkette `Sicherheitseinrichtung ausgefallen` (Zur besseren Darstellung sind die drei Komponenten in getrennten Zeilen dargestellt):


```
eyJhbGciOiJFUzI1NiJ9.
X1IxLUFUMF9ERU1PLUNBU0gtQk9YNTI0XzM2NjU5N18yMDE1LTEyLTE3VDExOjIzOjQ0XzE0
LDc0XzAsMDBfMCwwMF81MSw5NV8wLDAwX1ZGSkJfNDdiZTczN2NiMWY2ZDFmMV9wL2pmWEd5
UlFIND0.
U2ljaGVyaGVpdHNlaW5yaWNodHVuZyBhdXNnZWZhbGxlbg
```


Referenzen – Muster-Code – Behandlung der ausgefallenen Sicherheitseinrichtung

Es wird dazu auf Prozess 3.1 verwiesen, der Beispiele im Zusammenhang mit den JWS-Modulen des Muster-Codes zeigt.

Referenzen – RKSV

- Z 5 der Anlage zur RKSV
- Z 6 der Anlage zur RKSV

Ausgabewerte

- Signierter Beleg in der kompakten Repräsentation der JWS-Signatur (Eingabewert für

Formatiert: Muster: Transparent (Hintergrund 1)

Formatiert: Muster: Transparent (Hintergrund 1)

[6] Achtung: In der Darstellung des maschinenlesbaren Codes wo statt der BASE64-URL-Kodierung die BASE64-Kodierung verwendet entspricht der Wert
U2ljaGVyaGVpdHNlaW5yaWNodHVuZyBhdXNnZWZhbGxlbg==

Prozess 4.1).

4 Aufbereitung der maschinenlesbaren Codes

Das Ergebnis der Signaturerstellung wird nun je nach Verwendung unterschiedlich aufbereitet. In dieser Prozessbeschreibung wird davon ausgegangen, dass zuerst für alle weiteren Fälle eine Basisvariante erstellt wird (der maschinenlesbare Code), der dann die Basis für die Erstellung der QR- OCR- oder Link-Repräsentation verwendet wird.

Daraus ergeben sich folgende Prozessbeschreibungen:

- 4.1 Aufbereitung des maschinenlesbaren Codes (Basis-Code)
- 4.2 Aufbereitung QR-Code
- 4.3 Aufbereitung OCR-Code
- 4.4 Aufbereitung Link
- 4.5 Aufbereitung für DEP

4.1 Aufbereitung des maschinenlesbaren Codes (Basis-Code)

Eingabewerte

- Signierter Beleg in der kompakten Repräsentation der JWS-Signatur (Ausgabewert von Prozess 3.1 oder 3.2).

Prozessbeschreibung

Es wird für die folgenden Prozesse davon ausgegangen, dass das Ergebnis der Signaturerstellung – die kompakte Repräsentation der JWS-Signatur – für die weiteren Prozesse verwendet wird.

Die kompakte Repräsentation der JWS-Signatur wird wie folgt für die Aufbereitung des maschinenlesbaren Codes verarbeitet:

- **Extraktion des Signaturwerts:** Aus der kompakten Repräsentation der JWS-Signatur wird das dritte Element – der Signaturwert – extrahiert. Der extrahierte Wert entspricht dem BASE64-URL-kodierten Signaturwert.
- **Transformation des Signaturwerts von BASE64-URL Kodierung zu BASE64 Kodierung:** Der BASE64-URL kodierte Signaturwert wird dekodiert und anschließend BASE64-kodiert.
- **Zusammenfügen der zu signierenden Daten und des Signaturwerts:** Die zu signierenden Daten – das Ergebnis von Prozess 2.3 – und der BASE64-kodierte Signaturwert werden in dieser Reihenfolge mit dem Trennzeichen _ zusammengefügt. Als Ergebnis erhält man den maschinenlesbaren Code.

Beispiele

Der folgende signierte Beleg ist in der kompakten JWS-Repräsentation gegeben:


```
eyJhbGciOiJFUzI1NiJ9.
X1IxLUFUMF9ERU1PLUNBU0gtQk9YNTI0XzM2NjU5Nl8yMDE1LTEyLTE3VDExOjIzOjQ0XzAs
MDBfMCwwMF8zLDY0Xy0yLDYwXzEsNzlfVkZKQl80N2JlNzM3Y2IxZjZkMWYxX1p2TnhKdzZh
MUE0PQ.
J7YC28zquHfHzMpx02TqElbXOTSgXQu5JAA9Xu1Xzzu5p8eUYT-
sgmyhzRps5nYyEp5Yh8ATIa9130zmuiACHw
```


Der BASE64-URL-kodierte Signaturwert (3. Element):


```
J7YC28zquHfHzMpx02TqElbXOTSgXQu5JAA9Xu1Xzzu5p8eUYT-
sgmyhzRps5nYyEp5Yh8ATIa9130zmuiACHw
```


Dieser Wert wird dekodiert und BASE64-kodiert. Das Ergebnis entspricht der folgenden Zeichenkette:


```
J7YC28zquHfHzMpx02TqElbXOTSgXQu5JAA9Xu1Xzzu5p8eUYT+sgmyhzRps5nYyEp5Yh8AT
Ia9130zmuiACHw==
```


Es werden nun die signierten Daten dekodiert. In diesem Beispiel entsprechen die BASE64-URL kodierten Daten der folgenden Zeichenkette (2. Element):


```
X1IxLUFUMF9ERU1PLUNBU0gtQk9YNTI0XzM2NjU5Nl8yMDE1LTEyLTE3VDExOjIzOjQ0XzAs
MDBfMCwwMF8zLDY0Xy0yLDYwXzEsNzlfVkZKQl80N2JlNzM3Y2IxZjZkMWYxX1p2TnhKdzZh
MUE0PQ
```


Der dekodierte Wert entspricht der folgenden Zeichenkette:
(Für die Lesbarkeit wurden Zeilenumbrüche eingefügt)

```
_R1-AT0
_DEMO-CASH-BOX524
_366596
_2015-12-17T11:23:44
_0,00
_0,00
_3,64
_-2,60
_1,79
```

Formatiert: Muster: Transparent (Hintergrund 1)

Formatiert: Muster: Transparent (Hintergrund 1)

Formatiert: Muster: Transparent (Hintergrund 1)

Formatiert: Muster: Transparent (Hintergrund 1)

Formatiert: Muster: Transparent (Hintergrund 1)

```
_VFJB
_47be737cb1f6d1f1
_ZvNxJw6a1A4=
```

Die signierten Daten werden nun mit dem Signaturwert über das Trennzeichen _ zusammengefügt. Dies ergibt den maschinenlesbaren Code der in weitere Folge für die QR/OCR oder Link-Repräsentation verwendet wird.
(Für die Lesbarkeit wurden Zeilenumbrüche eingefügt)

```
_R1-AT0
_DEMO-CASH-BOX524
_366596
_2015-12-17T11:23:44
_0,00
_0,00
_3,64
_-2,60
_1,79
_VFJB
_47be737cb1f6d1f1
_ZvNxJw6a1A4=
_J7YC28zquHfHzMpx02TqElbXOTSgXQu5JAA9Xu1Xzzu5p8eUYT+sgmyhzRps5nYyEp5Yh8A
TIa9130zmuiACHw==
```

Formatiert: Muster: Transparent (Hintergrund 1)

Referenzen – Muster-Code

Es wird dazu auf Prozess 3.1 verwiesen, der Beispiele im Zusammenhang mit den JWS-Modulen des Muster-Codes zeigt.

Referenzen – RKSV

- Z 12 zur Anlage der RKSV

Ausgabewerte

- Maschinenlesbarer Code als Basis für Repräsentation über QR/OCR-Code und Link

4.2 Aufbereitung QR-Code

Eingabewerte

- Maschinenlesbarer Code als Basis für Repräsentation über QR/OCR-Code und Link (Ausgabewert von Prozess 4.1).

Prozessbeschreibung

Der QR-Code wird anhand des Standards JIS X 0510:2004 erstellt. Der Inhalt des QR-Codes ist der in Prozess 4.1 aufbereitete maschinenlesbare Code. Es wird keine Vorgabe zur Fehlerkorrektur gemacht, allerdings muss berücksichtigt werden, dass das Lesen des Codes unter realen Bedingungen noch möglich ist. Im Muster-Code wird der Fehlerkorrektur Level `M` verwendet.

