Anwenderdokumentation PersoSim

V1.5.1

Die nachfolgende Anwenderdokumentation soll dem Anwender bei der Installation und den ersten Schritten im Umgang mit PersoSim helfen.

Inhaltsverzeichnis

1 Komponenten & Installation

PersoSim ist auf den 64-Bit Betriebssystemen Windows, Linux und MacOS lauffähig. Grundvoraussetzung für den Betrieb von PersoSim ist eine Java Laufzeitumgebung ab Version 17.

1.1 Simulator

Die Hauptanwendung PersoSim enthält den Karten-Simulator und gleichzeitig den virtuellen Kartenleser der vom Treiber (siehe unten) an das jeweilige Betriebssystem angebunden wird. Die Anwendung erfordert keine Installation im klassischen Sinne. Sie ist nach dem Entpacken der Zip-Datei bzw. des Tar-Balls sofort lauffähig.

Der Start erfolgt betriebssystemabhängig über die PersoSim.exe unter Windows bzw. PersoSim unter Linux oder MacOS. Nach dem Start erscheint die jeweilige GUI wie in Abbildung 1. Je nach Anwendungsfall (z.B. bei einer Anbindung per RemoteIFD) kann der Simulator jetzt unmittelbar genutzt werden.

1.2 Treiber

Um den virtuellen Kartenleser (und damit die simulierten Ausweise) regulär nutzen zu können, stehen betriebssystemspezifische Treiber zur Verfügung. Diese binden PersoSim als Kartenleser im jeweiligen PC/SC-Stack ein und erlauben damit den Zugriff aus jeder kompatiblen Anwendung.
Das Vorgehen für die Installation des Treibers ist betriebssystemabhängig. Im Folgenden wird die Installation für Windows, Linux und MacOS beschrieben.

Windows Für die Installation von PersoSim wird lediglich als Betriebssystem Windows 10 oder neuer vorausgesetzt.
Der Treiber erkennt selbständig, um welche Version von Windows es sich handelt und erkennt auch automatisch die Architektur (32 oder 64 Bit). Zur Installation des Treibers muss lediglich den Anweisungen des Installationsprogramms Folge geleistet werden. Im Gegensatz zu älteren Versionen des Treibers müssen keine Testzertifikate mehr separat installiert werden, allerdings kann es unter Windows notwendig sein, die Treibersignaturprüfung für die Installation zu deaktivieren. Nach erfolgreicher Installation erscheint der virtuelle Kartenleser ähnlich wie physische Kartenleser als ”PersoSim Virtual Reader” im Gerätemanager unter Smartcard-Leser und kann genutzt werden.

Linux Die folgende Anleitung zur Installation des Treibers unter Linux wurde für Ubuntu 12.04 LTS erstellt und dort getestet. Sie ist aber grundsätzlich auch auf anderen Linux-Derivaten durchführbar.
Zusätzlich zur Ubuntu 12.04 LTS Standardinstallation werden die folgenden Pakete benötigt:

- pcscd
- pcsc-tools
- libpcsclite-dev

Diese Pakete können per *sudo apt-get install pcscd pcsc-tools libpcsclite-dev* installiert werden.
Nach dem Entpacken des Archivs liegt der Treiber im Quellcode vor und muss zunächst kompiliert werden. Dies geschieht mit Hilfe des Aufrufs *make* im Verzeichnis des entpackten Archivs. Die anschließende Installation erfolgt mit dem Aufruf *sudo make install*. Der Treiber wird beim nächsten Start des PC/SC-Daemon (*sudo service pcscd restart*) automatisch geladen.

MacOS Um die notwendige Toolchain auf MacOS bereitzustellen muss Xcode installiert werden. Zusätzlich müssen die ”command line developer tools” installiert werden. Das genaue Vorgehen hängt von der genutzten MacOS- und Xcode-Version ab. Für aktuelle Versionen sollte der Konsolenbefehl *xcode-select --install* zu einem Dialog für die Installation der Kommandozeilenwerkzeuge führen.
Nach dem Entpacken des Archivs liegt der Treiber im Quellcode vor und muss zunächst kompiliert werden. Dies geschieht mit Hilfe des Aufrufs *make* im Verzeichnis des entpackten Archivs. Die anschließende Installation erfolgt mit dem Aufruf *sudo make install*. Nach einem Neustart wird der virtuelle Treiber von MacOS erkannt und kann mit PersoSim genutzt werden.

1.3 Editor

Der PersoSim Editor erlaubt das Modifizieren bestehender Profile oder auch die Erzeugung völlig neuer Personalisierungen für den Simulator. Damit können Karten mit beliebigen Inhalten simuliert werden.

Analog des Simulators erfordert auch der PersoSim Editor keine Installation im klassischen Sinne. Installation und Start funktionieren identisch durch Entpacken des Archivs und anschließendem Ausführen des enthaltenen Executables. Nach dem Start erscheint die GUI wie in Abbildung 6 in Kapitel 3.

