Spezifikation
MOA ID
2007-08-02

Konvention

MOA ID 1.4

Öffentlicher Entwurf

Bezeichnung	Spezifikation Module für Online Applikationen - ID
Kurzbezeichnung	MOA ID 1.4
Version	1.4
Datum	2007-08-02
Dokumentenklasse	Konvention
Dokumentenstadium	Entwurf öffentlich
Kurzbeschreibung	Dieses Dokument spezifiziert ein Servermodul für die weitere Umsetzung der E-Government-Strategien. Das Basismodul ID dient zur sicheren Identifikation und Authentifikation von Benutzern mittels Bürgerkarte. Dabei wird die digitale Signatur zum sicheren Beweis der Authentizität verwendet. Das Design sieht vor, dass Online-Applikationen die Identifikation und Authentifikation an MOA ID auslagern können. MOA-ID beinhaltet ab Version 1.3 neben den MOA-ID **Standardfunktionalitäten** für den Einsatz in der öffentlichen Verwaltung zusätzlich Erweiterungsfunktionen für die Privatwirtschaft, den so genannten **MOA-WID Modus**. Der MOA-WID Modus ermöglicht eine Identifikation von Kunden/Bürgern in Anwendungen der Privatwirtschaft mittels *wirtschaftsbereichsspezifischem Personenkennzeichen* (wbPK, vgl. [E-GovG]). Ab Version 1.4 bietet MOA-ID die Möglichkeit einer *erweiterten* Infobox-Validierung, das heißt, es können neben der Personenbindung auch weitere ausgelesene Infoboxen über ein einheitliches Interface applikationsspezifisch validiert werden. Mit dem vorliegenden Dokument wird eine weitere Vereinheitlichung des E-Governments und sowie eine Vereinfachung der Umsetzung von modernen Online-Applikationen angestrebt.
Autoren	ARGE Spezifikation MOA; Ansprechpartner: Stabsstelle IKT-Strategie des Bundes, Bundeskanzleramt. *Email:* Rudolf.Schamberger@cio.gv.at Gregor.Karlinger@cio.gv.at Bundesministerium für Finanzen *Email:* Ludwig.Moser@bmf.gv.at
Arbeitsgruppe	ARGE Spezifikation MOA

Inhaltsverzeichnis

Dokumenten Information

Versionen

Version	Datum	Beschreibung	Autor
1.0	8.10.2002	Version 1.0	ARGE
1.1	30.06.2003	Version 1.1 • Funktionserweiterung MOA-ID Ergänzung um BKU Auswahl. • Editoriale Ergänzungen	ARGE
1.2	15.03.2004	Version 1.2 • Funktionserweiterungen MOA-ID • Umstellung auf Stammzahl und bPK	ARGE
1.3	22.07.2005	Version 1.3 • Funktionserweiterung auf wbPK	IAIK
1.4	06.03.2007	Version 1.4 • Erweiterte Infobox-Validierung	A-SIT

Bundeskanzleramt Republik Österreich

Bundesministerium für Finanzen

1 Anwendungsfälle

Die primäre Funktion von MOA-ID ist die sichere Identifikation und Authentifikation von Benutzern unter Anwendung des Konzepts Bürgerkarte. Dabei werden die Personenbindung und die Signaturfunktion der Bürgerkarte verwendet um den Benutzer sicher zu authentifizieren. MOA-ID überprüft die benutzerbezogenen Daten die aus der s.g. Personenbindung der Bürgerkarte stammen und kann diese bzw. Teile daraus in Folge beliebigen Onlineanwendungen zur Verfügung stellen. Als weitere wichtige Funktion berechnet MOA-ID seit der Version 1.2 das so genannte bereichsspezifische Personenkennzeichen (bPK) und stellt ab Version 1.3 der Wirtschaft das so genannte wirtschaftsbereichsspezifische Personenkennzeichen (wbPK) zur Verfügung.

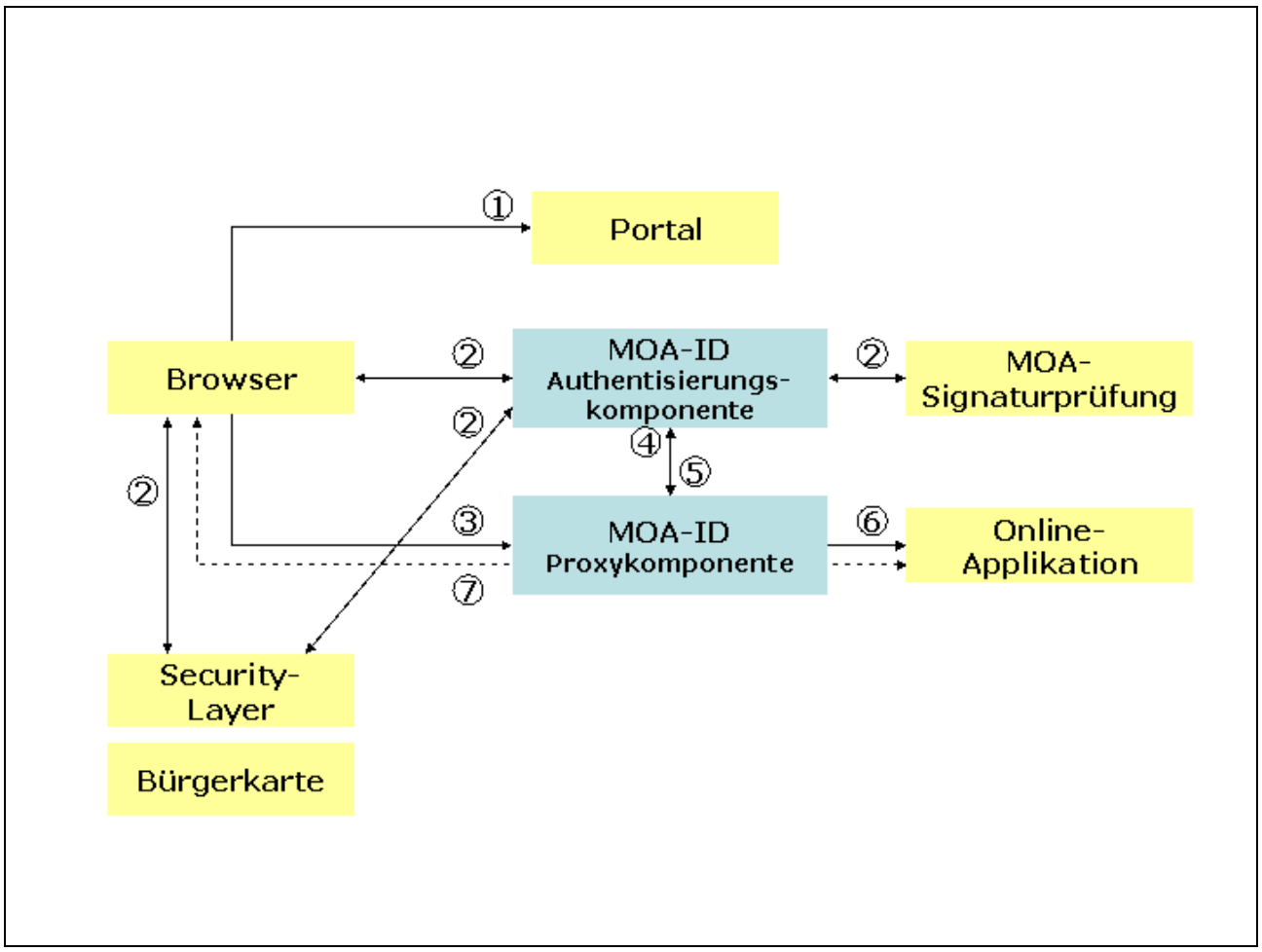
Der grundlegende Unterschied zur Standard-Version besteht darin, dass *die Errechnung eines wbPK aus der Stammzahl nicht beim Auftraggeber eines privaten Bereichs durchgeführt werden darf* (vgl. [E-GovG] §12(1).4), und deshalb an die Bürgerkartenumgebung ausgelagert werden muss.

Ab Version 1.4 bietet MOA-ID die Möglichkeit einer *erweiterten* Infobox-Validierung. Über die BKU selektiert der Anwender *vor* dem Anmeldevorgang welche Infoboxen zusätzlich ausgelesen werden sollen (*siehe Abschnitt 4.2.2 Abfrage der Personenbindung und ggf. weiterer Infoboxen*). Für die Validierung dieser Infoboxen stellt MOA-ID ein Interface zur Verfügung, das es ermöglicht, die entsprechenden `InfoboxReadResponses` gemäß den für den jeweiligen Typ festgeschriebenen Regeln und Vorschriften zu validieren, und in weiterer Folge relevante Daten in die Anmeldedaten aufzunehmen.

Die Anwendungsfälle, die MOA abdecken muss, sind in der folgenden Liste zusammengefasst:

- Auswahl einer Bürgerkartenumgebung (BKU-Auswahl)
- Kommunikation mit dem Browser des Benutzers und der Bürgerkartenumgebung
- Authentisierung eines Benutzers (Bürger oder Behördenmitarbeiter) mittels Bürgerkarte und Security-Layer
- Berechnung des bereichsspezifischen Personenkennzeichens (bPK).
  (Die Berechnung des bPK wird im MOA-WID-Modus nicht durchgeführt, da an Stelle des bPK das von der Bürgerkartenumgebung übermittelte wirtschaftsbereichsspezifische Personenkennzeichen (wbPK) verwendet wird).
- Fehlerhandling
- Konfiguration von MOA-ID
- Weitergabe der Anmeldedaten an nachfolgende Online-Anwendungen.

2 Überblick



MOA-ID besteht aus zwei Komponenten, der Authentisierungskomponente und der Proxykomponente, die mit dem Browser des Benutzers und Online-Applikationen Schnittstellen aufweisen. Mit dem Security-Layer des Benutzers wird sowohl über den Browser als auch direkt kommuniziert.

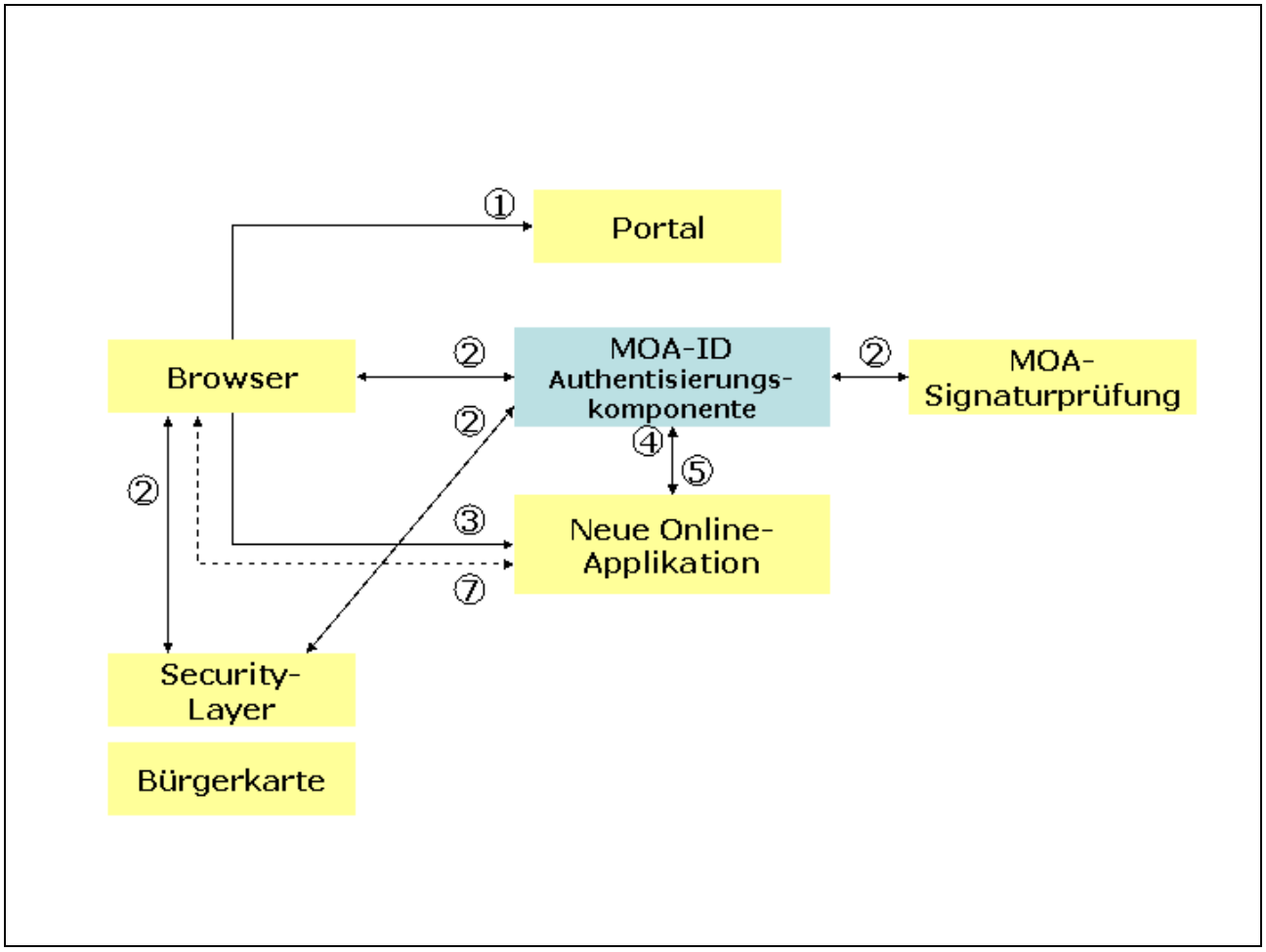
Die Authentisierungskomponente führt die eigentliche Authentisierung des Benutzers durch und übergibt der Proxykomponente die Anmeldedaten. Die Proxykomponente übernimmt die Anmeldedaten von der Authentisierungskomponente, führt die Anmeldung an der Online Applikation durch und schleust in der Folge Daten an die Online-Applikation und Daten an den Benutzer durch. Diese beiden Komponenten können auf unterschiedlichen Rechnern oder auf dem gleichen Rechner eingesetzt werden.