Beispiele

Gegeben ist der maschinenlesbare Code (aufbereitet anhand von Prozess 4.1):
(Für die Lesbarkeit wurden Zeilenumbrüche eingefügt)

```
_R1-AT0
_DEMO-CASH-BOX524
_366596
_2015-12-17T11:23:44
_0,00
_0,00
_3,64
_-2,60
_1,79
_VFJB
_47be737cb1f6d1f1
_ZvNxJw6a1A4=
_J7YC28zquHfHzMpx02TqElbXOTSgXQu5JAA9Xu1Xzzu5p8eUYT+sgmyhzRps5nYyEp5Yh8A
TIa9130zmuiACHw==
```

Es wird nun ein QR-Code für diese Zeichenkette erstellt:

Referenzen – Muster-Code

- Klasse:
  - `at.asitplus.regkassen.core.modules.print.SimplePDFPrinterModule`
  - https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/modules/print/SimplePDFPrinterModule.java
- Methoden: *createQRCode* und *printReceipt*

Referenzen – RKSV

- Z 12 zur Anlage der RKSV

Ausgabewerte

- QR-Code der auf Beleg gedruckt wird

Formatiert: Muster: Transparent (Hintergrund 1)

4.3 Aufbereitung OCR-Code

Eingabewerte

- Maschinenlesbarer Code als Basis für Repräsentation über QR/OCR-Code und Link (Ausgabewert von Prozess 4.1).

Prozessbeschreibung

Der OCR-Code wird durch eine geringe Abwandlung des maschinenlesbaren Codes erstellt. Die Transformation betrifft die Umkodierung der BASE64-kodierten Felder auf BASE32-kodierte Felder, um die maschinelle Lesbarkeit des OCR-Codes zu verbessern. Dabei werden folgende BASE64-kodierten Felder des maschinenlesbaren Codes dekodiert und anschließend BASE32-kodiert:

- **Stand-Umsatz-Zaehler-AES256-ICM**
- **Sig-Voriger-Beleg**
- **Signaturwert**

Der umkodierte maschinenlesbare Code wird direkt im OCR-A Font auf den Beleg gedruckt. Die Font-Größe ist nicht definiert, es muss aber berücksichtigt werden, dass das Einlesen des Codes von einem Beleg noch möglich ist.

Beispiele

Gegeben ist der maschinenlesbare Code eines Belegs (aufbereitetet anhand von Prozess 4.1):
(Für die Lesbarkeit wurden Zeilenumbrüche eingefügt)

```
_R1-AT0
_DEMO-CASH-BOX524
_366610
_2015-12-17T11:23:44
_0,00
_0,00
_0,00
_0,00
_0,00
_DngfhpghKu8=
_47be737cb1f6d1f1_hArhdCTSjbM=
_Ubz6l2+TFt80EWAf6CTr/Xocpgn9UhOhSvlGMqQsML1LxieEE7bWqB5Y6HAwaC0NSA50swr
GHxKs/UOGPS1SJA==
```

Die Felder **Stand-Umsatz-Zaehler-AES256-ICM**, **Sig-Voriger-Beleg** und **Signaturwert** werden extrahiert:

- **Stand-Umsatz-Zaehler-AES256-ICM:**
  `DngfhpghKu8=`
- **Sig-Voriger-Beleg:**
  `hArhdCTSjbM=`
- **Signaturwert:**
  `Ubz6l2+TFt80EWAf6CTr/Xocpgn9UhOhSvlGMqQsML1LxieEE7bWqB5Y6HAwaC0NSA50swrGHxKs/UOGPS1SJA==`

Diese BASE64-kodierten Felder werden dekodiert und BASE32-kodiert: Dies ergibt:

- **Stand-Umsatz-Zaehler-AES256-ICM:**
  `BZ4B7BUYEEVO6===`
- **Sig-Voriger-Beleg:**
  `QQFOC5BE2KG3G===`
- **Signaturwert:**
  `KG6PVF3PSMLN6NARMAP6QJHL7V5BZJQJ7VJBHIKK7FDDFJBMGC6UXRRHQQJ3NVVIDZMOQ4BQNAWQ2SAOOSZQVRQ7CKWP2Q4GHUWVEJA=`

Die Felder werden im maschinenlesbaren Code eingetragen. Dieser modifizierte Code entspricht dem OCR-Code der auf den Beleg aufgedruckt wird.
(Für die Lesbarkeit wurden Zeilenumbrüche eingefügt)

Formatiert: Muster: Transparent (Hintergrund 1)

Formatiert: Schriftart: Courier, Englisch (USA)

Formatiert: Schriftart: Courier, Hervorheben

```
_R1-AT0
_DEMO-CASH-BOX524
_366610
_2015-12-17T11:23:44
_0,00
_0,00
_0,00
_0,00
_0,00
_BZ4B7BUYEEVO6===
_47be737cb1f6d1f1
_QQFOC5BE2KG3G===
_KG6PVF3PSMLN6NARMAP6QJHL7V5BZJQJ7VJBHIKK7FDDFJBMGC6UXRRHQQJ3NVVIDZMOQ4B
QNAWQ2SAOOSZQVRQ7CKWP2Q4GHUWVEJA=
```

Formatiert: Muster: Transparent (Hintergrund 1)

Referenzen – Muster-Code

- Klasse:
  - at.asitplus.regkassen.common~~core.base~~.util.CashBoxUtils
  - https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-common/src/main/java/at/asitplus/regkassen/common/util/CashBoxUtils.java
  - ~~https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/base/util/CashBoxUtils.java~~
- Methode: *getOCRCodeRepresentationFromJWSCompactRepresentation*

Formatiert: Schriftfarbe: Grau-85 %

Formatiert: Schriftfarbe: Grau-85 %

Referenzen – RKSV

- Z 14 zur Anlage der RKSV

Ausgabewerte

- OCR-Code der auf Beleg gedruckt wird

4.4 Aufbereitung Link

Eingabewerte

- Maschinenlesbarer Code als Basis für Repräsentation über QR/OCR-Code und Link (Ausgabewert von Prozess 4.1).

Prozessbeschreibung

Ist es nicht möglich den maschinenlesbaren Code als QR-Code oder als OCR-Kette auf den Beleg zu drucken, kann eine aufgedruckte URL verwendet werden, die den maschinenlesbaren Code über das HTTP-Protokoll auf einem externen Server zur Verfügung stellt. Die URL kann dabei aus beliebigen Komponenten bestehen. Wesentliche Anforderung ist, dass ein Teil der URL (Teil des Pfades, als Parameter, als Fragment etc.) einem Teil des SHA256-Hash-Werts entspricht der über den maschinenlesbaren Code gerechnet wird. Die Notwendigkeit für die Hash-Wert Bildung gibt sich daraus, dass der LINK an den Beleg gebunden sein muss, um sicherzustellen, dass der Inhalt des Links nach der Belegerstellung nicht geändert wird.