2 Benutzung des PersoSim Simulators

2.1 Benutzeroberfläche

PersoSim stellt im Vordergrund die Benutzeroberfläche des eingebundenen virtuellen Kartenlesers dar (siehe Abbildung 1). Alle weiteren Funktionen sind über Tabs und Menüs erreichbar und werden nachfolgend erläutert.

Abbildung 1: PersoSim Hauptfenster

2.2 Laden von Personalisierungen

Personalisierungen innerhalb von PersoSim lassen sich sowohl über die Kommandozeile als auch mit Hilfe von Menüpunkten über die grafische Benutzeroberfläche laden. Wird keine Personalisierung explizit ausgewählt so wird standardmäßig Profil 1 verwendet. Nach dem Laden einer Personalisierung ist diese sofort aktiviert. Dies entspricht dem Auflegen eines Ausweises auf einen Hardware-Kartenleser. Es stehen diverse vorgefertigte Standardprofile zur Auswahl. Hierbei handelt es sich z.B. um Personalausweise sowie Unionsbürgerkarten. Eine aktuelle Übersicht der verschiedenen Personalisierungen und der darin verwendeten Daten befindet sich auf der PersoSim-Projektseite im Bereich Downloads.

Laden von Personalisierungen über das Menü: Die wohl komfortabelste Möglichkeit, Personalisierungen zu laden, ist das Anwendungsmenü. Der Menüpunkte ”Personalization” bietet hierfür folgende Unterpunkte:

- **Select Perso from file** Öffnet einen Datei-Dialog und erlaubt die Auswahl einer beliebigen Personalisierung aus einer vorliegenden XML-Datei.
- **Select predefined Perso template** Erlaubt die direkte Auswahl der oben beschriebenen Standardprofile.
- **Remove card „**entfernt“ die simulierte Karte von Leser.

Laden von Personalisierungen über die Kommandozeile: Auch auf der Kommandozeile (siehe Abschnitt 2.4) lassen sich Personalisierungen einfach laden. Hierfür wird das Kommando loadperso<Dateiname> benutzt. Alternativ kann statt des Dateinamens auch eine Zahl angegeben werden, die als Abkürzung zu einem der angebotenen Standardprofile dient.

2.3 Wechseln des verwendeten Lesertyps

Der virtuelle Kartenleser, der im Rahmen von PersoSim genutzt werden kann, kann so konfiguriert werden, dass er sowohl einen Basis- als auch ein Standardleser simuliert. Die Auswahl des Lesertyps erfolgt über den Menüpunkt ”Reader Type”.

Basisleser besitzen in der Regel weder Display noch Tastatur. Dementsprechend verfügt auch der PersoSim-Basisleser lediglich über die beiden Infofelder im oberen Bereich, die anzeigen, ob eine Karte simuliert und über welche Verbindung die Simulation bereitgestellt wird (vgl. Abbildung 2).

Abbildung 2: PersoSim als Basisleser

Standardleser verfügen über ein Display und eine Tastatur, über die z.B. Informationen gemäß BSI TR-03119 angezeigt und Passwörter eingegeben werden können. Entsprechend verfügt der PersoSim-Standardleser über ein Display, in dem Anweisungen und die eingegebene Pin angezeigt werden, sowie über ein Tastenfeld zur Eingabe der Pin. Unterhalb des Displays wird der zur Authentisierung übertragene CHAT angezeigt (wenn man mit der Maus darüber verweilt, werden die einzelnen Bits menschenlesbar aufgeschlüsselt).

Zusätzlich erlaubt es der PersoSim-Standardleser im rechten Bereich häufig genutzte Passwörter zu speichern und per Doppelklick direkt zu nutzen. Gerade bei der Durchführung von mehreren Tests kann die Eingabe eines Passworts (PIN, CAN oder PUK) relativ zeitaufwendig werden. Aus diesem Grund gibt es die Möglichkeit, beliebige Ziffernfolgen als Passwörter vorab zu definieren. Abbildung 3 zeigt diese Funktion exemplarisch: Hier sind als Passwörter sowohl ’123456’ (PersoSim default PIN) als auch ’500540’ (CAN für default PersoSim-Profile) hinterlegt. Die Default PUK ‘9876543210‘ könnte ebenfalls dort angegeben werden. Nach einem Rechtsklick auf „Passwords“ im rechten Bereich können Passwörter hinzugefügt, editiert oder gelöscht werden. Bei der Verwendung von PersoSim kann anschließend je nach Kontext das entsprechende Passwort per Doppelklick ausgewählt werden. Zur weiteren Vereinfachung dient die Aktivierung des automatischen Logins (AutoLogin). Wird diese Option nach Auswahl eines Passworts angehakt, wird dieses Passwort immer verwendet und muss nicht mehr per Doppelklick ausgewählt werden.

Beim Programmstart wird automatisch der Standardleser aktiviert.