Die folgende Tabelle beschreibt einen typischen An- und Abmeldevorgang in mehreren Schritten (die Nummerierung der Schritte entspricht den Nummern in der oben gezeigten Grafik):

Schritt	Beschreibung
1.	Der Benutzer verbindet sich zu einem Web-Portal, über das die verfügbaren Online-Applikationen (OA) erreichbar sind. Jeder Link zu einer OA verweist auf die Authentisierungskomponente mit zwei Parametern: • URL der tatsächlich zu verwendeten Applikation: z.B. `https://app1.gv.at/Beispiel` • Geschäftsbereich: als String kodiert • Im MOA-WID-Modus wird nur die URL der Applikation, nicht aber der Geschäftsbereich verwendet. Ist der Geschäftsbereich trotzdem angegeben, so wird er ignoriert.
2.	Der Benutzer verbindet sich über HTTPS mit der Authentisierungskomponente, die die Authentisierung des Benutzers durchführt:
2.1.	MOA-ID-AUTH bietet dem Benutzer optional eine Auswahl von verfügbaren Bürgerkartenumgebungen an.
2.2.	Die Authentisierungskomponente erzeugt eine HTML-Seite zum Auslesen der Personen-Bindung mit folgenden Inhalten: • `<FORM>` mit Security-Layer `<InfoboxReadRequest>`-Kommando zum Auslesen der Personenbindung. Innerhalb des `<InfoboxReadRequest>` wird im MOA-WID-Modus die Stammzahl des privaten Unternehmens an die Bürgerkartenumgebung übergeben. Diese HTML-Seite wird an den Browser geschickt.
2.3.	Der Browser schickt das Kommando per HTTP POST an den Security-Layer. Der Security-Layer • liest die Personenbindung und ggf. weitere Infoboxen • berechnet das wbPK und ersetzt in der Personenbindung die Stammzahl der natürlichen Person durch das wbPK (nur MOA-WID-Modus) • sendet die Personenbindung und ggf. weitere Infobox-Inhalte über die angegebene `DataURL` (siehe [SecLayer1.1]) an die Authentisierungskomponente zurück.
2.4.	Die Authentisierungskomponente validiert die Personenbindung und ggf. weitere Infoxbox-Token (ab Version 1.4; bei früheren Versionen werden weitere Infobox-Token ignoriert) und sendet, als Antwort auf die Personenbindung, eine XML Antwortseite, die direkt das Kommando zum Signieren des von der Authentisierungskomponente generierten AUTH-Blocks enthält.
2.5.	Der Request wird vom Security-Layer verarbeitet (Siehe [SecLayer1.1-Bindung] Abschnitt 3.3.2 Punkt 5b). Die signierten Daten werden direkt and die `DataURL` zurückgesendet.
2.6.	Die Authentisierungskomponente überprüft den signierten AUTH-Block und legt für den Benutzer die Anmeldedaten (siehe *Abschnitt 4.5*) an.
2.7.	Ist der obige Authentisierungsvorgang erfolgreich, dann wird eine Redirect-Seite mit einem SAML-Artifact (siehe *Abschnitt 4.3*) erstellt und zum Browser gesendet.
3.	Der Browser führt das Redirect zur angegebenen URL (Proxykomponente von MOA-ID) durch. Als Parameter wird das eindeutige SAML-Artifact übergeben, mit dem die nachfolgende Applikation die Anmeldedaten von der Authentisierungskomponente abfragen kann.
4. 5.	Die nachfolgende Applikation (Proxykomponente) verwendet dieses eindeutige SAML-Artifact, um die Anmeldedaten via SOAP von der Authentisierungskomponente zu erhalten. Danach werden die Anmeldedaten in der

	Authentisierungskomponente gelöscht.
6.	Die Proxykomponente liest eine Konfigurationsdatei die beschreibt, wie die Anmeldedaten an die nachfolgende Applikation übergeben werden müssen, und meldet den Benutzer bei der Applikation an. Handelt es sich um eine Applikation aus der Wirtschaft, so ist ggf. eine ZMR-Anfrage zur vollständigen Authentifizierung erforderlich (vgl. [MOA-WID-OA]).
7.	In der Folge werden die Antworten der (stateless) OA an den Benutzer weitergeleitet und die Anfragen des Benutzers an die OA weitergeleitet.

Im Falle einer neu entwickelten Online-Applikation kann die Funktionalität und die Schnittstellen der Proxykomponente von der Online-Applikation übernommen werden.



Im Folgenden werden die Proxykomponente und die neu entwickelte Online-Applikation als *nachfolgende Applikation* bezeichnet.

3 Allgemeine Anforderungen

3.1 Unterstützte Plattformen

Es muss Java Runtime Environment ab Version 1.3 unterstützt werden.

3.2 TLS Authentisierung

Sämtliche Schnittstellen zwischen den Komponenten und Applikationen können mittels TLS [TLS] authentisiert werden:

Schnittstelle	TLS Authentisierung	Beschreibung
Browser - Authentisierungskomponente	Server	verpflichtend
Browser – nachfolgende Applikation	Server	verpflichtend
Authentisierungskomponente – MOA Signaturprüfung [MOA SP-SS]	Server, Client	konfigurierbar
nachfolgende Applikation - Authentisierungskomponente	Server, Client	konfigurierbar
Proxykomponente – Online-Applikation	Server, Client	konfigurierbar

3.3 Skalierbarkeit und Verfügbarkeit

MOA-ID muss skalierbar und auf einen 7 * 24h Betrieb ausgelegt sein.

3.4 Namespace

Alle in dieser Spezifikation definierten XML-Elemente sind dem Namespace http://reference.e-government.gv.at/namespace/moa/20020822# zugeordnet.

4 Authentisierungskomponente

Die Funktionsweise der Authentisierungskomponente orientiert sich an dem in [SAML-Binding] im Abschnitt 4.1.1 („Browser/Artifact Profile of SAML“) beschriebenen Vorgang, erweitert durch die Erstellung des vom Benutzer signierten AUTH-Blocks mit dem Security-Layer [SecLayer1.1].

4.1 Schnittstelle zur verweisenden Webseite

Die Authentisierungskomponente wird immer durch eine andere (verweisende) Webseite aufgerufen. Diese Webseite kann z.B. Teil eines Portals sein.

Der Aufruf der Authentisierungskomponente durch die verweisende Webseite (aufrufende Applikation) erfolgt durch das Aufrufen einer von der Authentisierungskomponente bereitgestellten URL durch den Browser des Benutzers.

4.1.1 StartAuthentication

Der Aufruf erfolgt durch einen Verweis der Form:

```
<a href="https://<moa-id-server-und-pfad>/StartAuthentication?
        Target=<geschäftsbereich>&
        OA=<oa-url>&
        Template=<template-url>">
```

Wobei:

- `<moa-id-server-und-pfad>` angibt, wo MOA-ID-AUTH installiert ist.
- `Target=<geschäftsbereich>` (siehe [GB], [GB-BK]) angibt, für welches Verfahren der Benutzer authentisiert werden soll. (Dieser Parameter entfällt im MOA-WID-Modus, bzw. wird ignoriert, falls er vorhanden ist).
- `OA=<oaurl>` mittels URL angibt, an welche Seite in der nachfolgenden Applikation der Benutzer nach erfolgreicher Authentisierung weitergeleitet werden muss. Die URL (bzw. ein eindeutiger Teil davon) dient außerdem als Schlüssel für die Auswahl der applikationsspezifischen Konfiguration in der Authentisierungskomponente (siehe Abschnitt 4.6 Konfiguration).
- `Template=<template-url>` eine optionale HTML-Vorlage für der Anmeldeseite von MOA-ID-AUTH angibt, über die der Bürger den Authentisierungsvorgang startet. Über diesen Parameter kann das Aussehen der Anmeldeseite an das Aussehen der Online-Applikation angepasst werden. Ab Version 1.3.1 ist es auch möglich, dieses Template über die Konfigurationsdatei zu laden. Details dazu können dem MOA-ID Installationshandbuch entnommen werden.

  Ein Template für die Anmeldeseite von MOA-ID-AUTH kann aus folgender Grundstruktur aufgebaut werden:

```
<form name="CustomizedForm" action="<BKU>" method="post">
 <input type="hidden"
        name="XMLRequest"
        value="<XMLRequest>"/>
 <input type="hidden"
        name="DataURL"
        value="<DataURL>"/>
 <input type="hidden"
        name="PushInfobox"
        value="<PushInfobox>"/>
 <input type="submit" value="Anmeldung mit Bürgerkarte"/>
</form>
<form name="CustomizedInfoForm"
 action="<BKU>"
 method="post">
 <input type="hidden"
        name="XMLRequest"
        value="<CertInfoXMLRequest>"/>
 <input type="hidden"
        name="DataURL"
        value="<CertInfoDataURL>"/>

Hier finden Sie weitere Informationen
zur Überprüfung der Zertifikate.<br/>
 <input type="submit" value="Weitere Info"/>
</form>
```

Innerhalb dieser `<form>`-Elemente können Texte, Beschriftungen und Styles modifiziert werden, und es können zusätzliche Elemente darin aufgenommen werden.

Die vorgegebene Grundstruktur ist aber in jedem Fall einzuhalten, und es müssen die speziellen Tags `<BKU>` (kommt 2x vor), `<XMLRequest>`, `<DataURL>`, `<CertInfoXMLRequest>` und `<CertInfoDataURL>` darin enthalten sein. Das Tag `<PushInfobox>` muss ab Version 1.4 vorhanden sein, wenn MOA-ID auch andere Infoboxen als die Personenbindung bearbeiten kann (*siehe Abschnitt 4.2.2 Abfrage der Personenbindung und ggf. weiterer Infoboxen*).

Die Weiterleitung an die nachfolgende Applikation erfolgt durch einen HTTP-Redirect (Status Code 302).

4.1.2 BKU-Auswahl

MOA-ID-AUTH bietet die Möglichkeit, die Bürgerkartenumgebung (BKU) auszuwählen, über die in weiterer Folge die Bürgerkarte ausgelesen wird. Der Aufruf erfolgt dann durch einen Verweis der Form:

```
<a href="https://<moa-id-server-und-pfad>/SelectBKU?
        Target=<geschäftsbereich>&
        OA=<oa-url>&
        Template=<template-url>&
        BKUSelectionTemplate=<bku-template-url>">
```

Wobei:

- Der *Target*-Parameter im MOA-WID-Modus nicht verwendet wird.
- BKUSelectionTemplate=<bku-template-url> eine optionale HTML-Vorlage für der BKU-Auswahlseite von MOA-ID-AUTH angibt. Über diesen Parameter kann das Aussehen der

BKU-Auswahlseite an das Aussehen der Online-Applikation angepasst werden. Ab Version 1.3.1 ist es auch möglich, dieses Template über die Konfigurationsdatei zu laden. Details dazu können dem MOA-ID Installationshandbuch entnommen werden.

Ein Template für die BKU-Auswahl von MOA-ID-AUTH kann aus folgender Grundstruktur aufgebaut werden:

```
<form name="CustomizedForm" method="post" action="<StartAuth>">
 <BKUSelect>
 <input type="submit" value="Auswählen"/>
</form>
```

Innerhalb dieser `<form>`-Elemente können Texte, Beschriftungen und Styles modifiziert werden, und es können zusätzliche Elemente darin aufgenommen werden.

Auch dabei ist die vorgegebene Grundstruktur einzuhalten, die speziellen Tags `<StartAuth>` und `<BKUSelect>` sind verpflichtend.

4.2 Schnittstelle zur Bürgerkarte

4.2.1 Information bezüglich der Wurzelzertifikate

Für die Sicherheit des Identifikationsmoduls ist es wesentlich, dass der Benutzer zuverlässig die Authentizität der Authentisierungskomponente überprüfen kann. Diese Authentizität wird durch die Verwendung des TLS-Protokolls sichergestellt. Da TLS-Implementierung in gängigen Web-Browsern viele Wurzelzertifikate als vertrauenswürdig vorinstalliert haben, ist es notwendig, dass der Benutzer die Richtigkeit der verwendeten Serverzertifikate manuell überprüft. Der folgende Ablauf soll sicherstellen, dass der Benutzer diese Überprüfung durchführt bzw. bewusst auf diese Überprüfung verzichtet.

Bevor die Personenbindung ausgelesen wird, bekommt der Benutzer eine Möglichkeit (mittels Info-Button) eine signierte Informationsseite von der Authentisierungskomponente zu laden, welche automatisch im Secure-Viewer des Security-Layers sicher angezeigt wird (sofern die SL-Implementierung dies unterstützt). Diese Informationsseite kann eine Beschreibung beinhalten, wie der Benutzer die Authentizität des Servers mittels TLS überprüfen kann, und wird vom CIO mit einem Zertifikat signiert, dessen Wurzelzertifikat in der SL-Implementierung als vertrauenswürdig installiert ist (z.B. ein A-Trust Zertifikat). Die Signatur und der Zertifikatspfad zum Wurzelzertifikat sind vom Benutzer zu überprüfen.

4.2.2 Abfrage der Personenbindung und ggf. weiterer Infoboxen

Diese Schnittstelle dient zum Abfragen der Personenbindung und ggf. weiterer Infoboxen (ab Version 1.4) vom Security-Layer (siehe *Abschnitt 2 Überblick, Schritt 2.2*).