Dabei wird wie folgt vorgegangen:

- **Hashwert-Berechnung**: Es wird der SHA-256 Algorithmus auf die UTF-8 Kodierung des maschinenlesbaren Codes angewendet. Als Ergebnis erhält man den SHA-256 Hash-Wert mit einer Länge von 32 Bytes (`HASH-VALUE-32`).
- **Extraktion von 8 Bytes**: Vom Ergebnis (`HASH-VALUE-32`) werden die ersten 8 Bytes extrahiert (Bytes 0-15). Das Ergebnis wird in weiterer Folge als `HASH-VALUE-8` bezeichnet.
- **Kodierung des Ergebnisses**: Die 8 extrahierten Bytes (`HASH-VALUE-8`) werden BASE64-URL kodiert. Ergebnis: `HASH-VALUE-8-BASE64-URL`
- **Konstruktion des Links**: Das Ergebnis (`HASH-VALUE-8-BASE64-URL`) wird für die Konstruktion des "Links" verwendet. Es spielt prinzipiell keine Rolle an welcher Stelle und mit welcher Methode der Wert in die URL integriert wird. Die folgenden Beispiele zeigen prinzipielle Möglichkeiten:
  - **Teil des Pfades**: In diesem Fall ist der Wert Teil des Pfades: z.B. `http(s)://test.example.at/dir1/dir2/.../dirN/HASH-VALUE-8-BASE64-URL`.
    Der Wert muss nicht der letzte Teil der URL sein: z.B.: `http(s)://test.example.at/dir1/ HASH-VALUE-8-BASE64-URL/.../dirN/`.
    Wobei es aus Platzgründen sinnvoll ist keine oder nur wenige Verzeichnisse zu verwenden. Im Idealfall wäre eine URL die ohne Verzeichnisse auskommt zu verwenden: `http(s)://test.example.at/HASH-VALUE-8-BASE64-URL`.
  - **Als Parameter**: `http(s)://test.example.at/dir1 /dir2/.../?value=HASH-VALUE-8-BASE64-URL`.
    Auch hier gilt, dass die Länge der URL möglichst klein gehalten werden sollte.
    Der Name des Parameters spielt keine Rolle (hier wurde „value“ verwendet).
  - **Als Fragment**: z.B. `http(s)://test.example.at/dir1/dir2/...#HASH-VALUE-8-BASE64-URL`

Der maschinenlesbare Code muss nun beim Aufrufen des "Links" mit dem HTTP-Befehl "GET" abgerufen werden können. Dabei sind folgende Punkte relevant:

- **HTTP-Header**: Es muss der Content-Type `application/json` gesetzt sein.
  `Content-Type: application/json`
- **Ergebnis**: Es muss das in JSON aufbereitete Ergebnis zurückgegeben werden, wobei der Platzhalter `MACHINE-READABLE-CODE` mit dem maschinenlesbaren Code (äquivalent zur Darstellung für den QR-Code) ersetzt werden muss.

```
{
      "code": "MACHINE-READABLE-CODE"
}
```

Beispiel – Aufbereitung der „Link-Repräsentation“ eines maschinenlesbaren Codes

Das spezifizierte Verfahren wird anhand eines Beispiels demonstriert.

Maschinenlesbarer Code wie folgt:
(Für die Lesbarkeit wurden Zeilenumbrüche eingefügt)

```
_R1-AT0
_DEMO-CASH-BOX703
_496410
_2015-12-17T11:22:26
_61,33
_15,42
_38,26
_0,00
_42,81
_udSL0zTzFaA=
_b869153bc32a1c9
_TZgOKfzheUs=
_d7VoZ/nyr/8Mt2NVBZ0L4ivQwdlE3CmCFmz10bA5NXhGQ1QkunsDTqKvOSy//3nO8WT0+yp
QqrDXMvyOGOXl9w==
```

Formatiert: Muster: Transparent (Hintergrund 1)

Formatiert: Schriftart: Arial, Englisch (USA)

Beispiel URLs:
Für das Aufbereiten des Links wird von den folgenden Beispielen für Basis-URLs ausgegangen (http/https mit/ohne Verzeichnis).

- http://cashbox.example.at/HASH-VALUE-8-BASE64-URL
- https://cashbox.example.at/HASH-VALUE-8-BASE64-URL/machineReadableCode
- https://cashbox.example.at/receipts/HASH-VALUE-8-BASE64-URL
- https://cashbox.example.at/receipts/machinereadablecode#HASH-VALUE-8-BASE64-URL
- https://cashbox.example.at/receipts?value=HASH-VALUE-8-BASE64-URL

Berechnung des Hash-Werts:
Von den 32 Bytes des Ergebnisses der SHA-256 Hashwert-Berechnung über die UTF-8 Kodierung des maschinenlesbaren Codes werden die ersten 8 Bytes extrahiert und BASE64-URL-kodiert. Das Ergebnis für das gegebene Beispiel entspricht dem folgenden Wert: `ikkiOyDQ2Bo`

Erstellen des "Links":
Das Ergebnis wird für die beiden Beispiel-URLs wie folgt eingefügt

- http://cashbox.example.at/ikkiOyDQ2Bo
- https://cashbox.example.at/ikkiOyDQ2Bo/machineReadableCode
- https://cashbox.example.at/receipts/ikkiOyDQ2Bo
- https://cashbox.example.at/receipts/machinereadablecode#ikkiOyDQ2Bo
- https://cashbox.example.at/receipts?value=ikkiOyDQ2Bo

Abrufen des maschinenlesbaren Codes über den "Link":
Durch das Ausführen des HTTP GET Befehls auf einen der aufbereiteten Links wird für das im Bespiel verwendeten maschinenlesbaren Code die folgende Antwort gegeben (es wird nur der einzig relevante Header gezeigt, andere typisch verwendete HTTP Header werden aus Gründen der Übersichtlichkeit nicht gezeigt).

```
HTTP/1.1 200 OK
Content-Type: application/json

{
      "code" : "_R1-AT0_DEMO-CASH-BOX703_496410
                _2015-12-17T11:22:26
                _61,33_15,42_38,26_0,00_42,81
```

Formatiert: Muster: Transparent (Hintergrund 1)

```
                    _udSL0zTzFaA=_b869153bc32a1c9_TZgOKfzheUs=
                    _d7VoZ/nyr/8Mt2NVBZ0L4ivQwdlE3CmCFmz10bA5NXhGQ1Qkuns
                    DTqKvOSy//3nO8WT0+ypQqrDXMvyOGOXl9w=="
}
```


Referenzen – Muster-Code

Code noch nicht vorhanden.

Referenzen – RKSV

- § 11 ~~91~~ Abs. 2 Z 1 RKSV

Ausgabewerte

- Link der auf maschinenlesbaren Code im angegeben Format verweist

Formatiert: Englisch (USA)

Formatiert: Englisch (USA)

Formatiert: Englisch (USA)

Formatiert: Englisch (USA)

4.5 Aufbereitung für DEP

Eingabewerte
• Signierter Beleg in der kompakten Repräsentation der JWS-Signatur (Ausgabewert von Prozess 3.1 oder 3.2).
Prozessbeschreibung
Die kompakte Repräsentation der JWS-Signatur wird direkt an das DEP übergeben.
Referenzen – RKSV
• Z 14 zur Anlage der RKSV
Ausgabewerte
• Übergabe an das DEP

5 Export des Datenerfassungsprotokolls

Eingabewerte

- Teile des Datenerfassungsprotokolls[7] die die Historie der signierten Belege anhand der in Prozess 4.5 definierten Aufbereitung enthalten.

Prozessbeschreibung

JSON-Notation:
Beim DEP-Exportformat handelt es sich um ein Objekt in JSON-Notation das folgende Felder enthält:

- `Belege-Gruppe`
  - Liste (JSON Array) bestehend aus einem oder mehreren JSON-Objekten die wie folgt aufgebaut sind:
    - `Signaturzertifikat`
      - Der Wert entspricht dabei dem BASE64-kodierten Wert des im DER-Format kodierten Signaturzertifikats.
    - `Zertifizierungsstellen`
      - Liste (JSON Array) bestehend aus den Zertifizierungsstellen die für die Ausstellung des Signaturzertifikats verwendet wurden. Ein Eintrag dieser Liste entspricht dabei dem BASE64-kodierten Wert des im DER-Format kodierten Zertifikats der jeweiligen Zertifizierungsstelle.
    - `Belege-Kompakt`
      - Liste (JSON Array) bestehend aus den Belegen die mit dem angegebenen `Signaturzertifikat` signiert wurden. Die Belege werden dabei in der kompakten Repräsentation der JWS-Signatur abgelegt (entspricht Aufbereitung nach Prozess 4.5).