Es gilt zu beachten, dass die Eingabeelemente auf dem Pinpad des Standardlesers nur

dann und solange aktiviert sind wie sie im Rahmen einer Kommunikation mit dem simulierten Ausweis benötigt werden. D.h. zu Programmstart sind diese zunächst ohne Funktion. Sobald eine Eingabe erforderlich ist, erscheint ein entsprechender Hinweis auf dem Display.

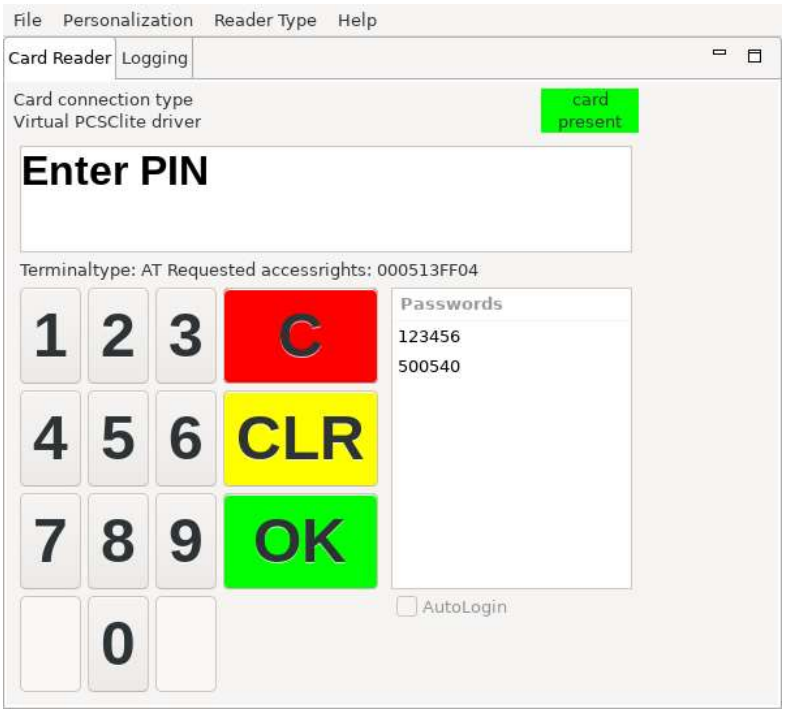

Abbildung 3: PersoSim als Standardleser

2.4 Logging und Konsole

Über den Tab ”Logging” erreicht man die Konsole, in der Log-Informationen über sämtliche Kommandos an den Simulator sowie dessen Rückmeldungen protokolliert werden (siehe Abbildung 4). Hier können über das Kontextmenu sowohl Einstellungen zum Logging-Verhalten vorgenommen als auch der Log-Inhalt in eine Datei geschrieben werden.

Außerdem bietet diese Konsole die Möglichkeit, den Simulator direkt mit Kommandos in der Kommandozeile zu steuern. Eine Liste der verfügbaren Kommandos lässt sich mit dem Kommando help aufrufen.

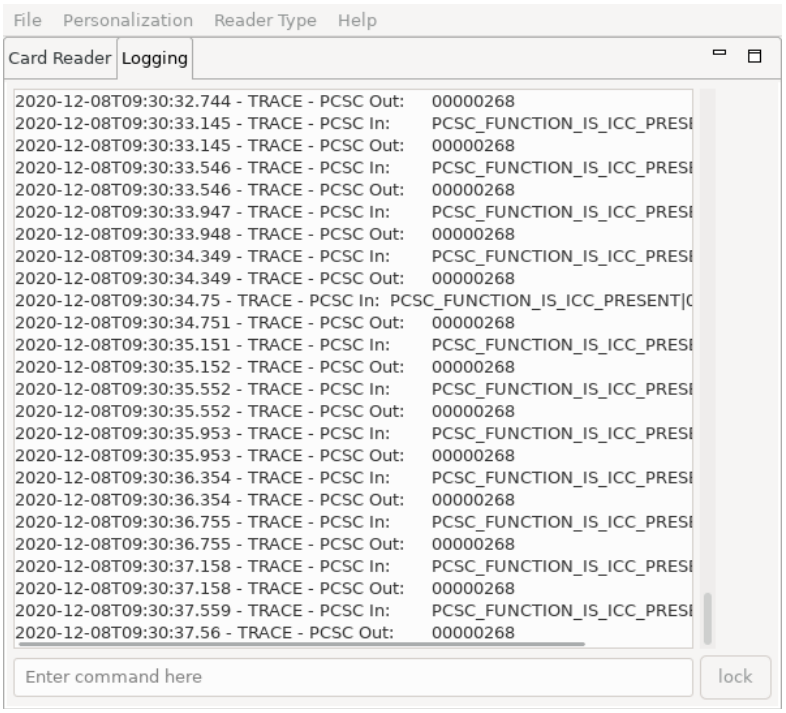

Abbildung 4: PersoSim Konsole

2.5 Smartphone als Kartenleser

Die BSI TR-03112-6 sieht die Funktion ”Remote IFD” vor, auch bekannt unter dem Namen ”Smartphone als Kartenleser”. Mit dieser Funktion lässt sich ein NFC-fähiges

Smartphone als Kartenleser im Kontext des Personalausweises nutzen. Auch PersoSim unterstützt diese Funktion sowohl in der Desktop-Anwendung als auch in der Android- App und kann so von einem eID-Client per Netzwerk angesprochen werden.