Die Authentisierungskomponente erzeugt eine HTML-Seite die zumindest das folgende `<LINK>`-, `<FORM>`- und `<SCRIPT>`-Element enthält:

```
<LINK rel="stylesheet" type="text/css" href="<css-url>">

<FORM name="getIdentityLinkForm"
      action=<bku-url>
      method="POST">
    <INPUT type="hidden" name="XMLRequest"/>
    <INPUT type="hidden" name="DataURL"
           value=<verifyIdentityLinkServletID>?MOASessionID=< MOASessionID>/>
    <INPUT type="hidden" name="PushInfobox"
           value="<PushInfobox>"/>
```

```
    <INPUT type="submit" value="Auslesen der Personenbindung"/>
</FORM>

<A href='http://<Infopage>'> Wichtige Information zu Zertifikaten</A>
```

<bku-url> ist die URL, die bei der BKU-Auswahl returniert wird (bzw. http://localhost:3495/http-security-layer-request als Default-Wert).

<MOASessionID> ist die von MOA-ID generierte session id.

Das Feld XMLRequest muss folgenden Wert beinhalten:

```
<InfoboxReadRequest
  xmlns="http://www.buergerkarte.at/namespaces/securitylayer/20020225#">
  <InfoboxIdentifier>
    IdentityLink
  </InfoboxIdentifier>
  <BinaryFileParameters/>
</InfoboxReadRequest>
```

Während bei der MOA-ID Standard Version das Element BoxSpecificParameters grundsätzlich fehlen kann, wird es im MOA-WID-Modus immer verwendet, um die zur Berechnung des wbPK benötigte Stammzahl des Wirtschaftsunternehmens zu übergeben. Dabei hat dann das Kindelement IdentityLinkDomainIdentifier einen Wert, der sich aus dem Präfix urn:publicid:gv.at:wbpk, der Kennung für die Art der Stammzahl (z.b. FN für Firmenbuchnummer) und der Stammzahl des Wirtschaftsunternehmens (z.B. 468924i) zusammensetzt (jeweils getrennt durch das + Zeichen):

```
<InfoboxReadRequest
  xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
  <InfoboxIdentifier>
    IdentityLink
  </InfoboxIdentifier>
  <BinaryFileParameters ContentIsXMLEntity="true"/>
  <BoxSpecificParameters>
    <IdentityLinkDomainIdentifier>
      urn:publicid:gv.at:wbpk+FN+468924i
    </IdentityLinkDomainIdentifier>
  </BoxSpecificParameters>
</InfoboxReadRequest>
```

Das Feld DataURL enthält eine URL, welche einen eindeutigen, für jeden Auslesevorgang unterschiedlichen Identifier beinhaltet. Diese URL verweist auf ein Servlet, welches die Personenbindung entgegennimmt und in Folge eine Antwort darauf erzeugt (Beschreibung siehe *Abschnitt 4.2.3*). Weiters ist darauf zu achten, dass im MOA-WID-Modus der *Target*-Parameter in dieser URL nicht enthalten ist.

Das Feld PushInfobox (ab Version 1.4) enthält, durch Beistriche getrennt, die Identifier jener Infoboxen, die MOA-ID zusätzlich zur Personenbindung bearbeiten kann (siehe *Abschnitt 4.7.2 Erweiterte Infoboxüberprüfung)*. Die BKU wird nur Inhalte jener Infoboxen, die über dieses Feld identifiziert werden, an MOA-ID übermitteln. Hat der Anwender auch andere Infoboxen selektiert, so wird die BKU den Anmeldevorgang entweder abbrechen, oder den Anwender fragen, ob die Anmeldung nur mit den Infoboxen, die MOA-ID verarbeiten kann, fortgesetzt werden soll (vgl [SecLayerPI]). Ist MOA-ID so konfiguriert, dass bestimmte Infoboxen von der BKU übermittelt werden *müssen* (vgl. [MOA-ID Config]), so wird MOA-ID den Anmeldevorgang abbrechen, wenn zumindest eine dieser Infoboxen in der BKU-Antwort nicht enthalten ist.

Beispiel: Kann MOA-ID die Infoboxen für Vollmachten (Identifier="Mandates") und GDA-Token (Identifier="EHSPToken") überprüfen, so werden die Identifier dieser beiden Infoboxen über den PushInfobox Parameter folgendermaßen an die BKU übergeben:

PushInfobox=Mandates,EHSPToken

MOA-ID selbst entnimmt dabei diese Identifier aus Konfigurationsdatei, wo sie sowohl global als auch lokal je Onlineapplikation (überschreibt für die jeweilige Applikation den globalen Wert) gesetzt werden können (vgl. [MOA-ID Config]).

4.2.3 Signieren des AUTH-Block

Das im *Abschnitt 4.2.2 Abfrage der Personenbindung und ggf. weiterer Infoboxen* über die DataURL referenzierte Servlet nimmt die Personenbindung und ggf. weitere Infobox-Token entgegen.

Ist der <InfoboxReadRequest> erfolgreich, und kann die Validierung der Personenbindung und ggf. weiterer Infobox-Token (siehe *Abschnitt 4.7.2 Erweiterte Infoboxüberprüfung)* erfolgreich durchgeführt werden, so erzeugt das Servlet die XML-Antwortseite mit dem Befehl zum Signieren des AUTH-Blocks (siehe *Abschnitt 4.4*). (Dabei ist zu beachten, dass im MOA-WID-Modus die Validierung des Manifests (vgl. [PersBind]) fehlschlagen muss, da ja die Stammzahl der natürlichen Person in der Personenbindung durch das wbPK ersetzt wurde). Schlägt das Lesen der Infobox(en) oder das Validieren des Responses fehl, so wird eine HTML-Seite mit einer Fehlerbeschreibung erzeugt und der Authentisierungsvorgang abgebrochen. Die erzeugte XML- bzw. HTML-Seite wird an den Security-Layer retourniert, der je nach Content-Type der Antwort entsprechend reagiert. (siehe [SecLayer1.1-Bindung] Abschnitt 3.3.2 Punkt 5)

Folgende Schnittstelle definiert die XML-Seite, die dazu dient den AUTH-Block mit dem Security-Layer zu signieren (siehe *Abschnitt 2 Überblick, Schritt 2.4*).

XML-Antwortseite zum Signieren des AUTH-Blocks:

```
<CreateXMLSignatureRequest
  xmlns="http://www.buergerkarte.at/namespaces/securitylayer/20020831#">
  <KeyboxIdentifier>CertifiedKeypair</KeyboxIdentifier>
  <DataObjectInfo Structure="enveloping">
    <DataObject>
      <XMLContent>
        [der in Abschnitt 4.4 beschriebene AUTH-Block]
      </XMLContent>
    </DataObject>
    <TransformsInfo>
      [mögliche Transformationen]
    </TransformsInfo>
  </DataObjectInfo>
<SignatureInfo>
  [Spezifikation wo die Signatur eingebettet werden soll]
</SignatureInfo>
</CreateXMLSignatureRequest>
```

<TransformsInfo> enthält die Transformationen, die angewendet werden können, um die XML-Struktur des AUTH-Blocks in ein für den Benutzer verständliches Format zu konvertieren, das im Viewer des Security-Layers angezeigt werden kann. Es müssen sämtliche in *Abschnitt 4.4 AUTH-Block* spezifizierten Informationen dem Benutzer angezeigt werden. Dabei ist zu beachten, dass die Inhalte von ggf. durch die erweiterte Infobox-Validierung in den AUTH-Block aufgenommen <saml:Attribute>-Elementen in diese Transformationen einzubeziehen sind (ab Version 1.4). Diese Transformationen müssen für alle Security-Layer-Implementierungen, die unterstützt werden, konfiguriert werden können. Alle diese Transformationen müssen dem Security-Layer zur Verfügung gestellt werden, sodass dieser die geeignete Transformation auswählen kann.

Ist die Erstellung bzw. Prüfung der Signatur nicht erfolgreich, wird eine HTML-Seite mit einer Fehlerbeschreibung erzeugt und der Authentisierungsvorgang abgebrochen. Die erzeugte

HTML-Seite wird an den Browser retourniert.

Ist die Erstellung und Prüfung der Signatur erfolgreich, wird entsprechend der SAML Spezifikation ein Redirect, welches das SAMLartefact (siehe *Abschnitt 4.3*) enthält, vom Server über die Security-Layer-Implementierung an den Browser retourniert.

Als DataURL dürfen nur URLs spezifiziert werden, die die Authentisierungskomponente referenzieren.

4.3 Schnittstelle zur nachfolgenden Applikation

Die nachfolgende Applikation wird durch den HTTP-Redirect von der Authentisierungskomponente aufgerufen, und stellt die Authentisierung an der Applikation selbst sicher. Der HTTP-Request, mit dem die nachfolgende Applikation aufgerufen wird, hat die Form:

```
GET https://<oa-url>?Target=<geschäftsbereich>&SAMLArtifact=<saml-artifact>
```

Wobei:

- <oa-url> die URL angibt, an der die nachfolgende Applikation aufgerufen werden kann, um den Benutzer zu authentisieren. Sie ist identisch mit dem Wert des beim Aufruf der Authentisierungskomponente übergebenen Parameters OA.
- Target=<geschäftsbereich> den Geschäftsbereich angibt, für den der Benutzer authentisiert werden soll. Dieser Parameter wird im MOA-WID-Modus nicht verwendet.
- SAMLArtifact=<saml-artifact> das SAML-Artifact übergibt, das während des Authentisierungsvorganges mit der Authentisierungskomponente erzeugt wurde, und anhand dessen die nachfolgende Applikation die Identität des Benutzers verifizieren kann. Die Codierung des SAML-Artifact ist in [SAML-Binding], Abschnitt 4.1.1.8 beschrieben.

Die nachfolgende Applikation muss mit Hilfe des SAML-Artifact einen SOAP-Request an die Authentisierungskomponente zusammenstellen, wie in [SAML-Binding], Abschnitte 3 und 4.1.1.6 beschrieben. Bei erfolgreicher Rückmeldung der SOAP-Response, wie in Abschnitt 4.1.1.6 in [SAML-Binding] beschrieben, ist der Benutzer an der nachfolgenden Applikation angemeldet.

4.4 AUTH-Block

Der AUTH-Block ist eine Datenstruktur, die von der Authentisierungskomponente und vom Security-Layer mittels eines XSLT Stylesheets aufgebaut wird, und enthält folgende Informationen:

- Vorname, Nachname aus Personenbindung
- URL der Authentisierungskomponente
- URL und
    - Geschäftsbereich der nachfolgenden Applikation (Standardfunktionalität)
    - wbPK und Stammzahl des Wirtschaftsunternehmens (MOA-WID-Modus)
- Aktuelle Zeit
- optional: Daten aus weiteren Infoboxen, die vom Bürgerkartenbenutzer signiert werden müssen (ab Version 1.4)

Die Personenbindung ist nicht im AUTH-Block enthalten, weil:

- die Stammzahl der natürlichen Person nicht gespeichert werden darf

- die Daten der Personenbindung (öffentliche Schlüssel, Stammzahl) für den Benutzer wenig aussagekräftig sind und daher eine Anzeige im Secure Viewer des Security-Layer nicht sinnvoll ist
- die Daten der Personenbindung würden drei mal zwischen Security-Layer und Authentisierungskomponente übertragen werden

Das wbPK ist im AUTH-Block enthalten (nur MOA-WID-Modus), weil:

- das Speichern des wbPK kein Problem darstellt
- das wbPK damit vom Bürgerkarten-Benutzer signiert wird (das von der Bürgerkartenumgebung übermittelte wbPK ersetzt in der signierten Personenbindung die ursprünglich enthaltene Stammzahl der natürlichen Person, wodurch die erweiterte Validierung (Validierung des Manifests) fehlschlägt. Durch die Aufnahme des wbPK in den AUTH-Block bestätigt jedoch der Benutzer die Echtheit des wbPK.)

Der AUTH-Block wird als SAML-Assertion [SAML-Assertions] dargestellt und vom Benutzer signiert.

Das <saml:Assertion>-Element muss folgende verpflichtende Attribute enthalten:

Name	Beschreibung
MajorVersion	Muss den Wert 1 haben
MinorVersion	Muss den Wert 0 haben
AssertionID	Die Assertion-ID wird in der Folge nicht verwendet und kann daher auf einen beliebigen Wert gesetzt werden.
Issuer	Der Vorname und Nachname des Benutzers durch ein Leerzeichen getrennt (der Vorname wird aus dem <GivenName>-Element, der Nachname aus dem <FamilyName>-Element der Personenbindung entnommen.
IssueInstant	Zeitpunkt der Erstellung der Assertion

Das Element enthält weiters genau ein <saml:AttributeStatement>-Element und genau ein <ds:Signature>-Element.

4.4.1 <saml:AttributeStatement>

Das <saml:AttributeStatement>-Element enthält genau ein <saml:Subject>-Element und mehrere <saml:Attribute>-Elemente.

4.4.2 <saml:Subject>

<saml:Subject> enthält ein <saml:NameIdentifier> Element als Inhalt.

4.4.3 <saml:NameIdentifier>

Enthält die URL der Authentisierungskomponente als Inhalt.

4.4.4 <saml:Attribute>

<saml:Attribute> kommt mindestens zwei Mal vor, um den Geschäftsbereich (Standardfunktionalität) bzw. das wbPK (MOA-WID-Modus) und die URL der nachfolgenden Applikation darzustellen. Weitere <saml:Attribute> Elemente mit für die Signatur relevanten

Daten aus anderen Infoboxen als die der Personenbindung können vorhanden sein (ab Version 1.4).

Jedes <saml:Attribute> enthält genau ein <saml:AttributeValue>.

4.4.4.1 Geschäftsbereich bzw. wbPK

Dieses <saml:Attribute> hat, abhängig vom Kontext, in dem es verwendet wird, eine andere Bedeutung. Während es bei der Standardfunktionalität den Geschäftsbereich der nachfolgenden Applikation abbildet, stellt es im MOA-WID-Modus das wirtschaftsbereichsspezifische Personenkennzeichen (wbPK) dar.

4.4.4.1.1 Geschäftsbereich (Standardfunktionalität)

<saml:Attribute> für den Geschäftsbereich hat folgende Attribute:

Name	Beschreibung
AttributeName	Geschäftsbereich
AttributeNamespace	http://reference.e-government.gv.at/namespace/moa/20020822#

<saml:Attribute> für den Geschäftsbereich enthält genau ein <saml:AttributeValue> mit dem Geschäftsbereich als xs:token. Der Geschäftsbereich wurde vorher als Parameter dem Servlet übergeben.