Dies entspricht der folgenden Darstellung, wobei Platzhalter für die Zertifizierungsstellen, das Signaturzertifikat (`BASE64-DER-Zertifikat`) und die Belege (`JWS-REP-Beleg`) verwendet wurden. `...` verdeutlicht, dass ein oder mehrere Elemente in der Liste vorkommen können:

```
{
  "Belege-Gruppe": [
    {
      "Signaturzertifikat": "BASE64-DER-Zertifikat",
      "Zertifizierungsstellen": [
        "BASE64-DER-Zertifikat",
        "...",
        "..."
      ],
      "Belege-kompakt": [
        "JWS-REP-BELEG 1"
        "...",
        "..."
      ]
    },
    {...},
    {...}
}
```

Formatiert: Muster: Transparent (Hintergrund 1)

Schritte und Anmerkungen:
Für die Erstellung des Exports des Datenerfassungsprotokolls im Sinne der RKSV werden folgende Schritte durchgeführt:

- **Aufbereiten der Belege**: Die JWS-Repräsentationen (anhand der Aufbereitung von

[7] Es werden hier nur die Teile beachtet die im Sinne der RKSV notwendig sind. Andere Elemente des DEPs werden hier nicht berücksichtigt.

Prozess 4.5) der Belege des DEPs der Kasse werden chronologisch (im Sinne der Reihenfolge der Verkettung) sortiert.

- **Gruppierung der Belege nach Signaturzertifikaten (wenn vorhanden)**: Die chronologisch sortierten Belege werden anhand der verwendeten Signaturzertifikate gruppiert. Die chronologische Reihenfolge (im Sinne der Verkettung) der Belege darf nicht geändert werden. Dies ist durch ein Beispiel erläutert: Belege 1-10 mit Signaturzertifikat 1 erstellt, Belege 11-20 mit Signaturzertifikat 2 erstellt, Belege 21-30 wieder mit Signaturzertifikat 1 erstellt. Um die Reihenfolge einzuhalten, entstehen in diesem Beispiel drei Beleggruppen: Die erste Beleggruppe wird für das Signaturzertifikat 1 erstellt und enthält die Belege 1-10, die zweite Beleggruppe wird für das Signaturzertifikat 2 erstellt und erhält die Belege 11-20 und die dritte Beleggruppe wird für das Signaturzertifikat 1~~3~~ erstellt und erhält die Belege 21-30.
- **Gruppierung der Belege, wenn kein Signaturzertifikat vorhanden**: In diesem Fall wird eine Beleggruppe erstellt die kein Signaturzertifikat enthält (das Feld `Signaturzertifikat` ist leer, das Feld `Zertifizierungsstellen` entspricht einer leeren Liste) und alle Belege – unabhängig vom Signaturzertifikat mit dem sie signiert wurden – werden chronologisch sortiert im Feld `Belege-kompakt` als Liste abgelegt.

Geschlossene Gesamtsysteme:
Bei geschlossenen Gesamtsystemen können Zertifikate für die Repräsentation des öffentlichen Schlüssels verwendet werden. In diesem Fall gilt das gleiche wie für offene Systeme. Allerdings besteht für geschlossene Gesamtsysteme auch die Möglichkeit, dass nur öffentliche Schlüssel verwendet werden ohne Zertifikate für die Repräsentation zu verwenden. Um in diesem Fall korrekte Exports zu erstellen gibt es zwei Möglichkeiten: (1) Es kann ein selbstsigniertes Zertifikat erstellt werden das den jeweiligen Schlüssel erhält. Dieses kann dann im Feld Signaturzertifikat abgelegt werden. Das Feld Zertifizierungsstellen bleibt leer. (2) Die Felder (wie auch bei offenen Systemen) können leer gelassen werden.

Beispiel – Export des DEPs wenn Signaturzertifikate vorhanden

Im folgenden Beispiel wird der Export eines DEPs mit einem Signaturzertifikatwechsel dargestellt (es sind also insgesamt zwei Signaturzertifikate in Verwendung). Aus Gründen der Darstellbarkeit wurde statt der JWS-Repräsentation der Belege nur der Platzhalter `JWS-REP-BELEG` verwendet.
Die Signaturzertifikate und der Zertifizierungsstellen entsprechend in diesem Beispiel selbstsignierten Demo-Zertifikaten. Bei einem offenen System werden hier die tatsächlichen Zertifikate des jeweiligen ZDAs verwendet.


```
{
  "Belege-Gruppe": [
    {
      "Signaturzertifikat":
"MIIBSTCB8KADAgECAggg8u0XLaoJ5TAKBggqhkjOPQQDAjAWMRQwEgYDVQQDDAtSZWdLYXN
zYSBDQTAeFw0xNTEyMTcxMDIzMzNaFw0xNTEyMTgxMDIzNDNaMB4xHDAaBgNVBAMME1NpZ25
pbmcgY2VydGlmaWNhdGUwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAQhRIDwsPXiuD0hyFO
T6R31/ho0YYs1jUXKUMUBTVv816S1sLuB6Cb+rh7y4wRv+jqLnIffvi4jjALZB88YZDu9oyA
wHjAMBgNVHRMBAf8EAjAAMA4GA1UdDwEB/wQEAwIHgDAKBggqhkjOPQQDAgNIADBFAiBOQsw
8uN4RL6R0wXnclmHVFuanU60SOHFMI6ja/r9ZLwIhAMelF0Z5K+Y1pfuf0R990h5ivzhXO2s
PkWHoTaQVQFXJ",
      "Zertifizierungsstellen": [

"MIIBQTCB6aADAgECAghmr9KTm3Bo6jAKBggqhkjOPQQDAjAXMRUwEwYDVQQDDAxSZWdLYXN
zYSBaREEwHhcNMTUxMjE3MTAyMzMyWhcNMTUxMjE4MTAyMzQyWjAWMRQwEgYDVQQDDAtSZWd
LYXNzYSBDQTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABB27V8Po7R92RTboOLjUC6dufRy
lnTU7HT8932uHnBtM3qcp9umBxmVWspPOFr90i8IyzU0eoaqwTHzNeATxpDGjIDAeMAwGA1U
dEwEB/wQCMAAwDgYDVR0PAQH/BAQDAgeAMAoGCCqGSM49BAMCA0cAMEQCIGUqX2+86EMOgNP
DSbv2JCpflAWZrMu5ZFpbckW3pYW4AiATnXn40x4QyVXSktQJJmPaLAsbF2kvOWgJ1ALUcuj
wnw==~~"\u003d\u003d"~~
```


Formatiert: Muster: Transparent (Hintergrund 1)

```
      ],
      "Belege-kompakt": [
        "JWS-REP-BELEG",
        "JWS-REP-BELEG",
        "JWS-REP-BELEG"
      ]
    },
    {
      "Signaturzertifikat":
"MIIBSDCB8KADAgECAghHvnN8sfbR8TAKBggqhkjOPQQDAjAWMRQwEgYDVQQDDAtSZWdLYXN
zYSBDQTAeFw0xNTEyMTcxMDIzMzRaFw0xNTEyMTgxMDIzNDRaMB4xHDAaBgNVBAMME1NpZ25
pbmcgY2VydGlmaWNhdGUwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAASPaj9uIdFY3cJIEPO
po+B23PHK+0BWMxWK7Yl/mI7Suer8TP8vK2V3YTjdH1eANvIhh2SMI9xiyq0E47QBeDWpoyA
wHjAMBgNVHRMBAf8EAjAAMA4GA1UdDwEB/wQEAwIHgDAKBggqhkjOPQQDAgNHADBEAiAGl2j
EXB0vwSOZT8/DnHd/MEF6yTiDsEPcNDD6hHQGcAIgPbasx0HB/R7hNUwUm6N7LRk7rVptSK/
s5lP7exkwOPU=",\u003d",
      "Zertifizierungsstellen": [

"MIIBQTCB6aADAgECAgjzviUdJ70FiDAKBggqhkjOPQQDAjAXMRUwEwYDVQQDDAxSZWdLYXN
zYSBaREEwHhcNMTUxMjE3MTAyMzM0WhcNMTUxMjE4MTAyMzQ0WjAWMRQwEgYDVQQDDAtSZWd
LYXNzYSBDQTBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABJNRHDj1gVebV03/YPqKv1wTOmM
YdmuLfjh6hw62YL7qdC2gqrDT/xtUIUqVpmUlwam3SSosM6nyaI5+Yvf2c3CjIDAeMAwGA1U
dEwEB/wQCMAAwDgYDVR0PAQH/BAQDAgeAMAoGCCqGSM49BAMCA0cAMEQCIDniU86fuFSqd1K
ok+qal/k71neMOAo97FgSM1Z/O5ubAiAbZlNT55YWCOJiK3wxD+oluaxtxpO3Rq7EGvusJzn
dTg==""\u003d\u003d"
      ],
      "Belege-kompakt": [
        "JWS-REP-BELEG",
        "JWS-REP-BELEG",
        "JWS-REP-BELEG",
        "JWS-REP-BELEG"
      ]
    }
  ]
}
```