Um diese Funktion zu nutzen muss zunächst die Anbindung im Menüpunkt ”Reader Type” → ”UseRemote IFD” entsprechend umgestellt werden. Der aktive Verbindungs- typ wird stets im oberen Bereich der Kartenleser-Oberfläche (oberhalb des PinPads) angezeigt.

Damit ein eID-Client mit PersoSim per Remote IFD kommunizieren kann, müssen die beiden Geräte/Programme miteinander gekoppelt werden. Der entsprechende Dialog ist im Menü”ReaderType” → ”Configure RemoteIFD” erreichbar (Abbildung 5).

Im oberen Bereich werden bereits gekoppelte Anwendungen angezeigt. Bei Bedarf können diese hier per Kontextmenu¨ entfernt werden.

Im mittleren Bereich kann der Kopplungsvorgang durch Klick auf ”Start pairing” initiiert werden. Dies unterbricht vorhandene Verbindungen für die Dauer des Koppelvorgangs und zeigt den notwendigen Kopplungscode an. Mithilfe dieses Kopplungscodes kann sich ein entsprechender eID-Client dann mit PersoSim verbinden (siehe hierzu auch die

Dokumentation des eID-Clients). Nach einer erfolgreichen Kopplung erfolgt die gegenseitige Authentisierung zertifikatsbasiert, sodass keine weitere Kopplung mehr notwendig ist.

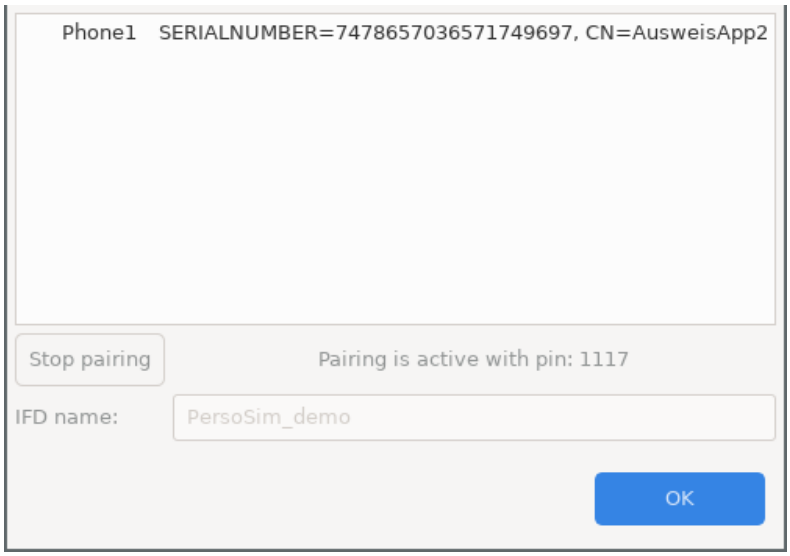

Abbildung 5: Konfiguration Remote IFD

Zu guter Letzt lässt sich im unteren Bereich der Name dieser PersoSim-Instanz konfigurieren. Dieser wird von eID-Clients zur Kopplung und zum Wiederverbinden genutzt, damit der Benutzer die entsprechende Gegenstelle auswählen kann.

3 PersoSim Controller

Der PersoSim Controller ist ein separates Executable und ermöglicht die Fernsteuerung von PersoSim per Konsole auf dem lokalen System. Das Executable liegt parallel zum PersoSim Executable. Es werden die folgenden Kommandos unterstützt:

- Start (um PersoSim zu starten)
- Stop (um PersoSim zu stoppen)
- Restart (um PersoSim neu zu starten)
- LoadPerso (um ein vorgegebenes Profil in PersoSim bereitzustellen)
- SendApdu (um eine APDU an den Simulator zu schicken)
- Reset (um den Simulator analog zum Smartcard-Reset zu resetten)
- Help (um die verfügbaren Kommandos anzuzeigen)

Start, Stop und Restart dienen dazu, das PersoSim Executable bzw. den entsprechenden Prozess zu steuern. Die anderen Kommandos werden an den laufenden PersoSim Prozess per SOAP geschickt. Nachfolgend ist die Ausgabe des Help Kommandos (unter Windows: *PersoSimControllerc.exe help*) dargestellt. Dort werden detailliert die verfügbaren Kommandos beschrieben:

PersoSimControllerc.exe <start|stop|restart> <fullPathToExecutable>
--> Start, stop or restart PersoSim.
Example (Windows): PersoSimControllerc.exe start "C:\PersoSimDir\PersoSimc.exe"
Example (Linux) : PersoSimController stop "/home/someuser/PersoSimDir/PersoSim"
Example (MacOS) : cd /home/someuser/PersoSimDir/PersoSim.app/Contents/Eclipse; ../MacOS/PersoSimController restart ../MacOS/PersoSim
or:
PersoSimControllerc.exe loadperso <absolutePathToPersoFile>|<nameOfPredefinedPersoTemplate>
--> Load a personalization file into PersoSim.
Example (Absolute Path, Windows): PersoSimControllerc.exe loadperso "C:\\PersoSimDir\profiledir\Profile02.perso"
Example (Perso Template) : PersoSimControllerc.exe loadperso Profile03.perso
or:
PersoSimControllerc.exe sendapdu <apduAsHexString>
--> Send an APDU in HEX String format to PersoSim.
Example: PersoSimControllerc.exe sendapdu 0084000000
or:
PersoSimControllerc.exe reset
--> Reset PersoSim analogous to a Smartcard Reset.
or:
PersoSimControllerc.exe help
--> Show this usage information.

Die SOAP-Schnittstelle von PersoSim kann auch direkt unabhängig vom PersoSim Controller verwendet werden. Die zugehörige WSDL kann in einem Standard-Browser aufgerufen werden. Die entsprechende URL lautet:
http://localhost:8890/persosim/PersoSimRemoteControl?wsdl

4 Erstellen eigener Profile mit PersoSim Editor

Zusätzlich zu den mitgelieferten Profilen innerhalb von PersoSim besteht die Möglichkeit, eigene Profile zu erstellen. Profile bestehen aus einem XML-File, das alle relevanten Informationen für die Simulation enthält. Damit der Anwender aber nicht im XML-File editieren muss, gibt es ein zusätzliches Werkzeug zum einfachen Editieren der Profile. Abbildung 6 zeigt die Oberfläche des Editors, der genauso wie der Simulator auf der Website zum Download bereitsteht.

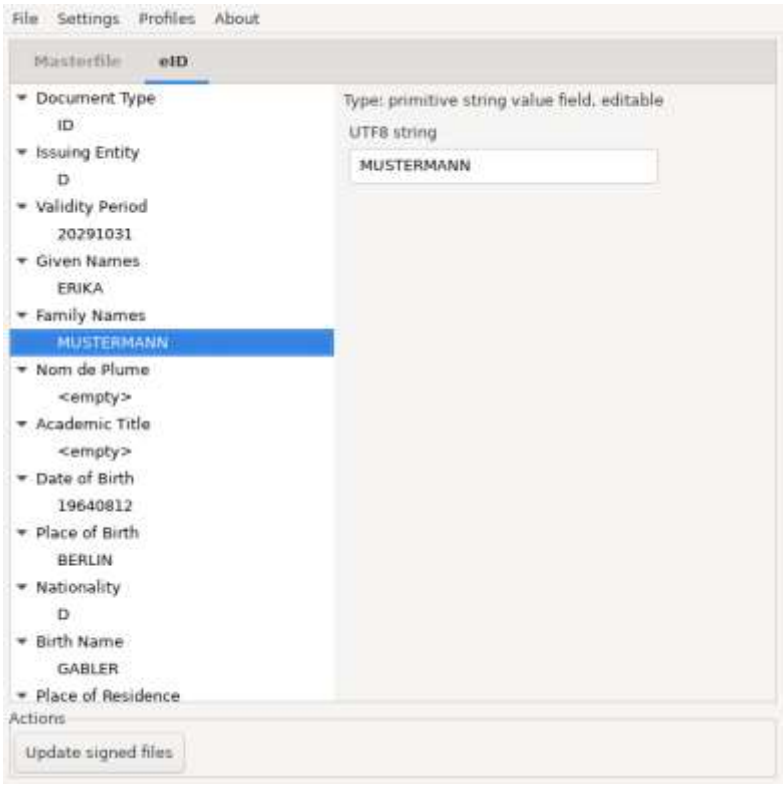

Abbildung 6: PersoSim Editor

Der Editor unterstützt den Anwender bei der Eingabe neuer Profile, indem er auf fehlerhafte Eingaben aufmerksam macht. So weist der Editor beispielsweise darauf hin, sobald der Anwender einen falschen Ländercode eingibt. Als Basis für eigene Profile kann der Anwender natürlich auch die mitgelieferten Profile nutzen.

Unter den Signatur-Einstellungen kann der Anwender ein DS-Zertifikat inkl. DS-Schlüssel eintragen, um das neue Profil auch final zu signieren. Die dazu notwendigen Dateien EF.CardSecurity und EF.ChipSecurity werden dabei neu erstellt, um das Profil zu vervollständigen. Erstellt man ein Profil 'from the scratch' kann man auch ein dazugehöriges EF.CardAccess erstellen lassen. Diese personalausweisspezifischen Dateien enthalten die Standardparameter für Kryptografie und Protokolle, die durch die BSI TR-03110 und BSI TR-03127 für den deutschen Personalausweis vorgegeben werden. EF.CardAccess, EF.CardSecurity und EF.ChipSecurity lassen sich nicht mit dem Editor neu generieren und nicht manuell bearbeiten.