4.4.4.1.2 Wirtschaftsbereichsspezifisches Personenkennzeichen (MOA-WID-Modus)

<saml:Attribute> für das wirtschaftsbereichsspezifische Personenkennzeichen (wbPK) hat folgende Attribute:

Name	Beschreibung
AttributeName	wbPK
AttributeNamespace	http://reference.e-government.gv.at/namespace/moa/20020822#

<saml:Attribute> für das **wbPK** enthält genau ein <saml:AttributeValue>, das als einziges Kindelement das <pr:Identification> Element aus der von der Bürgerkartenumgebung übermittelten Personenbindung aufnimmt (vgl. [PersBind]). Dieses Element enthält das **wbPK**, da die Stammzahl der natürlichen Person von der Bürgerkartenumgebung durch das wirtschaftsbereichsspezifische Personenkennzeichen ersetzt wurde. Damit ergibt sich für dieses <saml:Attribute> die in folgendem Beispiel dargestellte Struktur:

```
<saml:Attribute AttributeName="wbPK" AttributeNamespace="http://reference...">
  <saml:AttributeValue>
    <pr:Identification>
      <pr:Value>q4Tt5WKrnqFJGe5pDLDkiaDEi/8=</pr:Value>
      <pr:Type>urn:publicid:gv.at:wbpk+FN+468924i</pr:Type>
    </pr:Identification>
  </saml:AttributeValue>
</saml:Attribute>
```

4.4.4.2 URL der nachfolgenden Applikation

<saml:Attribute> hat folgende Attribute:

Name	Beschreibung
AttributeName	OA
AttributeNamespace	http://reference.e-government.gv.at/namespace/moa/20020822#

<saml:Attribute> für die URL der nachfolgenden Applikation enthält genau ein <saml:AttributeValue> mit der <oaurl> als xs:anyURI. Die URL wurde vorher als Parameter dem Servlet übergeben.

4.4.5 <ds:Signature>

Die XML Signatur der <saml:Assertion> wird gemäß Security-Layer Version 1.1 [SecLayer1.1] erstellt.

Die Signatur enthält ein <dsig:Reference> Element, das wie folgt auszuführen ist:

```
<dsig:Reference URI="">
  <dsig:Transforms>
    <dsig:Transform
      Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
      [XSLT Transformationen]
  </dsig:Transforms>
  <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
  <dsig:DigestValue> ..... </dsig:DigestValue>
</dsig:Reference>
```

Weiters enthält die Signatur ein <ds:KeyInfo> Element, welches ausreichend Information enthalten muss, dass eine automatische Validierung der Signatur möglich ist. Jedenfalls muss das X509 Signaturzertifikat eingebunden werden.

4.5 Anmeldedaten

Die Anmeldedaten werden in Form einer SAML-Assertion dargestellt, die, basierend auf dem signierten AUTH-Blocks und der vorher übersendeten zugehörigen Personenbindung, von der Authentisierungskomponente der nachfolgenden Applikation zur Verfügung gestellt wird. Die SAML-Assertion wird nicht signiert und enthält folgende Informationen:

- bPK (Standardfunktionalität) bzw. wbPK (MOA-WID-Modus)
- signierter AUTH-Block (konfigurierbar)
- signierte Personenbindung
  (konfigurierbar: keine Personenbindung, Personenbindung mit Stammzahl, Personenbindung ohne Stammzahl)
- PersonData Struktur (konfigurierbar: mit oder ohne Stammzahl) (zu beachten ist hier, dass die Stammzahl im MOA-WID-Modus durch das wbPK ersetzt wurde)
- Zertifikatstype (qualifiziertes Zertifikat oder einfaches Zertifikat)
- bkuURL (Die URL der Bürgerkartenumgebung)
- Information zur Behörde (optional)
- Zertifikat (konfigurierbar)

- Daten aus weiteren Infoboxen (optional ab Version 1.4)

Der Aufbau der SAML-Assertion für die Anmeldedaten erfolgt analog zum Aufbau des AUTH-Blocks (*Abschnitt 4.4*)

Das <saml:Assertion>-Element muss folgende verpflichtende Attribute enthalten:

Name	Beschreibung
MajorVersion	muss den Wert 1 haben
MinorVersion	muss den Wert 0 haben
AssertionID	Die Assertion-ID wird in der Folge nicht verwendet und kann daher auf einen eindeutigen Wert gesetzt werden.
Issuer	URL der Authentisierungskomponente
IssueInstant	Zeitpunkt der Erstellung der Assertion

Das Element enthält weiters genau ein <saml:AttributeStatement>-Element.

4.5.1 <saml:AttributeStatement>

Das <saml:AttributeStatement>-Element enthält genau ein <saml:Subject>-Element und mehrere <saml:Attribute>-Elemente.

4.5.2 <saml:Subject>

<saml:Subject> enthält ein <saml:NameIdentifier> Element.

4.5.3 <saml:NameIdentifier>

Das Element muss folgendes Attribut enthalten:

Name	Beschreibung
NameQualifier	Beschreibt die Domäne des Namens, hat im Standardmodus den Wert urn:publicid:gv.at:cdid+bpk, und entspricht im MOA-WID Modus dem als IdentityLinkDomainIdentifier an die Bürgerkartenumgebung übergebenen String (z.B.: urn:publicid:gv.at:wbpk+FN+468924i). (Vgl. Abschnitt 4.2.2 *Abfrage der Personenbindung und ggf. weiterer Infoboxen*)

Als Inhalt enthält <saml:NameIdentifier> das bPK (Standardfunktionalität) bzw. das wbPK (MOA-WID_Modus) in Base64 kodierter Darstellung.

4.5.4 <saml:SubjectConfirmation>

Ist konfiguriert, dass AUTH-Block und/oder Personenbindung mitgegeben werden sollen, werden diese im Element <saml:SubjectConfirmation> eingebunden.

Als <saml:ConfirmationMethod> wird
http://reference.e-government.gv.at/names/moa/20020822#cm benutzt.

<saml:SubjectConfirmationData> enthält den AUTH-Block und/oder die Personenbindung (in dieser Reihenfolge). Die Stammzahl der Personenbindung wird je nach Konfiguration ausgeblendet oder mitgegeben.

4.5.5 <saml:Attribute>

<saml:Attribute> kommt mehrmals vor um die oben erwähnten Informationen darzustellen:

Jedes <saml:Attribute> enthält genau ein <saml:AttributeValue>.

4.5.5.1 PersonData

Das Attribut für die Personendaten ist immer vorhanden und enthält abhängig von der Konfiguration die Stammzahl (siehe *Abschnitt 4.6*).

<saml:Attribute> für PersonData hat folgende Attribute:

Name	Beschreibung
AttributeName	PersonData
AttributeNamespace	http://reference.e-government.gv.at/namespace/persondata/20020228#

<saml:AttributeValue> enthält genau die PersonData-Struktur (identisch zu der in der Personenbindung strukturiert). Diese PersonData-Struktur enthält: Vorname, Nachname, Geburtsdatum, Stammzahl (optional).

4.5.5.2 Zertifikatstyp

Dieses Attribut beschreibt den Typ des Zertifikats, mit dessen zugehörigen privaten Schlüssel der AUTH-Block vom Benutzer signiert wurde. <saml:Attribute> für den Zertifikatstyp hat folgende Attribute:

Name	Beschreibung
AttributeName	isQualifiedCertificate
AttributeNamespace	http://reference.e-government.gv.at/namespace/moa/20020822#

<saml:Attribute> für den Zertifikatstyp enthält genau ein <saml:AttributeValue> mit folgendem Inhalt:

true	wenn das Zertifikat ein qualifiziertes Zertifikat ist
false	wenn das Zertifikat ein einfaches Zertifikat ist

4.5.5.3 bkuURL

Dieses Attribut enthält die URL der verwendeten Bürgerkartenumgebung. <saml:Attribute> für die bkuURL hat folgendes Attribute:

Name	Beschreibung
AttributeName	bkuURL
AttributeNamespace	http://reference.e-government.gv.at/namespace/moa/20020822#

<saml:Attribute> für die bkuURL enthält genau ein <saml:AttributeValue> mit der URL der Bügerkartenumgebung.

4.5.5.4 Behördenzertifikat

Dieses Attribut beschreibt ob bei der Signatur des AUTH-Block ein Behördenzertifikat verwendet wurde. Ein Behördenzertifikat ist ein Zertifikat mit der Erweiterung *Behördeneigenschaft* (siehe [OID]).

Ein <saml:Attribute> für die Information ob ein Behördenzertifikat vorliegt ist nur dann in den Anmeldedaten vorhanden, wenn zur Signaturerstellung ein Behördenzertifikat verwendet wurde.

Das <saml:Attribute> hat folgende Attribute:

Name	Beschreibung
AttributeName	isPublicAuthority
AttributeNamespace	urn:oid:1.2.40.0.10.1.1.1 (siehe [RFC3061])

Das <saml:Attribute> für die Information ob ein Behördenzertifikat vorliegt enthält genau ein <saml:AttributeValue> mit folgendem Inhalt:

True	wenn das Zertifikat ein Behördenzertifikat ist, aber kein Behördenkennzeichen im Zertifikat verfügbar ist
Behördenbezeichnung	wenn das Zertifikat ein Behördenzertifikat ist und ein Behördenkennzeichen im Zertifikat verfügbar ist

4.5.5.5 Zertifikat

Dieses optionale Attribut enthält das Zertifikat des Signators. <saml:Attribute> für das Zertifikat hat folgende Attribute:

Name	Beschreibung
AttributeName	SignerCertificate
AttributeNamespace	http://reference.e-government.gv.at/namespace/moa/20020822#

<saml:Attribute> für das Zertifikat enthält genau ein <saml:AttributeValue> mit dem Zertifikat des Signators in Base64 kodierter Darstellung.

4.5.5.6 Daten aus weiteren Infoboxen (ab Version 1.4)

Für etwaige Daten aus weiteren Infoboxen werden weitere <saml:Attribute> Elemente in die Anmeldedaten aufgenommen, wobei jedes Datum auf ein eigenes <saml:Attribute> abgebildet wird

4.6 Konfiguration

Folgende Daten müssen für die Kommunikation mit allen zu unterstützenden Security-Layer Implementierungen konfiguriert werden können:

- Transformationen für unterschiedliche Datenformate für die sichere Anzeige des AUTH-Blocks.
- Infoboxen, die mittels Pushverfahren (vgl. [SecLayerPI]) von der BKU übermittelt werden. MOA-ID muss dazu der BKU mitteilen, welche Infoboxen bearbeitet werden

können (siehe *Abschnitt 4.2.2 Abfrage der Personenbindung und ggf. weiterer Infoboxen*).

Folgende Daten müssen für die Kommunikation mit MOA-Signaturprüfung konfigurierbar sein:

- Konfiguration ob die Kommunikation mit MOA-Signaturprüfung via SOAP- oder via API-Aufrufe erfolgen soll.
- Angabe, ob die Schnittstelle zur MOA-Signaturprüfung mit TLS-Client- und/oder TLS-Server-Authentisierung geschützt werden muss.
    - verwendetes TLS-Client-Zertifikat der Authentisierungskomponente
    - akzeptierte TLS-Server-Zertifikate der MOA-Signaturprüfung
    - vertrauenswürdige Wurzelzertifikate für akzeptierte TLS-Server-Zertifikate der MOA-Signaturprüfung
- `TrustProfileID` und `VerifyTransformsInfoID` für die Überprüfung der Signatur der Personenbindung
- `TrustProfileID` und `VerifyTransformsInfoID` für die Überprüfung der Signatur des AUTH-Blocks

  `TrustProfileIDs` für die Überprüfung der Signaturen aus weiteren Infoboxen (ab Version 1.4). Pro Infobox (mit Ausnahme der Personenbindung, für die bereits ein eigenes Trustprofil besteht) muss eine `TrustProfileID` gesetzt werden können, wobei ein und dasselbe Profil für mehrere Infoboxen verwendet werden kann. Es muss möglich sein, die TrustProfileID sowohl global für jede weitere Infobox, als auch lokal je Online-Applikation zu setzen (ein lokaler Wert überschreibt dabei für die jeweilige Online-Applikation die global für die betroffene Infobox gesetzte ID) (vgl. [MOA-ID Config]).

  Weiters ist ein Default-Trustprofil vorzusehen, das zur Validierung einer Infobox herangezogen wird, falls für diese Infobox kein eigenes Trustprofil definiert wurde.

  Das Profil für die Überprüfung der Signatur der `Personenbindung` darf *nicht* zur Validierung anderer Infoboxen herangezogen werden. Weiters wird empfohlen, das Profil zur Validierung der Signatur des `AUTH-Blocks` nur für weitere Infoboxen zu verwenden, wenn die entsprechende Zertifikatskette zu einem bereits in diesem Profil enthaltenen Trust-Anchor führt.
- Ein Profil zur Verifikation der Transformationsketten ist für eine erweiterte Infobox-Validierung nicht vorgesehen.

Folgende Daten müssen für jede nachfolgende Applikation in der Authentisierungskomponente konfigurierbar sein:

- Angabe, ob die Stammzahl in der Personenbindung und in der PersonData-Struktur an die nachfolgende Applikation übergeben wird oder nicht
- Angabe, ob der AUTH-Block an die nachfolgende Applikation übergeben wird oder nicht.
- Angabe, ob die Personenbindung an die nachfolgende Applikation übergeben wird oder nicht.
- Angabe, ob die Schnittstelle zur nachfolgenden Applikation mit TLS-Client- und/oder TLS-Server-Authentisierung geschützt werden muss.
    - verwendetes TLS-Server-Zertifikat
    - akzeptierte TLS-Client-Zertifikate der nachfolgenden Applikation
    - vertrauenswürdige Wurzelzertifikate für akzeptierte TLS-Client-Zertifikate der nachfolgenden Applikation

Mehrere nachfolgende Applikationen müssen innerhalb derselben

Authentisierungskomponente konfiguriert werden können. Die im Aufruf übergebene URL der nachfolgenden Applikation (siehe Abschnitt 4.1 *Schnittstelle zur verweisenden Webseite*) wird zur Auswahl der Konfiguration herangezogen. Ein eindeutiges Präfix der URL der nachfolgenden Applikation muss deshalb als Schlüssel verwendet werden. Die URL muss zumindest aus `https://` und einem gültigen Domänen-Namen bestehen.