Beispiel – Export des DEPs wenn keine Signaturzertifikate vorhanden sind

Im folgenden Beispiel wird davon ausgegangen, dass kein Signaturzertifikat vorhanden ist. Alle Belege – auch wenn Sie mit unterschiedlichen Zertifikaten signiert wurden – werden in einer Gruppe chronologisch abgelegt.


```
{
  "Belege-Gruppe": [
    {
      "Signaturzertifikat": "",
      "Zertifizierungsstellen": [],
      "Belege-kompakt": [
        "JWS-REP-BELEG",
        "JWS-REP-BELEG",
        "JWS-REP-BELEG",
        "JWS-REP-BELEG",
        "JWS-REP-BELEG",
        "JWS-REP-BELEG",
        "JWS-REP-BELEG",
        "JWS-REP-BELEG"
      ]
    }
  ]
}
```


Formatiert: Muster: Transparent (Hintergrund 1)

Referenzen – Muster-Code

Kodierung des Exportformats:
• Paket:
  ○ `at.asitplus.regkassen.core.modules.DEP`
  ○ https://github.com/a-sit-plus/at-registrierkassen-mustercode/tree/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/modules/DEP
 	
Methode für Export:
• Klasse:
  ○ `at.asitplus.regkassen.core.modules.DEP.SimpleMemoryDEPModule`
  ○ https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/modules/DEP/SimpleMemoryDEPModule.java
• Methode: *exportDEP*

Referenzen – RKSV
• Z 3 zur Anlage der RKSV
Ausgabewerte
• Export des DEPs laut RSKV

6 Berechnung der Prüfsumme des AES-Schlüssels

Um Fehleingaben bei der Erfassung von AES-Schlüsseln bei der Registrierung von Registrierkassen über FinanzOnline vermeiden zu helfen, führt das BMF ein Prüfwertverfahren für die Erfassung der AES-Schlüssel über FinanzOnline ein, das ab Ende August 2016 zur Verfügung stehen wird. Das Prüfwertverfahren ist optional und besteht aus einem vierstelligen Prüfwert, der nach vorgegebenen Regeln (Berechnungsalgorithmus) aus dem AES-Schlüssel ermittelt werden kann. Wird der Prüfwert nach dem selben Berechnungsalgorithmus auch von der Registrierkasse ermittelt und vom Unternehmer zusätzlich zum AES-Schlüssel über FinanzOnline erfasst, stellt FinanzOnline durch eine Vergleichsrechnung sicher, dass der AES-Schlüssel fehlerfrei über FinanzOnline erfasst wurde.

Die Verwendung des SHA256-Hash-Wertes begründet sich in der Tatsache, dass die Kassensoftware die dafür benötigten Softwarebibliotheken bereits im Einsatz hat. Auch die Extraktion von einer gegebenen Anzahl von Bytes aus dem berechneten Hash-Wert muss bereits im Rahmen der RKSV-konformen Umsetzung vorhanden sein. Die Aufwände für die Implementierung sollen damit minimal gehalten werden.

Eingabewerte

- **base64AESKey**: BASE64-kodierter AES Schlüssel, mit dem die Kasse initialisiert wurde und der im FinanzOnline gemeldet werden soll.
- **N:** Die Anzahl der Bytes, die vom Hash-Wert extrahiert werden. Es wird **N=3** festgelegt.

Prozessbeschreibung

Berechnung der Prüfsumme:

- **Hashberechnung:** SHA256-Hash-Wert-Berechnung von **base64AESKey ➔ sha256hash** (Byte Array der Länge 32)
- Extraktion der ersten **N** Bytes aus **sha256hash ➔ sha256hashNbytes** (Byte Array der Länge **N**)
- BASE64-Kodierung von **sha256hashNbytes ➔ base64sha256hashNbytes** (keine BASE64-URL Kodierung!)
- Entfernen aller „=“ Zeichen aus **base64sha256hashNbytes ➔ valSumCalc**

Prüfsumme:

- **valSumCalc:** Prüfwert der vom Unternehmer im FinanzOnline eingegeben werden kann. FinanzOnline verwendet den gleichen Algorithmus für die Berechnung des Prüfwerts und informiert den Unternehmer, wenn der berechnete und der eingegebene Wert nicht identisch sind.

Beispiel für die Berechnung der Prüfsumme

Eingabewerte:

- **base64AESKey**: `cWhay3H4asTvRzXzXGZQ3KyBEu9BZaIxl8J+L4Bhr5A=`
- **N:** `N=3`

Prüfsumme:

- **valSumCalc:** `qx6p`

Referenzen – Muster-Code

Code für Prüfsummenerzeugung:

- Klasse:
  - `at.asitplus.regkassen.demo.AESCheckSum`
  - https://github.com/a-sit-plus/at-registrierkassen-mustercode/blob/master/regkassen-democashbox/src/main/java/at/asitplus/regkassen/demo/AESCheckSum.java

~~6~~7 Automatisierte Tests – Überblick

Um die Korrektheit der Implementierung einer RKSV-konformen Kasse zu überprüfen, werden in den Abschnitten 8 und 9 Testfallformate und Testfälle definiert, die von den Herstellern für das Durchführen von automatisierten Tests verwendet werden können. In einer späteren~~Ab~~ Version werden~~0.7 werde~~ die Prüftools in der Lage sein die Abdeckung dieser Testfälle überprüfen zu können. Generell können die Testfälle für zwei prinzipielle Prozessflüsse verwendet werden:

Testen mit den vordefinierten Test-Suites:

1. Eine Kassenimplementierung liest die definierten Testfälle ein. Für jeden Testfall werden die folgenden Prozesse durchgeführt:
    a. Die Kasse liest den Testfall ein, initialisiert die Kasse mit den vorgegebenen Parametern (z.B. AES Schlüsseln, Kassen-ID etc.) und erstellt die im jeweiligen Testfall angegebenen Belege in der vorgegebenen Reihenfolge mit den vorgegebenen Eigenschaften.
    b. Die erstellen Belege und Metadaten werden wie folgt exportiert:
        i. Es wird ein Verzeichnis mit dem Namen des jeweiligen Testfalls erstellt.
        ii. In dieses Verzeichnis wird ein RKSV DEP-Export durchgeführt und in in der Datei `dep-export.json` gespeichert.
        iii. Zusätzlich wird in dem Verzeichnis die Datei `cryptographicMaterialContainer.json` abgelegt, die die kryptographischen Daten der Kasse (AES-Schlüssel, Signaturzertifikate) enthält.
2. ACHTUNG DER FOLGENDE TEXT IST FÜR DIE AKTUELLE VERION DES PRÜFTOOLS NICHT GÜLTIG: ~~GILT ERST AB PRÜFTOOL VERSION 0.7:~~
   Die erstellten Verzeichnisse (entsprechend den Testfällen) mit den exportierten Daten werden dem Prüftool übergeben. Das Prüftool verfügt über die vordefinierten Testfälle und kann daher **(1)** feststellen ob es sich um korrekte Belege und Belegfolgen handelt und **(2)** ob die vorgegebenen Testfälle korrekt abgebildet wurden.

Testen mit beliebigen Belegfolgen:

ACHTUNG DER FOLGENDE TEXT IST FÜR DIE AKTUELLE VERION DES PRÜFTOOLS NICHT GÜLTIG:~~GILT ERST AB PRÜFTOOL VERSION 0.7~~

Neben den vordefinierten Test-Suites können beliebige eigenständige Test-Suites im angegebenen Format (siehe Abschnitt 8.1) erstellt werden. Bei der Prüfung der erstellten Belege können diese selbst definierten Test-Suites dem Prüftool übergeben werden. Das Prüftool wird dahingehend sowohl die Korrektheit der Belege und deren Abfolge als auch die Abdeckung des angegebenen Testfalls überprüfen.