Anhang A: Anpassungen von TA Trust Points innerhalb einer bestehenden Personalisierung

PersoSim erlaubt es dem Benutzer im Rahmen seiner Tests verschiedene Personalisierungen zu verwenden. Zu diesem Zweck stellt PersoSim bereits mehrere Standardprofile mit Personalisierungen bereit, die ein breites Spektrum an Möglichkeiten hierfür abdecken. Neben gewöhnlichen Personalisierungen wie sie auf der überwiegenden Mehrheit ausgegebener Personalausweise vorzufinden sind, finden sich in den Profilen auch solche, die weniger häufig anzutreffende aber nichtsdestotrotz zulässige Sonderfälle sowie deren Kombinationen abdecken. Eine genaue Liste der angebotenen Profile und ggf. ihrer Besonderheiten findet sich auf der PersoSim Projektseite. Jedes einzelne dieser Profile enthält eine zulässige Personalisierung. Eine vollständige und erfolgreiche Verifikation der Profile ist jedoch aufgrund abweichender Signaturen nur innerhalb der Test-PKI möglich.

Um den Austausch und Transfer von Profilen zu vereinfachen, kommen hierfür im Umfeld von PersoSim XML-Dateien zum Einsatz. Dieses Format stellt auch den einfachsten und bevorzugten Weg dar, um Änderungen an den Profilen vorzunehmen.

Diese Anleitung beschreibt, wie sich speziell Trust Points innerhalb einer bestehenden Personalisierung für die Terminal Authentication ändern lassen. Als Trust Points werden im Folgenden alle, einen bestimmten Terminaltyp (AT, IS, ST) authentifizierenden, Zertifikate bezeichnet. Die Menge dieser Trust Points in Bezug auf einen bestimmten Terminaltyp selbst wird hingegen als Trust Anchor bezeichnet.

Im Folgenden ist beschrieben, wie sich grundsätzlich Änderungen an Profilen vornehmen lassen insbesondere aber die an Trust Points.

Ausgangspunkt für alle Änderungen ist jeweils eine XML-Datei zu einem bestehenden Standard- Profil. Im Folgenden wird hierfür beispielhaft das Profil 1 verwendet. Dieses enthält bereits einen TA Trust Anchor für ein AT Terminal, in dem ein einziger Trust Point abgelegt ist.

Der Trust Anchor wird durch das Element `TrustPointCardObject`

```
<de.persosim.simulator.cardobjects.TrustPointCardObject id="continuousId">
        ...
</de.persosim.simulator.cardobjects.TrustPointCardObject>
```

unterhalb des Pfads

```
<de.persosim.simulator.perso.Profile01 id="1">
/<masterFile>
/<children>,
```

dargestellt, siehe auch:

```
<de.persosim.simulator.perso.Profile01 id="1">
        ...
   <mf id="12">
      <children id="13">
            ...
       <de.persosim.simulator.cardobjects.TrustPointCardObject id="77">
            ...
       </de.persosim.simulator.cardobjects.TrustPointCardObject>
            ...
     </children>
            ...
   </mf>
 </de.persosim.simulator.perso.Profile01>
```

Das bereits vorhandene Trust Anchor Element selbst sieht aus wie folgt:

```
 <de.persosim.simulator.cardobjects.TrustPointCardObject id="77">
         <parent class="de.persosim.simulator.cardobjects.MasterFile"
           reference="12"/>
         <children id="78"/>
         <lifeCycleState>CREATION</lifeCycleState>
         <currentCertificate id="79">
           ...
         </currentCertificate>
         <identifier id="94">
           <terminalType>AT</terminalType>
         </identifier>
 </de.persosim.simulator.cardobjects.TrustPointCardObject>
```