4.7 Funktionsbeschreibung

4.7.1 Überprüfung der Signatur der Personenbindung

Die Überprüfung der Signatur der Personenbindung erfolgt gemäß der Spezifikation für die Personenbindung [PersBind]. Die Überprüfung wird mittels des Kommandos `<VerifyXMLSignatureRequest>` des MOA-Modul für Signatur-Prüfung mit folgenden verpflichtenden Eingangs-Parametern durchgeführt:

`<VerifySignatureInfo>`	enthält die signierte Personenbindung
`<SignatureManifestCheckParams>`	Das Attribut `ReturnReferenceInputData` muss auf `false` gesetzt sein
`<TrustProfileID>`	Eine Referenz auf ein Profil von Wurzelzertifikaten

Anschließend wird überprüft, ob die Personenbindung mit einem für die Personenbindung zugelassenem Zertifikat signiert wurde.

Zu beachten ist hier, dass im MOA-WID-Modus die erweiterte Validierung (Validierung des Manifests) (vgl. [PersBind]) fehlschlägt.

4.7.2 Erweiterte Infoboxüberprüfung

Ab Version 1.4 bietet MOA-ID die Möglichkeit, neben der Personenbindung auch andere Infoboxen aus der BKU zu validieren. Da der Inhalt dieser Infoboxen nicht a priori bekannt ist, stellt MOA-ID ein generisches Interface zur Verfügung, das von einer implementierenden Klasse für die Validierung des jeweiligen Infobox-Tokens verwendet werden kann:

```
package at.gv.egovernment.moa.id.auth.validator;

import at.gv.egovernment.moa.id.auth.data.InfoboxValidationResult;
import at.gv.egovernment.moa.id.auth.data.InfoboxValidatorParams;

/**
 * Validates an InfoboxReadResponse.
 * An implementing class has to validate the content of the InfoboxReadResponse
 * according to the type specific rules and guidelines of the underlying
 * application.
 */
public interface InfoboxValidator {

  /**
   * This method validates an InfoboxReadResponse.
   * The method validates the content of the passed <code>infoboxReadResponse</code>
   * according to the type specific rules and guidelines of the underlying
   * application.
   *
   * @param params {@link at.gv.egovernment.moa.id.auth.data.InfoboxValidatorParams
   *                Parameters} needed by the validator.
   *
   * @return <code>True</code> if validation succeeds,
   *         otherwise <code>false</code>.
   *
```

```
    * @throws ValidateException  If an error occurs on validating the
    *                            InfoboxReadResponse.
    */
   public InfoboxValidationResult validate (InfoboxValidatorParams params)
     throws ValidateException;

}
```

Das Interface enthält als einzige Methode die Methode validate(InfoboxValidatorParams params)-Methode, die von MOA-ID aufgerufen wird, um den Inhalt eines InfoboxReadRequests zu validieren. Dieser Methode übergibt MOA-ID über ein Objekt vom Typ at.gv.egovernment.moa.id.auth.data.InfoboxValidatorParams

die vom jeweiligen InfoboxValidator benötigten Parameter:

```
package at.gv.egovernment.moa.id.auth.data;

import java.security.PublicKey;
import java.util.List;

import org.w3c.dom.Element;

/**
 * Parameters for validating an infobox.
 *
 * This interface is used by MOA-ID to provide parameters to an
 * {link at.gv.egovernment.moa.id.auth.validator.InfoboxValidator
 * InfoboxValidator}.
 *
 * @author Harald Bratko
 */
public interface InfoboxValidatorParams {

  /**
   * Returns a list of
   * {@link at.gv.egovernment.moa.id.auth.data.InfoboxToken InfoboxToken}
   * objects. The first token in this list is the one to be validated. Each further
   * token maybe needed to validate this first token.
   *
   * @return A list of
   * {@link at.gv.egovernment.moa.id.auth.data.InfoboxToken InfoboxToken}
   * objects.
   */
  public List getInfoboxTokenList();

  /**
   * Returns the ID of the trust profile to be used for validating
   * certificates. Maybe ignored by a validator, if no certificates
   * has to be validated.
   *
   * @return The ID of a trust profile.
   */
  public String getTrustProfileID();

  /**
   * Returns schema location URIs that may be needed by the
   * validator to parse infobox tokens.
   * Each entry in the list is a {@link Schema} specifying the location
   * of an XML schema.
   *
   * @return A list of {@link Schema} objects each of them specifying the
   *         location of an XML schema.
   */
  public List getSchemaLocations();
```

```java
/**
 * Returns the URL of the BKU.
 * Maybe needed by a validator.
 *
 * @return The url of the BKU.
 */
public String getBkuURL();

/**
 * Returns the target parameter.
 * <code>null</code> in the case of a business service.
 *
 * @return The target parameter.
 */
public String getTarget();

/**
 * Returns <code>true</code> if the application is a business
 * service, otherwise <code>false</code>. This may be useful
 * for the validating application.
 *
 * @return <code>True</code> if the application is a business
 * service, otherwise <code>false</code>
 */
public boolean getBusinessApplication();

/**
 * Returns the family name from the identity link.
 * Maybe needed by a validator.
 *
 * @return The family name from the identity link.
 */
public String getFamilyName();

/**
 * Returns the given name from the identity link.
 * Maybe needed by a validator.
 *
 * @return The given name from the identity link.
 */
public String getGivenName();

/**
 * The date of birth from the identity link.
 * The method returns the value of the <pr:DateOfBirth>
 * element from the identity link.
 * Maybe needed by a validator.
 *
 * @return The date of birth from the identity link.
 */
public String getDateOfBirth();

/**
 * Returns he identification value from the identity
 * link. This may be the <code>Stammzahl</code>
 * in the case of a public application or the
 * <code>wbPK</code> in the case of a business
 * application. This parameter is only returned
 * if specified within the config file.
 *
 * @return The identification value from the identity link.
 */
public String getIdentificationValue();

/**
 * Returns the type of the identification value
 * from the identity link. This may be
```

```
     * especially of interest for business
     * applications.
     *
     * @return The type of the identification value
     * from the identity link.
     */
    public String getIdentificationType();

    /**
     * Returns the public keys from the identity link.
     * Maybe needed by the application.
     *
     * @return PublicKey[] The public keys from the
     *                     identity link.
     */
    public PublicKey[] getPublicKeys();

    /**
     * Returns the identity link.
     * A validator may need other data from the identity link, than
     * family name, given name, date of birth and identification value.
     * The identity link element is only returned if specified within the
     * config file.
     *
     * @return The identity link.
     */
    public Element getIdentityLink();

    /**
     * Indicates whether source pins (<code>Stammzahl</code>en) should be hidden or not.
     * If an online application lying behind MOA-ID is not allowed to get source pins
     * (<code>Stammzahl</code>en), any source pins within <code>SAML attributes</code>
     * returned by the validator must suppressed:<br>
     * If the parameter <code>getHideStammzahl</code> is <code>true</code>, then the
     * validator <b>MUST</b> hide (replace by an empty string) any source pin
     * (<code>Stammzahl</code>) that may be included in a <code>SAML attribute</code>
     * returned by the validator.
     *
     * @return <code>true</code> if source pins (<code>Stammzahl</code>en) must be hidden,
     *         otherwise <code>false</code>.
     */
    public boolean getHideStammzahl();

    /**
     * Returns application specific parameters.
     * Each child element of this element contains
     * a validating application specific parameter. The
     * element is passed as specified within the config
     * file and its up to the implementing class to
     * parse and interprete its children.
     *
     * @return Application specific parameters.
     */
    public Element getApplicationSpecificParams();

}
```

Erläuterung der Methoden:

- `public List getInfoboxTokenList()`: gibt eine Liste von Objekten vom Typ `at.gv.egovernment.moa.id.auth.data.InfoboxToken` zurück. Das erste Token in dieser Liste ist das Token, das von der Prüfapplikation validiert werden muss. Wenn weitere Token vorhanden sind, dann können diese unter Umständen für die Validierung des ersten Tokens benötigt werden (zum Beispiel Validierung einer Token-Kette).

```
package at.gv.egovernment.moa.id.auth.data;
```

```
import org.w3c.dom.Element;

/**
 * Contains an infobox token included in an <code>InfoboxReadResponse</code>.
 * Depending on whether the token is extracted from an <code>XMLContent</code> or
 * a <code>Base64Content</code> it is either returned as DOM element
 * ({@link #getXMLToken()}) or base64 encoded string ({@link #getBase64Token()}).
 *
 * @author Harald Bratko
 */
public interface InfoboxToken {

  /**
   * The key of the corresponding infobox.
   * This is the value of <code>Key</code> attribute of the
   * <code>&lt;Pair&gt;</code> child element in an
   * <code>&lt;AssocArrayData&gt;</code> content of an InfoboxReadResponse.
   * Maybe <code>null</code> if the InfoboxReadResponse conatains
   * <code>BinaryFileData</code>.
   *
   *
   * @return The key identifier of the corresponding infobox.
   */
  public String getKey();

  /**
   * Specifies if this token is the first token (e.g in an array of tokens)
   * included in an <code>InfoboxReadResponse</code>. If <code>true</code>
   *  this token is the token to be validated by a corresponding
   * {@link at.gv.egovernment.moa.id.auth.validator.InfoboxValidator}
   * If <code>false</code> this token maybe needed to validate the primary token.
   *
   * @return <code>True</code> if this token is the first token.
   */
  public boolean isPrimary();

  /**
   * Returns the infobox token.
   * Maybe <code>null</code> if the token is returned by method
   * {@link #getBase64Token()} asa  base64 encoded string.
   * <br>
   * Note that this token is <code><i><b>not</b></i></code> validated against the
   * application specific schema (e.g. Mandates schema in the mandates context).
   * Thus the validating application <code><i><b>has to</b></i></code> validate
   * the token against the appropriate schema.
   *
   * @return The infobox token. If <code>null</code> the token is returned by
   *  method {@link #getBase64Token()} as base64 encoded string.
   */
  public Element getXMLToken();

  /**
   * Returns the infobox token.
   * Maybe <code>null</code> if the token is returned by method
   * {@link #getXMLToken()}as a DOM element.
   * <br>
   * Note, that the base64 encoded string actually may include more than one
   * infobox elements.
   *  *
   * @return The infobox token. If <code>null</code> the token is returned by
   *  method {@link #getBase64Token()} as base64 encoded string.
   *
   */
  public String getBase64Token();
}
```

Die Kernmethoden in diesem Interface sind die Methoden `getXMLToken()` und `getBase64Token()`, die das Infobox-Token zurückgeben. Je nachdem, wie der Inhalt der Infobox von der BKU übermittelt wird, wird das entsprechende Infobox-Token entweder als Objekt von Typ `org.w3c.dom.Element` (Infobox-Inhalt wird von der BKU als *XMLContent* übermittelt) über die Methode `getXMLToken()`, oder als base64 kodierter `String` (Infobox-Inhalt wird von der BKU als *BaseContent* übermittelt) durch die Methode `getBase64Token()` zur Verfügung gestellt. Dabei ist zu beachten, dass das jeweilige Token ***nicht validierend*** geparst wurde – das ist innerhalb von MOA-ID auch nicht möglich, da MOA-ID den Infobox-Inhalt nicht kennt, und daher dem Token auch kein Schema zuordnen kann – MOA-ID kann den InfoboxReadResponse also nur validierend gegen das SecurityLayer-Schema parsen – das validierende Parsen des Token gegen das entsprechende applikationsspezifische Schema ***muss*** von der jeweiligen Prüfapplikation übernommen werden. Weiters ist zu beachten, dass der über die Methode `getBase64Token()` zurückgegebene base64 kodierte String unter Umständen auch mehrere Token enthalten kann (wenn die BKU hier im Base64Content mehrere Token zusammenfasst). Die Methode getXMLToken() gibt in jedem Fall nur ein einziges Token zurück –wenn im entsprechenden XMLContent mehrere Token enthalten sind, erzeugt MOA-ID für jedes dieser Token ein eigenes `InfoboxToken` Objekt. Die Methoden `getBase64Token()` und `getXMLToken()` schließen einander wechselseitig aus, das heißt, liefert die Methode `getXMLToken()` das Infobox-Token als Objekt vom Typ `org.w3c.dom.Element`, dann gibt die Methode `getBase64Token()` den Wert null zurück, und umgekehrt.