~~7~~8 Formatdefinitionen für Testfälle

~~7.1~~8.1 *Testsuite-Format*

Die Testfälle werden im JSON-Format zur Verfügung gestellt, um es den Herstellern zu ermöglichen automatisierte Tests für Kassensysteme zu integrieren. Ein Testfall besteht im Wesentlichen aus den folgenden Daten:

- Parameter für die Initialisierung der Kasse: z.B. AES Schlüssel, Kassen-ID
- Definition einer beliebigen Abfolge von Belegen, wobei für jeden Beleg die Belegdaten (Belegnummer, Datum/Uhrzeit etc.) vorgegeben werden.

Diese Daten werden in einem JSON-Objekt zur Verfügung gestellt. Das verwendete Format wird im Folgenden erklärt und es wird angegeben welche Daten vom Prüftool in späteren Versionen~~Version 0.7~~ auch im Sinne der Vollständigkeit der Testfälle überprüft werden. Bestimmte Felder werden – obwohl sie im Prüftool nicht für die Vollständigkeit der Testfälle verwendet werden – dennoch im Testfall Format angegeben, um hier in Zukunft sehr spezifische Abläufe abbilden zu können.

Ein Testfall ist dabei in der folgenden Struktur aufgebaut (Java Repräsentation, siehe o https://github.com/a-sit-plus/at-registrierkassen-mustercode/tree/master/regkassen-core/src/main/java/at/asitplus/regkassen/core/base/cashboxsimulation/CashBoxSimulation.java):

- **cashBoxId** (**JSON *string***): Entspricht dem Feld **Kassen-ID** (siehe Prozess 2.1).
  ~~Ab Prüftool Version 0.7:~~ Dieses Feld wird vom Prüftool nicht überprüft. D.h. im Testfall ist zwar eine **Kassen-ID** vorgegeben, bei der Prüfung des Testfalls wird aber nicht darauf geachtet ob auch diese **Kassen-ID** bei der Erstellung der Belege verwendet wurde. Es kann also eine beliebige **Kassen-ID** beim Abarbeiten des Testfalls verwendet werden.
- **base64AesKey** (**JSON *string***): Entspricht dem BASE64-kodierten AES-256 Schlüssel der Kasse.
- **companyID** (**JSON *string***): Entspricht dem Ordnungsbegriff des Unternehmens. Das Format (ohne der Key-ID) wird in Prozess 2.1 für das Feld **Zertifikat-Seriennummer** definiert.
- **simulationRunLabel** (**JSON *string***): Hierbei handelt es sich um ein eindeutiges Kennzeichen für den angegebenen Testfall (der Dateiname des Testfalls entspricht diesem Label). Dieser Name wird vom Prüftool verwendet, um die Vollständigkeit der zu überprüfenden Testfälle zu überprüfen (in einer späteren~~ab~~ Version ~~0.7~~).
- **numberOfSignatureDevices** (**JSON *number***): Im Testfall wird auch die Anzahl der verwendeten Signatureinrichtungen definiert. Auf diese Signatureinrichtungen wird in den weiter unten genannten Informationen zu Belegerstellung referenziert. Da nicht davon auszugehen ist, dass jedes Kassensystem eine beliebige Anzahl von Signatureinrichtungen verwenden kann (vor allem wenn es sich dabei um produktive Signatureinrichtungen handelt) wird die verwendete Signatureinheit und deren tatsächliche Anzahl nicht vom Prüftool überprüft. Dieser Parameter dient im Wesentlichen dazu eine schnelle Setup-Möglichkeit für eine beliebige Anzahl von Test-Signatureinrichtungen zu ermöglichen.
- **cashBoxInstructionList** (**JSON *array***): Diese Liste enthält Vorlagen für die zu erstellenden Belege. Die Reihenfolge der zu erstellenden Belege entspricht der Reihenfolge in dieser Liste.
  - **signatureDeviceDamaged** (**JSON *string***):
    `true` gibt an, dass bei diesem Beleg die verwendete Signatureinrichtung beschädigt ist, die Signaturerstellung also anhand von Prozess 3.2 abgewickelt werden muss.
    `false` gibt an, dass die Signaturerstellungseinheit funktionsfähig ist und die Signaturerstellung nach Prozess 3.1 abgewickelt werden muss.
  - **receiptIdentifier** (**JSON *string***): Entspricht dem Feld **Belegnummer** (siehe Prozess 2.1).
    ~~Ab Prüftool Version 0.7:~~ Dieses Feld wird vom Prüftool nicht überprüft. D.h. im

Testfall ist zwar eine Belegnummer vorgegeben, bei der Prüfung des Testfalls wird aber nicht darauf geachtet ob auch diese **Belegnummer** bei der Erstellung des jeweiligen Beleges verwendet wurde. Es kann also eine beliebige **Belegnummer** beim Abarbeiten des Testfalls verwendet werden.

- **dateToUse** (**JSON *string***): Dieses Datum soll als Belegerstellungszeitpunkt verwendet werden. Das Format entspricht dem des Feldes **Beleg-Datum-Uhrzeit** das in Prozess 2.1 definiert ist.
  ~~Ab Prüftool Version 0.7:~~ Dieses Feld wird vom Prüftool nicht überprüft. D.h. im Testfall ist zwar ein Belegerstellungszeitpunkt vorgegeben, bei der Prüfung des Testfalls wird aber nicht darauf geachtet ob dieser auch bei der Erstellung des jeweiligen Beleges verwendet wurde. Es kann also ein beliebiger Belegerstellungszeitpunkt beim Abarbeiten des Testfalls verwendet werden.
- **usedSignatureDevice** (**JSON *number***): Ganzzahliger Index (beginnend bei 0) der die verwendete Signaturerstellungseinheit angibt.
  ~~Ab Prüftool Version 0.7:~~ Das Prüftool überprüft nicht welche Signaturstellungseinheit für die Belegsignierung verwendet worden ist. Die Belege der Testfälle können also mit einer beliebigen Signatureinrichtung aus einer beliebigen Menge von Signatureinrichtungen signiert werden.
- **typeOfReceipt** (**JSON *string***): Enthält den Typ des zu erstellenden Belegs. Die möglichen Werte entsprechen den Belegtypen *START_BELEG, STANDARD_BELEG, STORNO_BELEG, TRAINING_BELEG und NULL_BELEG*
- **simplifiedReceipt** (**JSON *object***):
  - **taxSetNormal** (**JSON *number***): entspricht dem Wert des Felds **Betrag-Satz-Normal** (siehe Prozess 2.1). In diesem Fall wird dem JSON-Format entsprechend das Zeichen `.` als Dezimaltrennzeichen verwendet.
  - **taxSetErmaessigt1** (**JSON *number***): entspricht dem Wert des Felds **Betrag-Satz-Ermaessigt1** (siehe Prozess 2.1). In diesem Fall wird dem JSON-Format entsprechend das Zeichen `.` als Dezimaltrennzeichen verwendet.
  - **taxSetErmaessigt2** (**JSON *number***): entspricht dem Wert des Felds **Betrag-Satz-Ermaessigt2** (siehe Prozess 2.1). In diesem Fall wird dem JSON-Format entsprechend das Zeichen `.` als Dezimaltrennzeichen verwendet.
  - **taxSetNull** (**JSON *number***): entspricht dem Wert des Felds **Betrag-Satz-Null** (siehe Prozess 2.1). In diesem Fall wird dem JSON-Format entsprechend das Zeichen `.` als Dezimaltrennzeichen verwendet.
  - **taxSetBesonders** (**JSON *number***): entspricht dem Wert des Felds **Betrag-Satz-Besonders** (siehe Prozess 2.1). In diesem Fall wird dem JSON-Format entsprechend das Zeichen `.` als Dezimaltrennzeichen verwendet.