Unterhalb des Trust Anchors befindet sich der einzige und derzeitig gültige Trust Point in Form des Elements `currentCertificate`.

```
 <currentCertificate id="79">
           <certificateProfileIdentifier>0</certificateProfileIdentifier>
           <certificateAuthorityReference id="80">
             <countryCode>DE</countryCode>
             <holderMnemonic>TESTeID</holderMnemonic>
             <sequenceNumber>00004</sequenceNumber>
           </certificateAuthorityReference>
           <publicKeyOid id="81">
             <oidByteArray id="82">04007F00070202020203</oidByteArray>
             <idString>id-TA-ECDSA-SHA-256</idString>
           </publicKeyOid>
           <publicKey  id="83">
             <algorithm>EC</algorithm>
```

```
<value>308201333081EC06072A8648CE3D02013081E0020101302C06072A8648CE3D0101022100A
9FB57DBA1EEA9BC3E660A909D838D726E3BF623D52620282013481D1F6E5377304404207D5A0975F
C2C3057EEF67530417AFFE7FB8055C126DC5C6CE94A4B44F330B5D9042026DC5C6CE94A4B44F330B
5D9BBD77CBF958416295CF7E1CE6BCCDC18FF8C07B60441048BD2AEB9CB7E57CB2C4B482FFC81B7A
FB9DE27E1E3BD23C23A4453BD9ACE3262547EF835C3DAC4FD97F8461A14611DC9C27745132DED8E54
5C1D54C72F046997022100A9FB57DBA1EEA9BC3E660A909D838D718C397AA3B561A6F7901E0E829
74856A70201010342000474FF63AB838C73C303AC003DFEE95CF8BF55F91E8FEBCB7395D942036E4
7CF1845EC786EC95BB453AAC288AD023B6067913CF9B63F908F49304E5CFC8B3050DD</value>
          </publicKey>
          <certificateHolderReference id="84">
            <countryCode>DE</countryCode>
            <holderMnemonic>TESTeID</holderMnemonic>
            <sequenceNumber>00004</sequenceNumber>
          </certificateHolderReference>
          <certificateHolderAuthorizationTemplate id="85">
            <objectIdentifier id="86">
              <oidByteArray id="87">04007F000703010202</oidByteArray>
              <idString>id-AT</idString>
            </objectIdentifier>
            <relativeAuthorization id="88">
              <role>CVCA</role>
              <authorization id="89">
                <storedBits id="90">
                  ...
                  <boolean>true</boolean>
                  ...
                  <boolean>false</boolean>
                  ...
                </storedBits>
              </authorization>
            </relativeAuthorization>
          </certificateHolderAuthorizationTemplate>
          <certificateEffective id="91">2012-05-10 22:00:00.0
            UTC</certificateEffective>
          <certificateExpiration id="92">2015-05-10 22:00:00.0
            UTC</certificateExpiration>
          <certificateExtensions id="93"/>
        </currentCertificate>
```

Ändern eines bestehenden Trust Points

Das vorhandene Trust Point Element `currentCertificate` soll nun so abgeändert werden, dass es den Trust Point aus DECVCAeIDCT00001.bin enthält passend zu DECVCAeIDCT00001.cvcert. Hierfür muss der Trust Point aus DECVCAeIDCT00001.bin zuerst z.B. mit Hilfe eines Hex- Editors in eine hexadezimale Repräsentation überführt werden.
Der hexadezimal kodierte Trust Point sieht aus wie folgt:

```
"308201333081EC06072A8648CE3D02013081E0020101302C06072A8648CE3D0101022100A9FB57D
BA1EEA9BC3E660A909D838D726E3BF623D52620282013481D1F6E5377304404207D5A0975FC2C305
7EEF67530417AFFE7FB8055C126DC5C6CE94A4B44F330B5D9042026DC5C6CE94A4B44F330B5D9BBD77
CBF958416295CF7E1CE6BCCDC18FF8C07B60441048BD2AEB9CB7E57CB2C4B482FFC81B7AFB9DE2
7E1E3BD23C23A4453BD9ACE3262547EF835C3DAC4FD97F8461A14611DC9C27745132DED8E545C1D5
4C72F046997022100A9FB57DBA1EEA9BC3E660A909D838D718C397AA3B561A6F7901E0E82974856A
7020101034200047EC402F29B04079C9D89A8F732AD09BABC6538849128C6539B6C0F17EA4B72F56
DD632376FA8CFD0E08E0DCA0F54802344F3137599121D20F9CADD358E5C3C7E"
```

Dieser Hex-String ersetzt nun unter dem Element `publicKey` den alten im Sub-Element `value`. Als nächstes müssen in den Sub-Elementen des Elements `certificateHolderReference` der countryCode, der holderMnemonic und die sequenceNumber angepasst werden.
Gleiches gilt bei Bedarf auch für das Gültigkeitsdatum.

```
 (<certificateEffective
      id="91">     2012-05-10
      22:00:00.0 UTC
</certificateEffective>)
```

und das Ablaufdatum

```
 (<certificateExpiration
      id="92">      2015-05-10
      22:00:00.0 UTC
</certificateExpiration>).
```