- `public String getTrustProfileID()`: die ID des Trustprofils, das für die Validierung von Zertifikaten verwendet wird. Diese ID wird aus der Konfigurationsdatei entnommen. Falls für die Validierung eines Infobox-Tokens keine Zertifikate überprüft werden müssen, wird die Prüf-Applikation (das ist die Applikation, die das Infobox-Token validiert) diesen Parameter ignorieren.
- `public boolean getBusinessApplication()`: `true`, wenn es sich um eine privatwirtschaftliche Applikation handelt, `false`, wenn die nachfolgende Applikation eine Anwendung aus dem öffentlichen Bereich ist. Dies kann für die Validierung des Infobox-Tokens relevant sein, muss es aber nicht.
- `public List getSchemaLocations()`: Über diese Methode ist es möglich, der Prüf-Applikation die Lage von XML-Schemas zu übergeben, die diese zum validierenden Parsen von Infobox-Token verwenden kann. Die entsprechenden URIs werden über die Konfigurationsdatei gesetzt (siehe [MOA-ID Config]). Jeder Eintrag in der zurückgegeben Liste ist ein Objekt vom Typ `at.gv.egovernment.moa.id.auth.data.Schema`:

```
package at.gv.egovernment.moa.id.auth.data;

/**
 * Specifies the location of a schema.
 *
 * @author Harald Bratko
 */
public interface Schema {

  /**
   * The namespace URI of this schema.
   *
   * @return The namespace of this schema.
   */
  public String getNamespace();

  /**
   * The location URI of this schema.
   * Relative paths have to be interpreted relative to the
   * location of the MOA-ID config file.
```

```
    *
    * @return The location URI of this schema.
    */
   public String getSchemaLocation();
}
```

Anmerkung: Schemas, die die Prüfapplikation benötigt, sind wie folgt handzuhaben:

- Die Konfigurationsdatei spezifiziert die Lage zusätzlicher Schemas (siehe oben).
- Wird die Prüfapplikation in das Standard-MOA-ID Modul integriert, so können entsprechende Schemas alternativ auch im *res/resources/schemas* Verzeichnis des *common*-Projektes hinterlegt werden.
- Schemas können auch in einem Resources-Verzeichnis der Prüfapplikation vorgehalten werden, wobei der Verzeichnisname zumindest dem vollen Package-Namen der obersten Hierarchie des entsprechenden Validator-Packages (siehe Abschnitt 4.7.2.3 *Zuordnung eines InfoboxReadResponse zu einer validierenden Klasse*) entsprechen muss. In diesem Fall muss dieses Verzeichnis der jar-Datei, die die Klassen der Prüfapplikation enthält, beigefügt werden.

- `public String getBkuURL()`: Die URL der Bürgerkartenumgebung.
- `public String getTarget()`: Der Target-Parameter. Es kann sein, dass die Prüfapplikation FremdbPks berechnen muss.
- `public String getFamilyName()`: Der Familienname aus der Personenbindung. Dieser kann für die Validierung des Infobox-Tokens relevant sein, muss es aber nicht.
- `public String getGivenName()`: Der Vorname aus der Personenbindung. Dieser kann für die Validierung des Infobox-Tokens relevant sein, muss es aber nicht.
- `public String getDateOfBirth()`: Das Geburtsdatum aus der Personenbindung. Dieses kann für die Validierung des Tokens relevant sein, muss es aber nicht.
- `public String getIdentificationValue()`: Die Stammzahl (öffentlicher Bereich) oder das wbPK (privatwirtschaftlicher Bereich) aus der Personenbindung. Dieser Parameter wird nur dann übergeben, wenn dies über die Konfigurationsdatei erlaubt wird (siehe [MOA-ID Config]).
- `public String getIdentificationType()`: Der Typ der Stammzahl (öffentlicher Bereich) oder des wbPKs (privatwirtschaftlicher Bereich) aus der Personenbindung. Dieser Parameter wird vor allem für Applikationen aus dem privatwirtschaftlichen Bereich von Interesse sein.
- `public PublicKey[] getPublicKeys()`: Die öffentlichen Schlüssel aus der Personenbindung. Können für die Validierung des Tokens relevant sein, müssen es aber nicht.
- `public boolean getHideStammzahl()`: Diese Methode gibt den Wert `true` zurück, wenn die nachfolgende Applikation die Stammzahl **nicht** erhalten darf. In diesem Fall **muss** die Prüfapplikation die Stammzahl in **allen** an MOA-ID zurückzugebenden `SAML Attributen` (die eine Stammzahl enthalten) ausblenden bzw. durch einen leeren String ersetzen.
- `public Element getIdentityLink()`: Die Personenbindung. Dieser Parameter wird nur dann übergeben, wenn dies über die Konfigurationsdatei erlaubt wird (siehe [MOA-ID Config]). Im Regelfall wird die Prüfapplikation mit den bereits zur Verfügung gestellten Daten aus der Personenbindung (Familienname, Vorname, Geburtsdatum, Typ und ggf. Wert der Stammzahl (oder des wbPKs), sowie öffentliche Schlüssel) das Auslangen finden. Die gesamte Personenbindung sollte nur dann an die Prüf-Applikation weitergegeben werden, wenn für die Prüfung des Token noch andere

Parameter aus ihr relevant sind. Die Stammzahl darf innerhalb der Personenbindung *nur dann* an die Prüfapplikation übergeben werden, wenn diese die Stammzahl erhalten darf (siehe [MOA-ID Config]).

- `public Element getApplicationSpecificParameters()`: Dieses Element enthält (prüf-)applikationsspezifische Parameter und wird, falls vorhanden, genau so übergeben, wie es in der Konfigurationsdatei spezifiziert ist (siehe [MOA-ID Config]). Es obliegt der Prüf-Applikation, dieses Element richtig zu interpretieren.

4.7.2.1 Erfolgreiche erweiterte Validierung

Die `validate(InfoboxValidatorParams params)` Methode gibt ein Objeckt vom Typ `at.gv.egovernment.moa.id.auth.data.InfoboxValidationResult` zurück:

```
package at.gv.egovernment.moa.id.auth.data;

/**
 * Includes the result of an extended infobox validation.
 *
 * If validation succeeds, an array of
 * {@link at.gv.egovernment.moa.id.auth.data.ExtendedSAMLAttribute
 * ExtendedSAMLAttributes} maybe provided. Each of these SAML-Attributes will be either
 * appended to the final SAML-Assertion passed to the online application or to the
 * AUTH-Block, or to both.
 * <br>
 * If validation fails the implementing class has to provide a short error message.
 *
 * @author Harald Bratko
 */
public interface InfoboxValidationResult {

  /**
   * The method returns <code>true</code> if validation succeeds. In that case
   * method {@link  #getExtendedSamlAttributes()} may provide an array of
   * {@link at.gv.egovernment.moa.id.auth.data.ExtendedSAMLAttribute
   * ExtendedSAMLAttributes} that should be appended to the final SAML-Assertion or the
   * AUTH-Block or to both.
   * <br>
   * The method returns <code>false</code> if validation fails. In that case
   * method {@link #getErrorMessage()} has to provide a short error description.
   *
   * @return <code>True</code> if validation succeeds,
   *         otherwise <code>false</code>.
   */
  public boolean isValid();

  /**
   * Returns an array of {@link at.gv.egovernment.moa.id.auth.data.ExtendedSAMLAttribute
   * ExtendedSAMLAttributes} that should be added to the SAML-Assertion
   * provided to the online application.
   * The SAML-Attributes in that array will be added to the final
   * SAML-Assertion, the AUTH-Block, or both, exactly in the order as they are arranged
   * in the array this method returns.
   *
   * @return An array of {@link at.gv.egovernment.moa.id.auth.data.ExtendedSAMLAttribute
   * ExtendedSAMLAttributes} that should be added to the SAML-Assertion
   * provided to the online application, the AUTH-Block, or both. If no attributes should
   * be added this array maybe <code>null</code> or empty.
   */
  public ExtendedSAMLAttribute[] getExtendedSamlAttributes();

  /**
   * A short error description that should be displayed by MOA-ID if
   * validation of the InfoBoxReadResponse fails.
   *
   * @return An short error message if validation fails.
```

```
    */
   public String getErrorMessage();

}
```

Kann die Prüf-Applikation das Infobox-Token erfolgreich validieren, so wird über dieses Interface in der Regel eine Reihe von SAML-Attributen, die in die Anmeldedaten aufzunehmen sind, zurückgegeben. MOA-ID kann diese Attribute über die Methode

`public ExtendedSAMLAttribute[] getExtendedSamlAttributes()`

abrufen. Jedes darin enthaltene Attribut wird in genau der Reihenfolge, in der es im zurückgegebenen Array aufscheint, an den AUTH-Block oder an die SAML-Assertion, die an die nachfolgende Online-Applikation übergeben wird, oder an beide angehängt. Die Prüf-Applikation muss zu diesem Zweck das Interface at.gv.egovernment.moa.id.auth.data.ExtendedSAMLAttribute implementieren:

```
package at.gv.egovernment.moa.id.auth.data;

/**
 * A SAML-Attribute to be appended to the final SAML-Assertion
 * that will be passed to the online application.
 */
public interface ExtendedSAMLAttribute {
  /**
   * Add this attribute only to the SAML-Assertion
   * passed to the online application, but not to
   * the AUTH-Block.
   */
  public final static int NOT_ADD_TO_AUTHBLOCK = 0;
  /**
   * Add this attribute to both, the AUTH-Block and the
   * final SAML-Assertion passed to the online application.
   */
  public final static int ADD_TO_AUTHBLOCK = 1;
  /**
   * Add this attribute to only the AUTH-Block, but not
   * to the final SAML-Assertion passed to the online application.
   */
  public final static int ADD_TO_AUTHBLOCK_ONLY = 2;

  /**
   * The value of the SAML-Attribute. This must be either a
   * <code>org.w3c.Element</code> or a <code>java.lang.String</code>
   * object. Each other type will be ignored. <br>
   * If, for example, the type of the actual SAML-Attribute is a
   * <code>&lt;xsd:boolean&gt;</code> the value must be either the String
   * <code>&quot;true&quot;</code> or <code>&quot;false&quot;</code>.
   * Or the <code>&lt;xsd:integer&gt;</code> number <code>273</code>
   * has to be the String <code>&quot;273&quot;</code>.
   *
   * @return The value of the SAML-Attribute. Must not be <code>null</code>.
   */
  public Object getValue();

  /**
   * The name of the SAML-Attribute.
   *
   * @return The name of the SAML-Attribute. Must not be <code>null</code>.
   */
  public String getName();

  /**
   * The namespace of the SAML-Attribute.
```

```
    * An application will use the context specific namespace URI for the attribute
    * it returns.
    * However, if the application cannot explicitely assign a namespace URI, the
    * {@link at.gv.egovernment.moa.util.Constants#MOA_NS_URI default} MOA namespace URI
    * should be used.
    *
    * @return The namespace of the SAML-Attribute. Must not be <code>null</code>.
    */
  public String getNameSpace();

  /**
    * Specifies if this SAML-Attribute should be added to the AUTH-Block.
    * <br>
    * Depending on the returned value, this SAML-Attribute should be only added to the
    * final SAML-Assertion passed to the online application (0), to both, the final
    * assertion and the AUTH-Block (1) or to the AUTH-Block only (2).
    *
    * @return <ul>
    *         <li>0 - add this SAML-Attribute to the final SAML-Assertion only</li>
    *         <li>1 - add this SAML-Attribute to both, the final SAML-Assertion and the
    *             AUTH-Block</li>
    *         <li>2 - add this SAML-Attribute to the AUTH-Block only
    *         </ul>
    */
  public int getAddToAUTHBlock();

}
```

Das Objekt, das von diesem Interface über die Methode `getValue()` zurückgegeben wird, hat entweder vom Typ `org.w3c.dom.Element` oder vom Typ `java.lang.String` zu sein. Andere Typen (z.B. `<xsd:boolean>`, `<xsd:integer>`) müssen als Objekt vom Typ `java.lang.String` übergeben werden. Wird ein Objekt vom Typ `org.w3c.dom.Element` zurückgegeben, so wird dieses als einziges Kind-Element des entsprechenden `<saml:AttributeValue>`-Elements in die Anmeldedaten aufgenommen. Ein Objekt vom Typ `java.lang.String` wird als Text-Knoten im entsprechenden `<saml:AttributeValue>` eingefügt.

Im Regelfall wird die Prüf-Applikation dem `SAML-Attribut` den kontext-spezifischen Namespace zuordnen (z.B.: `"http://reference.e-government.gv.at/namespace/mandateprofile/20041105#"` für ein Attribut, das von einer Vollmachten-Prüfapplikation zurückgegeben wird). Kann jedoch kein Namespace zugeordnet werden, dann kann der Default-MOA-Namespace („`http://reference.e-government.gv.at/namespace/moa/20020822#`“) als `AttributeNamespace` verwendet werden.

Gibt einer der Methoden `getName()`, `getValue()` oder `getNameSpace()` den Wert `null` zurück, so wird MOA-ID den Anmeldevorgang mit einer entsprechenden Fehlermeldung abbrechen.

Es kann notwendig sein, dass ein `SAML-Attribut` auch in den vom Bürgerkarten-Besitzer zu signierenden `AUTH-Block` aufzunehmen ist. Dazu stellt das Interface `ExtendedSAMLAttribute` die Methode `public int getAddToAUTHBlock()` zur Verfügung. Abhängig vom Rückgabewert dieser Methode wird das entsprechende `SAML-Attribut` *nur* an die Anmeldedaten (Rückgabewert = 0), an die Anmeldedaten *und* den AUTH-Block (Rückgabewert = 1) oder *nur* an den `AUTH-Block` (Rückgabewert = 2) angehängt (das Interface stellt dazu entsprechende Konstanten zur Verfügung). Falls diese Methode einen anderen Wert zurückgibt, ist mit einer entsprechenden Fehlermeldung abzubrechen.

Es ist zu beachten, dass ein Hinzufügen eines `SAML-Attributes` zum `AUTH-Block` noch nicht bedeutet, dass der darin enthaltene Wert auch signiert wird. Dies muss durch die entsprechende Transformationskette, die sowohl global, als auch je Onlineapplikation konfiguriert werden kann (siehe [MOA-ID Config]), sichergestellt werden. Aus diesem Grund sollten an den AUTH-Block angehängte `SAML-Attribute` eher vom Typ `java.lang.String`

sein, als komplexe XML-Strukturen (es macht zum Beispiel keinen Sinn, eine komplette Vollmacht an den AUTH-Block anzuhängen).

Da MOA-ID die Verarbeitung mehrer Infoboxen gestattet, muss gewährleistet sein, dass die über die jeweiligen Prüf-Applikationen zurückgegebenen `SAML-Attribute` in der entsprechenden Reihenfolge den Anmeldedaten oder ggf. dem `AUTH-Block` hinzufügt werden. Wenn eine Prüf-Applikation ein Infobox-Token nur validiert, aber den Anmeldedaten oder dem AUTHBlock keine Attribute hinzufügt, so gibt die Methode `getExtendedSamlAttributes()`ein *leeres* `Array` oder den Wert `null` zurück.