Beispiele:
Im folgenden Beispiel ist ein Testfall definiert, der eine Kasse initialisiert die anschließend vier Belege (Startbeleg, Standardbeleg, Trainingsbeleg, Nullbeleg) erstellt. Für das Signieren der Belege sollen 2 Signaturerstellungseinheiten verwendet werden. Dieser werden in den Beleginstruktionen mit den Indices 0 und 1 identifiziert.

```
{
  "cashBoxId": "CASHBOX-DEMO-1",
  "base64AesKey": "4wG+kZghA5c/259BW9o+40m1Q9Co27l29YHQUSrPFU4=",
  "companyID": "U:ATU12345678",
  "simulationRunLabel": "test-case-1",
  "numberOfSignatureDevices": 2,
  "cashBoxInstructionList": [
    {
      "signatureDeviceDamaged": false,
      "receiptIdentifier": "A1",
      "dateToUse": "2016-02-01T17:50:10",
      "usedSignatureDevice": 0,
```

Formatiert: Block, Muster: Transparent (Hintergrund 1)

```
      "simplifiedReceipt": {
        "taxSetNormal": 0.0,
        "taxSetErmaessigt1": 0.0,
        "taxSetErmaessigt2": 0.0,
        "taxSetNull": 0.0,
        "taxSetBesonders": 0.0
      },
      "typeOfReceipt": "START_BELEG"
    },
    {
      "signatureDeviceDamaged": false,
      "receiptIdentifier": "A2",
      "dateToUse": "2016-03-05T15:31:23",
      "usedSignatureDevice": 1,
      "simplifiedReceipt": {
        "taxSetNormal": 2.1,
        "taxSetErmaessigt1": 2.0,
        "taxSetErmaessigt2": 0.0,
        "taxSetNull": 0.0,
        "taxSetBesonders": 1.0
      },
      "typeOfReceipt": "NORMALER_BELEG"
    },
    {
      "signatureDeviceDamaged": false,
      "receiptIdentifier": "A3",
      "dateToUse": "2016-03-08T15:37:23",
      "usedSignatureDevice": 1,
      "simplifiedReceipt": {
        "taxSetNormal": 2.0,
        "taxSetErmaessigt1": 2.24,
        "taxSetErmaessigt2": 0.0,
        "taxSetNull": 19.26,
        "taxSetBesonders": 1.0
      },
      "typeOfReceipt": "TRAINING_BELEG"
    },
    {
      "signatureDeviceDamaged": false,
      "receiptIdentifier": "A4",
      "dateToUse": "2016-03-08T15:40:23",
      "usedSignatureDevice": 1,
      "simplifiedReceipt": {
        "taxSetNormal": 0.0,
        "taxSetErmaessigt1": 0.0,
        "taxSetErmaessigt2": 0.0,
        "taxSetNull": 0.0,
        "taxSetBesonders": 0.0
    },
    "typeOfReceipt": "NULL_BELEG"
  ]
}
```


~~7.2~~8.2 Output-Format

Für jeden Test-Fall wird ein eigenes Verzeichnis angelegt, das den Namen des Testfalls erhält. In diesem Verzeichnis werden unterschiedliche Dateien/Verzeichnisse gespeichert. Die folgenden Dateien bzw. deren Formate haben zwar für eine produktive Kasse (mit Ausnahme des DEP-Exports) keine Bedeutung, allerdings spielen sie bei der Überprüfung der Implementierung der

Kasse eine wichtige Rolle, da die Dateien vom Prüftool verwendet werden, um die Testfälle einer Kasse und vor allem deren Abdeckung prüfen zu können (Die Abdeckung der Testfälle wird bei der Prüfung nicht berücksichtigt~~in Version 0.7 des Prüftools geplant~~):

`dep-export.json`
In dieser Datei werden die erstellten Belege im DEP Export Format (Detailspezifikation, Abs 3) gespeichert.

`cryptographicMaterialContainer.json`
Die Datei enthält den AES-Schlüssel der Kasse sowie die verwendeten Signaturzertifikate (offenes System oder geschlossenes System) oder öffentliche Schlüssel (geschlossenes Gesamtsystem). Für die Datei wird folgendes JSON-Objekt verwendet:

- **base64AESKey (JSON *string*)**: Der BASE64-kodierte AES-256 Schlüssel mit dem der Umsatzzähler der Kasse verschlüsselt wird.
- **certificateOrPublicKeyMap (JSON *object*)**: Hierbei handelt es sich um ein JSON-Objekt, das die verwendeten Signaturzertifikate (offenes oder geschlossenen System) oder die öffentlichen Schlüssel (geschlossenes System) abbildet. Beide Varianten werden wie folgt abgebildet:
  - **id (JSON *string*)**: Dieser Wert erhält das Identifikationsmerkmal des verwendeten Signaturzertifikats oder des öffentlichen Schlüssels. Bei einem offenen System ist das die Hexadezimaldarstellung der Seriennummer des Zertifikats. Bei einem geschlossenen System ist dies der Ordnungsbegriff des Unternehmens ergänzt um die Key-ID des verwenden Schlüssels (siehe Beispiele unten bzw. Feld **Zertifikat-Seriennummer** in Prozess 2.1).
    - **id (JSON *object*)**: Das unter **id** gespeicherte JSON-Objekt hat folgende Struktur:
      - **id (JSON *string*)**: enthält nochmal den Wert des vorher genannten Feldes **id**
      - **signatureDeviceType (JSON *string*)**:
        - **CERTIFICATE**: Handelt es sich um ein Zertifikat, wird hier der Typ `CERTIFICATE` angegeben (gültig für offene und geschlossene Systeme).
        - **PUBLIC_KEY**: Handelt es sich um einen öffentlichen Schlüssel, wird der Typ `PUBLIC_KEY` angegeben (gültig nur für geschlossene Systeme).
      - **signatureCertificateOrPublicKey (JSON *string*)**:
        - Ablage eines X509-Zertifikats (signatureDeviceType = `CERTIFICATE`). In diesem Fall wird das X509-Zertifikat im DER-Format dargestellt und BASE64-kodiert abgelegt.
        - Ablage eines öffentlichen Schlüssels (signatureDeviceType = `PUBLIC_KEY`). Der Schlüssel wird im DER/ASN1 Format abgelegt, das für die die Repräsentation von öffentlichen Schlüsseln in X509 Zertifikaten verwendet wird.[8]

Beispiel für ein offenes System mit drei Signaturerstellungseinheiten:
Im folgenden Beispiel erkennt man aufgrund der Darstellung der Seriennummer des Zertifikats (Hexadezimaldarstellung), dass es sich um ein offenes System handelt. Bei einem geschlossenen System würde der Ordnungsbegriff des Unternehmens ergänzt um die Key-ID verwendet werden.

```
{
```

Formatiert: Muster: Transparent (Hintergrund 1)