Nach erfolgreichem Abschluss aller Änderungen sieht die Personalisierung im Wesentlichen aus wie folgt:

```
<de.persosim.simulator.perso.Profile01 id="1">
      ...
  <mf id="12">
    <children id="13">
          ...
      <de.persosim.simulator.cardobjects.TrustPointCardObject id="77">
        <parent class="de.persosim.simulator.cardobjects.MasterFile"
          reference="12"/>
        <children id="78"/>
        <lifeCycleState>CREATION</lifeCycleState>
        <currentCertificate id="79">
          <certificateProfileIdentifier>0</certificateProfileIdentifier>
          <certificateAuthorityReference id="80">
            <countryCode>DE</countryCode>
            <holderMnemonic>TESTeID</holderMnemonic>
            <sequenceNumber>00004</sequenceNumber>
          </certificateAuthorityReference>
          <publicKeyOid id="81">
            <oidByteArray id="82">04007F00070202020203</oidByteArray>
            <idString>id-TA-ECDSA-SHA-256</idString>
          </publicKeyOid>
          <publicKey  id="83">
            <algorithm>EC</algorithm>
<value>308201333081EC06072A8648CE3D02013081E0020101302C06072A8648CE3D0101022100A
9FB57DBA1EEA9BC3E660A909D838D726E3BF623D52620282013481D1F6E5377304404207D5A0975F
C2C3057EEF67530417AFFE7FB8055C126DC5C6CE94A4B44F330B5D9042026DC5C6CE94A4B44F330B
5D9BBD77CBF958416295CF7E1CE6BCCDC18FF8C07B60441048BD2AEB9CB7E57CB2C4B482FFC81B7A
FB9DE27E1E3BD23C23A4453BD9ACE3262547EF835C3DAC4FD97F8461A14611DC9C27745132DED8E5
45C1D54C72F046997022100A9FB57DBA1EEA9BC3E660A909D838D718C397AA3B561A6F7901E0E829
74856A7020101034200047EC402F29B04079C9D89A8F732AD09BABC6538849128C6539B6C0F17EA4
B72F56DD632376FA8CFD0E08E0DCA0F54802344F3137599121D20F9CADD358E5C3C7E</value>
          </publicKey>
          <certificateHolderReference id="84">
            <countryCode>DE</countryCode>
            <holderMnemonic>TESTeID</holderMnemonic>
            <sequenceNumber>00004</sequenceNumber>
          </certificateHolderReference>
          <certificateHolderAuthorizationTemplate id="85">
            <objectIdentifier id="86">
              <oidByteArray id="87">04007F000703010202</oidByteArray>
              <idString>id-AT</idString>
            </objectIdentifier>
            <relativeAuthorization id="88">
              <role>CVCA</role>
              <authorization id="89">
                <storedBits id="90">
                  ...
                  <boolean>true</boolean>
                  ...
                  <boolean>false</boolean>
                  ...
                </storedBits>
              </authorization>
            </relativeAuthorization>
```

```
          </certificateHolderAuthorizationTemplate>
          <certificateEffective id="91">2012-05-10 22:00:00.0
            UTC</certificateEffective>
          <certificateExpiration id="92">2015-05-10 22:00:00.0
            UTC</certificateExpiration>
          <certificateExtensions id="93"/>
        </currentCertificate>
        <identifier id="94">
          <terminalType>AT</terminalType>
        </identifier>
      </de.persosim.simulator.cardobjects.TrustPointCardObject>
            ...
    </children>
            ...
  </mf>
</de.persosim.simulator.perso.Profile01>
```

Hinzufügen eines weiteren Trust Points zu einem bestehenden Trust Anchor

Für den Fall, dass ein Trust Point nicht geändert, sondern neu bzw. zusätzlich hinzugefügt werden soll, kann man einen bestehenden Trust Point

```
<currentCertificate id="continuousId">
        ...
</currentCertificate>
```

bzw.

```
<previousCertificate id="continuousId">
        ...
</previousCertificate>
```

einfach in dieselbe Hierarchieebene kopieren und entsprechend der obigen Anleitung ändern.

Die Reihenfolge, in der die Trust Points auf die Karte aufgebracht wurden, wird über den Namen des Elements festgelegt, in dem der TrustPoint abgelegt wird. Als Namen stehen hierbei `currentCertificate` sowie `previousCertificate` zur Verfügung. Der zuletzt hinzugefügte Trust Point wird als Element `currentCertificate` abgelegt während ein vorausgehender Trust Point als Element `previousCertificate` abgelegt wird. Dabei gilt zu beachten, dass innerhalb eines Trust Anchors lediglich zwei Trust Points, d.h. jeweils genau ein `currentCertificate` sowie ein `previousCertificate` abgelegt werden können.

```
<de.persosim.simulator.cardobjects.TrustPointCardObject  id="77">
        <parent      class="de.persosim.simulator.cardobjects.MasterFile"
           reference="12"/>
        <children id="78"/>
        <lifeCycleState>CREATION</lifeCycleState>
          ...
        <currentCertificate id="continuousId">
          ...
        </currentCertificate>
          ...
        <previousCertificate id="continuousId">
          ...
        </previousCertificate>
          ...
        <identifier id="94">
          <terminalType>AT</terminalType>
        </identifier>
</de.persosim.simulator.cardobjects.TrustPointCardObject>
```

Um unbeabsichtigten Konflikten beim Unmarshalling zu entgehen, wird strengstens empfohlen rekursiv die `id="continuousId"` jedes einzelnen kopierten Tags und seiner Subtags auf laufende Werte zu setzen, die noch von keinem anderen Tag reserviert wurden. Es gilt dabei zu beachten, gültige Verweise (`reference="continuousId"`) auf externe Objekte unverändert zu belassen oder bei internen Objekten entsprechend der neuen Ids anzupassen.