Der Anmeldevorgang wird mit dem Signieren des `AUTH-Blocks` nur dann fortgesetzt, wenn *alle* Infoboxen erfolgreich überprüft werden konnten.

4.7.2.2 Fehlerhafte erweiterte Validierung

Wenn die Validierung eines Infobox-Tokens fehlschlägt, so gibt die Methode `isValid()` des `InfoboxValidationResult` den Wert `false` zurück. In diesem Fall hat die implementierende Klasse MOA-ID über die Methode

```
public String getErrorMessage()
```

eine kurze Meldung, die die Ursache des Fehlschlagens beschreibt, zur Verfügung zu stellen. Diese Meldung muss in einer für einen durchschnittlichen Benutzer verständlichen Form abgefasst sein, da sie im Browser-Fenster wie folgt (am Beispiel einer Vollmachten Infobox) angezeigt wird:

Überprüfung der Mandates-Infobox fehlgeschlagen: text

Anstelle des `text`-Parameters wird die von der Prüfapplikation zurückgegebene Fehlermeldung angezeigt.

4.7.2.3 Zuordnung eines InfoboxReadResponse zu einer implementierenden Klasse

Wie bereits eingangs erwähnt, steuert der Benutzer über die BKU, welche weiteren Infoboxen im Rahmen eines Anmeldevorgangs ausgelesen werden sollen. Die BKU fügt in die Antwort auf den Request zum Auslesen der Personenbindung die Inhalte dieser weiteren Infoboxen als zusätzliche Parameter an den *XML-RESPONSE*-Parameter (der die Personenbindung enthält) an (vgl. [SecLayerPI]), z.B:

`XML-RESPONSE=<sl:InfoboxReadResponse>//Personenbindung</sl:InfoboxReadResponse>`

`&Mandates=<sl:InfoboxReadResponse>//Vollmachten</sl:InfoboxReadResponse>`

`&EHSPToken=<sl:InfoboxReadResponse>//GDA-Token</sl:InfoboxReadResponse>`

Die Namen dieser zusätzlichen Parameter entsprechen dabei exakt den Bezeichnern der jeweiligen Infoboxen, also z.B. `Mandates` für die Infobox `Mandates`, oder `EHSPToken` für die Infobox `EHSPToken`. Dies ist von essentieller Bedeutung, da MOA-ID a priori nicht wissen kann, welche Infobox welcher Prüf-Applikation zuzuordnen ist. Indem man nun sowohl dem Parameter, der den jeweiligen InfoboxResponse aus der von der BKU übermittelten Antwort enthält, als auch der das Interface `InfoboxValidator` implementierenden Klasse der darunterliegenden Prüf-Applikation einen einheitlichen Namen gibt, kann MOA-ID den InfoboxReadResponse eindeutig einer Prüf-Applikation zuordnen. Damit ergibt sich für den Namen der implementierenden Klasse folgende zwingend einzuhaltende Vorschrift:

Package-Name:

`at.gv.egovernment.moa.id.auth.validator.[lowercase(Infobox-Bezeichner)]`

Klassenname:

`[Infobox-Bezeichner]Validator`

Ist der erste Buchstabe des Infobox-Bezeichners ein *Klein-Buchstabe*, so hat der Klassennamen mit dem entsprechenden *Groß-Buchstaben* zu beginnen.

Beispiel1:

Infoboxbezeichner: `Mandates`

Parameter-Name in der von der BKU übermittelten Antwort: `Mandates`

Klassenname:

`at.gv.egovernment.moa.id.auth.validator.mandates.MandatesValidator`

Beispiel2:

Infoboxbezeichner: `EHSPToken`

Parameter-Name in der von der BKU übermittelten Antwort: `EHSPToken`

Klassenname:

`at.gv.egovernment.moa.id.auth.validator.ehsptoken.EHSPTokenValidator`

Diese Default-Klassen können auch über die Konfigurationsdatei überschrieben werden, wobei Validator-Klassen sowohl global als auch lokal je Onlineapplikation gesetzt werden können (siehe [MOA-ID Config]). MOA-ID reagiert folgendermaßen auf einen BKU-Response, der außer der Personenbindung noch weitere `InfoboxReadResponses` enthält:

Die Personenbindung wird wie bisher behandelt. Für alle weiteren Infoboxen überprüft MOA-ID wie folgt anhand des jeweiligen Parameternamens (s.o.), ob eine das Interface `InfoboxValidator` implementierende Klasse vorhanden ist, und initiiert durch Aufruf der `validate()`-Methode die Prüfung des entsprechenden Tokens:

- Ist über die Konfiguration (lokal) für die aktuelle Online-Applikation eine Validator-Klasse für die jeweilige Infobox angegeben, so wird diese für die Überprüfung verwendet.
- Ist über die Konfiguration (lokal) für die aktuelle Online-Applikation *keine* Validator-Klasse angegeben, so wird, falls vorhanden, die global für die aktuelle Infobox spezifizierte Validator-Klasse verwendet.
- Ist sowohl lokal für die aktuelle Online-Applikation als auch global keine Validator-Klasse angegeben, so wird die Default-Validatorklasse (s.o.) für die aktuelle Infobox verwendet. Dies ist der Regelfall – das heißt, im Normalfall wird es *nicht* notwendig sein, Validatorklassen über die Konfigurationsdatei zu setzen.
- Kann MOA-ID keine passende Validator-Klasse für eine Infobox finden, so wird mit einer entsprechenden Fehlermeldung abgebrochen. Dieser Fall sollte jedoch niemals auftreten, da MOA-ID im Rahmen des `InfoboxReadRequests` zum Auslesen der Personenbindung der BKU über den `PushInfobox`-Parameter mitteilt (siehe auch Abschnitt 4.2.2 *Abfrage der Personenbindung und ggf. weiterer Infoboxen*), welche Infoboxen überprüft werden können. Daher kann MOA-ID nur im Falle einer fehlerhaften Konfiguration einen Infobox-Inhalt von der BKU erhalten, der nicht bearbeitet werden kann (bzw. für den keine entsprechende Validator-Klasse zur Verfügung steht).

4.7.3 Überprüfung des signierten AUTH-Blocks

Die Überprüfung der Signatur wird mittels des Kommandos `<VerifyXMLSignatureRequest>`

des MOA-Modul für Signatur-Prüfung mit folgenden verpflichtenden Eingangs-Parametern durchgeführt:

<VerifySignatureInfo>	enthält den signierten AUTH-Block
<SignatureManifestCheckParams>	Das Attribut ReturnReferenceInputData muss auf true gesetzt sein.
<ReturnHashInputData>	muss enthalten sein
<TrustProfileID>	Eine Referenz auf ein Profil von Wurzelzertifikaten

Der AUTH-Block muss mit einem privaten Schlüssel des Benutzers signiert sein, dessen korrespondierender öffentliche Schlüssel Teil der Personenbindung ist.

5 Proxykomponente

Die Proxykomponente ermöglicht eine einfache Anbindung bestehender Online-Applikationen an den Authentisierungs-Mechanismus des MOA Identifikationsmoduls, indem es:

- die SAML-Struktur von der Authentisierungskomponente anfordert, um die Identität des Benutzers zu sichern
- eine einmalige Authentisierung an der Online-Applikation vornimmt („stateful“ Online-Applikation) oder die Login-Parameter bei jedem Zugriff auf die Online-Applikation übergibt („stateless“ Online-Applikation).
- die Online-Applikation durch transparentes Proxying vor nicht authentisierten Zugriffen schützt.

5.1 Schnittstelle zum Aufbau des Requests an die Online-Applikation

Zum Aufbau des Requests an die Online-Applikation muss innerhalb der Proxykomponente eine Klasse aufgerufen werden, die folgendes Interface implementiert. Die implementierende Klasse muss in der Proxykomponente global konfiguriert werden können. Die Proxykomponente muss eine Default-Implementierung bereitstellen, die die im Abschnitt 5.5 *Konfiguration* beschriebenen Konfigurationsmöglichkeiten abdeckt.

```
/**
 * Builds an HttpURLConnection to a java.net.URL which is derived
 * from an {@link HttpServletRequest} URL, by substitution of a
 * public URL prefix for the real URL prefix.<br>
 * The HttpURLConnection has been created by java.net.URL#openConnection, but
 * it has not yet been connected to by java.net.URLConnection#connect.<br>
 * The field settings of the HttpURLConnection are:
 * <ul>
 * <li><code>allowUserInteraction = false</code></li>
 * <li><code>doInput = true</code></li>
 * <li><code>doOutput = true</code></li>
 * <li><code>requestMethod = request.getMethod()</code></li>
 * <li><code>useCaches = false</code></li>
 * </ul>
 *
 * @param request
 *        the incoming request which shall be forwarded
 * @param publicURLPrefix
 *        the public URL prefix to be substituted by the real URL prefix
 * @param realURLPrefix
 *        the URL prefix to substitute the public URL prefix
 * @param sslSocketFactory
 *        factory to be used for creating an SSL socket in case
 *        of a URL for scheme <code>"https:"</code>;
 *        if <code>null</code>, the default SSL socket factory would be used
 * @return a URLConnection created by java.net.URL#openConnection,
 *         connecting to the requested URL with <code>publicURLPrefix</code>
 *         substituted by <code>realURLPrefix</code>
 * @throws IOException if an I/O exception occurs during opening the connection
 * @see java.net.URL#openConnection()
 * @see com.sun.net.ssl.HttpsURLConnection#getDefaultSSLSocketFactory()
 */
```

5.2 Schnittstelle zur Online-Applikation

Online-Applikationen werden unterschieden in stateless und stateful Online-Applikationen.

Stateless Online-Applikationen verlieren alle Benutzerinformation, wenn die Verbindung abgebaut wird (typischerweise am Ende eines HTTP-Requests). Es ist also notwendig, Benutzerinformatin (z.B. bPK) bei jedem Request mitzugeben.

Stateful Online-Applikationen behalten die Benutzerinformation auch wenn die Verbindung abgebaut wird. Üblicherweise werden für stateful Online-Applikationen Cookies eingesetzt. Benutzerinformation ist also nur beim Session-Aufbau notwendig.

Aus Sicht der Proxykomponente sind stateful Online-Applikationen eine Untermenge (ein Spezialfall) von stateless Online-Applikationen. Damit ist das Session-Handling der Online-Applikation (Timeout, etc.) für die Proxykomponente transparent.

5.2.1 Typisches Setup für stateless Online-Applikationen

Bei stateless Online-Applikationen muss jeder HTTP-Request und die dazugehörige Antwort der Online-Applikation durch die Proxykomponente gehen. Um performance-kritisches URL-Rewriting zu vermeiden, muss folgende Lösung verwendet werden:

- Der DNS-Eintrag der Online-Applikation verweist auf die Proxykomponente
- Die Proxykomponente kennt zu jedem DNS-Eintrag die wirkliche Adresse der Online-Applikation
- Die Online-Applikation ist mittels Virtual Hosting so konfiguriert, dass sie Requests, die auf den nach außen sichtbaren Domänen-Namen der Online-Applikation lauten, entgegennimmt

5.2.2 Typisches Setup für stateful Online-Applikationen

- Die Proxykomponente und die Online-Applikation haben unterschiedliche DNS-Namen, wobei einer der Namen eine Subdomain der anderen ist.
- Der erste Zugriff nach dem Authentisierungsvorgang erfolgt auf eine „Login-URL“, deren DNS-Name auf die Proxykomponente auflöst. Die Proxykomponente leitet wie im Fall einer stateless Online-Applikation den Request an die Online-Applikation weiter (die Online-Applikation muss also auch per Virtual Hosting diese URL auflösen können).
- Das Login-Script der Online-Applikation prüft, ob der Request von der Proxykomponente kommt. Wenn ja, baut sie die Session auf (z.B. durch Setzen eines Cookies seitens der Online-Applikation im Browser des Benutzers, das für die Domäne der Online-Applikation gültig ist). Enstprechendes Firewall-Setup vorausgesetzt genügt ein einfaches Prüfen der IP-Adresse.
- Von diesem Zeitpunkt an gehen alle Requests an der Proxykomponente vorbei direkt an die Online-Appliktion, da der Benutzer an eine entsprechende Session gebunden ist.
- Um das Problem des Timeouts bzw. des Logout zu umgehen, besitzt die Proxykomponente ein sehr kurzes Timeout (Sekunden bzw. Minutenbereich). Das Session-Management obliegt ganz der Online-Appliktion.

5.3 Schnittstelle zum Browser des Benutzers

Die Proxykomponente verfügt über eine eigene HTTP-Session mit dem Benutzer, in der folgende Informationen server-seitig hinterlegt werden:

- Die Anmeldedaten des Benutzers

5.4 Proxy-Funktion

Die Proxykomponente muss bei stateless Online-Applikationen als transparenter Proxy zur

Online-Applikation fungieren:

- Alle Requests zur Online-Applikation müssen von der Proxykomponente auf ihre Berechtigung geprüft werden: ein direkter Zugriff ohne vorherige Verwendung der Authentisierungskomponente darf nicht möglich sein.
- Alle Requests müssen mit allen Parametern zur Online-Applikation weitergeleitet werden. Ausnahme: die in der Konfiguration der Online-Applikation angegebenen Login-Parameter müssen gefiltert werden. Im Falle einer stateless Online-Applikation müssen die Login-Parameter bei jedem Request mitgegeben werden.
- Die von der Online-Applikation generierte Antwort muss unverändert an den Benutzer zurückgegeben werden.