[8] RFC 3279 (https://tools.ietf.org/html/rfc3279)
RFC 5480 (https://tools.ietf.org/html/rfc5480)
BASE64-dekodierter Wert kann mit openssl verarbeitet werden:
`openssl ec -in key.file -pubin -inform DER -text`

```
      "base64AESKey": "BwZ8o6bv2XoABMSeLDvr5VrhNpwWLDklS1plk3TeBqU=",
      "certificateOrPublicKeyMap": {
        "816d29a919370f0c": {
          "id": "816d29a919370f0c",
          "signatureDeviceType": "CERTIFICATE",
          "signatureCertificateOrPublicKey":
"MIIBSTCB8KADAgECAgh+ktZW5sjw9DAKBggqhkjOPQQDAjAWMRQwEgYDVQQDDAtSZWdLYXN
zYSBDQTAeFw0xNjAzMDUxOTQ1MDJaFw0xNjAzMDYxOTQ1MTJaMB4xHDAaBgNVBAMME1NpZ25
pbmcgY2VydGlmaWNhdGUwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAQg/H2XYKNKhowvHES
HADoKvw2V63+A2acOUPmdbR51Vbca4lvHEr1S1843WNuxILXXZURn0740m0JmGeg2H5+goyA
wHjAMBgNVHRMBAf8EAjAAMA4GA1UdDwEB/wQEAwIHgDAKBggqhkjOPQQDAgNIADBFAiAYj25
MfdCWtyxSt3/ggkFvdJ/WIbiq0Oe99gwfbOnwaAIhANApaLXif7PDXvL4ombusXwEVtZg3AB
VTqviyb9Uu31T"
        },
        "69496db7ddc2c4f9": {
          "id": "69496db7ddc2c4f9",
          "signatureDeviceType": "CERTIFICATE",
          "signatureCertificateOrPublicKey":
"MIIBSjCB8KADAgECAghpSW233cLE+TAKBggqhkjOPQQDAjAWMRQwEgYDVQQDDAtSZWdLYXN
zYSBDQTAeFw0xNjAzMDUxOTQ1MDJaFw0xNjAzMDYxOTQ1MTJaMB4xHDAaBgNVBAMME1NpZ25
pbmcgY2VydGlmaWNhdGUwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAATfJN1E3YIooFRP+da
QCtUZUOiYAvkhS+VqR+rRdn2UAgxqtUC9sXmLOqK3VK7GZRUHL18jl4Yu4s1QnFzI4m/NoyA
wHjAMBgNVHRMBAf8EAjAAMA4GA1UdDwEB/wQEAwIHgDAKBggqhkjOPQQDAgNJADBGAiEAt1X
3jANM7Wk7ropwHmYPgpBgvj27JFNiaWUgn1i6ACQCIQCc9IxELqY+h2m6szroEIMwkX3/RQa
inR68YFb349UTRg=="
        },
        "bef17ac5afa0bee": {
          "id": "bef17ac5afa0bee",
          "signatureDeviceType": "CERTIFICATE",
          "signatureCertificateOrPublicKey":
"MIIBSjCB8KADAgECAggL7xesWvoL7jAKBggqhkjOPQQDAjAWMRQwEgYDVQQDDAtSZWdLYXN
zYSBDQTAeFw0xNjAzMDUxOTQ1MDJaFw0xNjAzMDYxOTQ1MTJaMB4xHDAaBgNVBAMME1NpZ25
pbmcgY2VydGlmaWNhdGUwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAARnSLvf5izHR8m0agp
98TheIbE44RLhPt7twWL5Rkw8Dtc4A9MvgI1CjHf5XVQm8r+LK/J8Rowvzhjrx7hEJgpJoyA
wHjAMBgNVHRMBAf8EAjAAMA4GA1UdDwEB/wQEAwIHgDAKBggqhkjOPQQDAgNJADBGAiEAomx
20AbXg4WVHjTnI6HKF/w/o3FZq4zOd/Z6rI7g9RICIQD2lfaHJ24hi119sDsbnB1MrX6q102
AFGeefZYhJF5hBg=="
        }
      }
    }
```


Beispiel für ein geschlossenes System mit drei Signaturerstellungseinheiten:
Im folgenden Beispiel erkennt man aufgrund des Vorhandenseins des Ordnungsbegriffs des Unternehmens und der Key-ID, dass es sich um ein offenes System handelt. In in diesem Fall ist das die UID `U:ATU12345678` ergänzt um die Key-ID (in diesem Beispiel `K0`, `K1` und `K2`).


```
    {
      "base64AESKey": "rajKSkCAOBycpmg/8+ScFQiio1viw/PCe26sOeFndNc=",
      "certificateOrPublicKeyMap": {
        "U:ATU12345678-K0": {
          "id": "U:ATU12345678-K0",
          "signatureDeviceType": "PUBLIC_KEY",
          "signatureCertificateOrPublicKey":
"MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEe/t5YftuNCWttEK81aItJ/n25JW1oGd7OZr
kcjL8IspCR1yF4RYJ9moAorvTz42OZ7oQbaxESNnJR4DydPeRvQ=="
        },
        "U:ATU12345678-K2": {
```


Formatiert: Muster: Transparent (Hintergrund 1)

```
          "id": "U:ATU12345678-K2",
          "signatureDeviceType": "PUBLIC_KEY",
          "signatureCertificateOrPublicKey":
"MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEXOyYa/QsGjKBDSbGj4U/dxrQJlRUQDKnCpl
RED4crDRt0jCQwAp9rcdoIsrY25y5nvUKRRgXcuLIQpJXzOS1yA=="
        },
        "U:ATU12345678-K1": {
          "id": "U:ATU12345678-K1",
          "signatureDeviceType": "PUBLIC_KEY",
          "signatureCertificateOrPublicKey":
"MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAELzQ2KdjGmQ9wXWQBd4IyGCtyWFcrYjdahWu
Pqt+dyeqLcF+Vqjocla/fMlErfAUNrJCAMV7LbyLqF6ncROiPYA=="
        }
      }
    }
```

~~8~~9 Testfälle

Ziel des vorliegenden Dokuments ist es, ein möglichst umfassendes Spektrum an Testszenarien zu generieren. Dazu werden zwei Strategien verfolgt.

- Die erste Strategie zielt darauf ab alle möglichen Transitionen zwischen Belegtypen systematisch abzudecken. Hier wird nicht auf typische reale Prozessfälle geachtet, sondern der Fokus darauf gelegt alle möglichen Transitionen abzudecken. Die Kernfunktionen einer Kasse können damit sehr effizient überprüft werden.
- Die zweite Strategie zielt darauf ab realistische Beleg-Abfolgen zu definieren, über die bestimmt Aspekte einer Registrierkassa getestet werden können. Diese Testfälle entsprechen auch den typischen Fällen bei der Anwendung einer Kasse (Inbetriebnahme, Ausfall der Signatureinrichtung, Sonderfälle). ~~(Geplant für Version 0.7 des Prüftools)~~

Basierend auf diesen beiden Strategien werden in den folgenden Abschnitten Testszenarien definiert. Ein Testszenario entspricht dabei einer Abfolge von Belegerstellungen, die von einer Registrierkasse unter Berücksichtigung der gesetzlichen Vorgaben durchgeführt werden muss.

~~8.1~~9.1 *Vollständige Testszenarien*

In diesem Abschnitt werden 8 Testszenarien definiert. Da pro Testszenario jeweils nur ein Startbeleg erstellt werden kann, es jedoch 8 Belegtypen (4 Typen jeweils mit/ohne SE) gibt, die auf einen Startbeleg folgen können, sind zumindest 8 Testszenarien notwendig, um alle identifizierten Transitionen abzudecken. Um möglichst umfangreiche Tests zu gewährleisten, beinhaltet jedes Testszenario alle möglichen Transitionen zumindest einmal (mit Ausnahme jener, die aufgrund der Startbeleg-bezogenen Einschränkung nicht durchgeführt werden können).

Obwohl alle Testszenarien alle Transitionen zumindest einmal enthalten, wurden die acht Szenarien so erstellt, dass sie sich deutlich unterscheiden. Einige Transitionen werden in jedem Testszenario mehrfach durchlaufen. Diese Notwendigkeit ergibt sich aus den Einschränkungen, denen mögliche Transitionen unterworfen sind. Das notwendige mehrfache Durchlaufen mancher Transitionen erklärt, warum die acht im Folgenden beschriebenen Testszenarien mitunter eine unterschiedliche Anzahl an Belegerstellungen definieren und die Anzahl der Belegerstellungen immer pro Testszenario immer über der Anzahl unterschiedlicher Transitionen liegt.

Diese Testszenarien stehen in dem in Abschnitt 8.1 beschriebenen Format unter https://github.com/a-sit-plus/at-registrierkassen-mustercode/releases zur Verfügung und werden auch von der Demokasse abgearbeitet: `TESTSUITE_TEST_SZENARIO_{1-8}.json`

~~8.2~~9.2 *Realistische Testszenarien*

Eingeplant für eine spätere Version dieses Dokuments.

~~9~~10 Use Cases

~~9.1~~10.1 *Inbetriebnahme der Kasse*

Eingeplant für eine spätere Version dieses Dokuments.

~~9.2~~10.2 *Erstellung von Standardbelegen*

Eingeplant für eine spätere Version dieses Dokuments.

~~9.3~~10.3 *Erstellung von Stornobelegen*

Eingeplant für eine spätere Version dieses Dokuments.

~~9.4~~10.4 *Erstellen eines Nullbelegs (Jahres- Monats- Tagesbeleg, Sammelbeleg nach Ausfall, Schlussbeleg etc.)*

Eingeplant für eine spätere Version dieses Dokuments.

~~9.5~~10.5 *Ausfall der Sicherheitseinrichtung*

Eingeplant für eine spätere Version dieses Dokuments.

~~9.6~~10.6 *Exportieren des DEP Protokolls*

Eingeplant für eine spätere Version dieses Dokuments.