5.5 Konfiguration

5.5.1 Konfiguration in der Proxykomponente

Für die Proxykomponente muss folgende Konfiguration durchgeführt werden können:

- Angabe des HTTP-Session-Timeout des Benutzers in der Proxykomponente. Nach einem Ablauf der Session ist der Benutzer nicht mehr an der Proxykomponente authentisiert. Alle server-seitig in der Session gespeicherten Benutzerdaten müssen gelöscht werden.
- Angabe der Klasse, die das `LoginParameterResolver` Interface implementiert (optional)

Folgende Daten müssen für die Kommunikation mit der Authentisierungskomponente konfigurierbar sein:

- Konfiguration ob die Kommunikation mit der Authentisierungskomponente via SOAP- oder via API-Aufrufe erfolgen soll.
- Angabe, ob die Schnittstelle zur Authentisierungskomponente mit TLS-Client und/oder TLS-Server-Authentisierung geschützt werden muss.
    - verwendetes TLS-Client-Zertifikat
    - akzeptierte TLS-Server-Zertifikate der Authentisierungskomponente
    - vertrauenswürdige Wurzelzertifikate für akzeptierte TLS-Server-Zertifikate der Authentisierungskomponente

In der Proxykomponente müssen je Online-Applikation folgende Konfigurationsmöglichkeiten gegeben sein:

- Eine optionale URL der Konfigurationsdatei für die Online-Applikation (siehe nächster Abschnitt). Falls die URL nicht angegeben ist, wird implizit folgende URL angenommen: `http://<oa-domain>/moaconfig.xml`, wobei `<oa-domain>` die nach außen sichtbare Domäne der Online-Applikation angibt.
- Das Mapping des nach außen sichtbaren Domänen-Namens (`<oa-domain>`) auf die wirkliche Domäne der Online-Applikation.
- Angabe, ob die Schnittstelle zur Online-Applikation mit TLS-Client und/oder TLS-Server-Authentisierung geschützt werden muss.
    - verwendetes TLS-Client-Zertifikat
    - akzeptierte TLS-Server-Zertifikate der Online-Applikation
    - vertrauenswürdige Wurzelzertifikate für akzeptierte TLS-Server-Zertifikate der Online-Applikation

Ein eindeutiges Präfix der URL der Online-Applikation muss deshalb als Schlüssel verwendet

werden. Die URL muss zumindest aus `https://` und einem gültigen Domänen-Namen bestehen.

5.5.2 Konfiguration der Online-Applikation

Die Konfigurationsdatei beschreibt die Art und Weise, wie die Proxykomponente die Anmeldung an der Online-Applikation durchführt. Die Lage der Konfigurationsdatei wird in der Konfiguration der Proxykomponente als URL hinterlegt. Die Konfigurationsdatei soll standardmäßig vom Web-Server der nachfolgenden Applikation geladen werden.

Die Konfigurationsdatei ist eine XML-Datei, deren Format in [MOA-ID Config Schema] beschrieben ist, und die folgende Element enthält:

Name	Beschreibung
`login-type`	Gibt an, ob die Online-Applikation ein einmaliges Login erwartet (`stateful`), oder ob die Login-Parameter bei jedem Request mitgegeben werden müssen (`stateless`). Im Fall einer stateful Online-Applikation werden die in der HTTP-Session der Proxykomponente gespeicherten Anmeldedaten nur für den Aufruf des Login-Scripts verwendet. Unmittelbar nach dem Aufruf müssen sie gelöscht werden.
`param-auth`	Konfiguriert die Übergabe der Authentisierungs-Parameter an die Online-Applikation mittels URL-Parametern.
`basic-auth`	Konfiguriert die Übergabe der Authentisierungs-Parameter an die Online-Appliktion mittels HTTP Basic Authentication.
`header-auth`	Konfiguriert die Übergabe der Authentisierungs-Parameter an die Online-Applikation in HTTP Request Headern.

5.5.2.1 Login mittels URL-Parameter

Das Element <param-auth> enthält eine Aufzählung von <parameter> Elementen, die folgende Attribute besitzen:

Name	Beschreibung
`Name`	Beschreibt den Namen des Parameters und ist ein frei zu wählender String.
`Value`	Beschreibt den Inhalt des Parameters und kann einen der durch MOAAuthDataType beschriebenen Werte annehmen.

Anhand der <parameter>-Elemente wird der Request für den Login-Vorgang (für stateful Online-Applikationen) folgendermaßen zusammengestellt:

`GET https://<login-url>?<p1.name=p1.resolvedValue>&<p2.name=p2.resolvedValue>...`

Die <login-url> ergibt sich aus dem Parameter OA des ursprünglichen Requests an die Authentisierungskomponente (siehe 4.1 *Schnittstelle zur verweisenden Webseite*). Der Wert `resolvedValue` wird in der Proxykomponente je nach Wert des Platzhalters eingesetzt.

5.5.2.2 Login mittels Basic Authentication

Das Element <basic-auth> enthält folgende 2 Elemente, die die UserId und das Passwort spezifizieren und für eine Basic Authentication analog [HTTP-Auth] angeben:

Name	Beschreibung
`Userid`	Die UserId des zu authentisierenden Benutzers; kann einen der durch

	MOAAuthDataType beschriebenen Werte annehmen.
Password	Das Passwort des zu authentisierenden Benutzers; kann einen der durch MOAAuthDataType beschriebenen Werte annehmen.

Aus diesen Angaben müssen je nach Wert der Platzhalter die entsprechenden HTTP-Header für die Basic Authentication erstellt und in den HTTP-Request an die Online-Applikation eingefügt werden.

5.5.2.3 Login mittels HTTP Request Header

Das Element <header-auth> enthält eine Aufzählung von <header> Elementen, die folgende Attribute besitzen:

Name	Beschreibung
Name	Beschreibt den Namen des Headers und ist ein frei zu wählender String.
Value	Beschreibt den Inhalt des Headers und kann einen der durch MOAAuthDataType beschriebenen Werte annehmen.

Die Header werden folgendermaßen in den Request an die Online-Applikation eingefügt:

<h1.name>:<h1.resolvedValue>

<h2.name>:<h2.resolvedValue>

...

Der Wert resolvedValue wird in der Proxykomponente je nach Wert des Platzhalters eingesetzt. Etwaige Header aus dem ursprünglichen Request an die Proxykomponente, die den selben Namen haben, müssen überschrieben werden.

5.5.2.4 MOAAuthDataType

Der Typ MOAAuthDataType dient an vielen Stellen der in diesem Abschnitt beschriebenen Konfiguration als Platzhalter für Anmeldedaten des Benutzers. Im Zuge des Anmeldevorgangs an der Online-Applikation müssen diese Platzhalter durch die entsprechenden Anmeldedaten des Benutzers ersetzt werden. Folgende Platzhalter sind definiert:

- MOAGivenName: der Vorname des Benutzers, wie in der PB enthalten
- MOAFamilyName: der Nachname des Benutzers, wie in der PB enthalten
- MOADateOfBirth: das Geburtsdatum des Benutzers, wie in der PB enthalten
- MOABPK: das BPK des Benutzers, wie von der Authentisierungskomponente
- MOAWBPK: das von der Bürgerkartenumgebung im Rahmen der Personenbindung übergebene WBPK
- MOAPublicAuthority: wird durch true ersetzt, falls es sich um einen Benutzer einer Behörde handelt, andernfalls wird false gesetzt
- MOABKZ: das Behördenkennzeichen (nur sinnvoll, wenn MOAPublicAuthority den Wert true ergibt)
- MOAQualifiedCertificate: wird durch true ersetzt, falls das Zertifikat des Benutzers qualifiziert ist, andernfalls wird false gesetzt
- MOAStammzahl: die Stammzahl des Benutzers; diese ist nur dann verfügbar, wenn die Online-Applikation die Stammzahl bekommen darf (und daher in der Personenbindung enthalten ist).
- MOAIPAddress: die IP-Adresse des Client des Benutzers

6 Definition der Schnittstelle zwischen Authentisierungskomponente und nachfolgenden Applikation

Über eine echte Client/Server-Schnittstelle verfügt nur die Authentisierungskomponente. Deren spezifizierte Funktionalität (die Abfrage der Identität) kann sowohl über SOAP (für Aufrufe über das Netzwerk) als auch über API Aufrufe verwendet werden.

6.1 SOAP

Im Dokument [SAML-Binding], Abschnitt 3, ist die Bindung von SAML-Anfragen an SOAP beschrieben. Dieses Binding muss zur Anfrage an die Authentisierungskomponente verwendet werden.

[MOA-ID WSDL] beschreibt die MOA-Schnittstelle bei Verwendung als Web-Service mit SOAP als Transportprotokoll. Aus dem WSDL-Dokument können mit geeigneten Tools Client-Stub- und Server-Tie-Objekte generiert werden.

Die SOAP Spezifikation basiert auf SOAP Version 1.1 [SOAP], die WSDL Spezifikation auf WSDL Version 1.1 [WSDL].

Das Dokument folgt dem hierarchischen Aufbau von WSDL-Dokumenten (siehe [WSDL] Abschnitt 1).

6.1.1 Beispiele für SOAP Messages

Beispiele für SOAP Requests und Response Messages zur Abfrage der SAML-Struktur sind ebenfalls in [SAML-Binding], Abschnitt 3, beschrieben.

6.1.1.1 Response im Fehlerfall:

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV :Server</faultcode>
      <faultstring>Server Error</faultstring>
      <detail>
        <moa:ErrorResponse
       xmlns:moa="http://reference.e-government.gv.at/namespace/moa/20020822#">
          <ErrorCode>2000</ErrorCode>
          <Info>Fehler im MOA Modul</Info>
        </moa:ErrorResponse>
      </detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

6.2 API

Die in [MOA-ID WSDL] definierten Aufrufe sind in geeigneten Java Interfaces modelliert. Die Art der Modellierung orientiert sich an der im Package `javax.xml.soap` definierten Interfaces ohne diese jedoch direkt zu referenzieren.

Die Server-seitige Schnittstelle ist durch das Interface `AuthenticationService` definiert.

Die Definitionen der weiteren verwendeten Interfaces sind in [MOA-ID API] dokumentiert.

6.2.1 Interface IdentificationService

```
package at.gv.egovernment.moa;
/**
 * Interface providing functions for querying authentication data.
 */
public interface AuthenticationService {
/**
 * Gets authentication data.
 *
 * @param request A request for authentication data containing a SAML artifact.
 * @return User authentication data.
 * @throws MOAException Error in server side MOA module.
 */
public GetAuthenticationDataResponse
       getAuthenticationData(GetAuthenticationDataRequest request)
       throws MOAException;
}
```

7 Referenzen

Referenz	Titel	Verfasser	Datum
[MOA-Auftrag]	Projekt Module für Online Applikationen (MOA) –Design- und Spezifikationsphase – Auftragsbeschreibung	BMF	05.07.2002
[UC BRZ]	Anwendungsfälle Spezifikation, Version 4.1	BRZ	03.09.2002
[SecLayer1.1]	Schnittstellenspezifikation Security-Layer Version 1.1; http://www.buergerkarte.at/konzept/securitylayer/spezifikation/20020831/	CIO	31.08.2002
[SecLayer1.1-Bindung]	Bindung des Schnittstellenprotokolls des Security-Layers für das Konzept Bürgerkarte an Transportprotokolle, Version 1.1; http://www.buergerkarte.at/konzept/securitylayer/spezifikation/20020831/	CIO	31.08.2002
[SecLayer1.2]	Schnittstellenspezifikation Security-Layer Version 1.2; http://www.buergerkarte.at/konzept/securitylayer/spezifikation/20040514/	CIO	18.05.2006
[SecLayerPI]	Erweiterung Spezifikation Bürgerkarte Auslesen von Infoboxen im Push-Verfahren	EGIZ	02.06.2006
[PersBind]	XML Definition der Personenbindung, Version 2.0; http://www.buergerkarte.at/konzept/personenbindung/spezifikation/20050214/	CIO	14.02.2005
[OID]	Object Identifier und Einsatz in X.509 Zertifikaten, Version 1.1	CIO	06.08.2002
[GB]	E-Government Geschäftsbereiche, Version 1.0; http://reference.e-government.gv.at/uploads/media/gb-1-0-20020524.pdf		24.05.2002
[GB-BK]	E-Government verwaltungsinterne Geschäftsbereiche - Spezifikation für den Datenschutz und die verwaltungsbereich-spezifisch abgeleitete Personenkennzeichnung, Version 1.0; http://reference.e-government.gv.at/uploads/media/gb-bk-1-0-20020524_02.pdf		24.05.2002
[MOA-ID Config]	MOA-ID Konfigurationshandbuch v.1.4 (liegt den MOA-ID-Distributionen in HTML-Format bei)		18.05.2006
[MOA-ID Config Schema]	Datei „MOA-ID-Configuration-1.1.xsd “	ARGE	30.06.2003
[MOA-ID WSDL]	Datei „MOA ID 1.x.wsdl“	ARGE	30.06.2003

[MOA-ID API]	Java Package `at.gv.egovernment.moa`	ARGE	30.06.2003
[HTTP-Auth]	HTTP Authentication: Basic and Digest Access Authentication, IETF RFC 2069	IETF	06.1999
[SAML-Assertions]	Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML), Committee Specification 01; http://www.oasis-open.org/committees/security/docs/cs-sstc-core-01.pdf	OASIS	31.05.2002
[SAML-Binding]	Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML), Committee Specification 01; http://www.oasis-open.org/committees/ security/cs-sstc-bindings-01.pdf	OASIS	31.05.2002
[TLS]	The TLS Protocol V.1.0, IETF RFC 2046	IETF	01.1999
[RFC3061]	A URN Namespace of Object Identifiers; IETF RFC 3061; http://www.ietf.org/rfc/rfc3061.txt	IETF	02.2001
[SOAP]	Simple Object Access Protocol (SOAP) V.1.1	W3C	08.05.2000
[WSDL]	Web Services Description Language (WSDL) V.1.1	W3C	15.03.2001
[XMLDSig]	XML-Signature Syntax and Processing; http://www.w3.org/TR/xmldsig-core/	W3C	12.02.2002
[MOA SP-SS]	Spezifikation MOA SP-SS 1.1	BMF/CIO	30.06.2003
[E-GovG]	Bundesgesetz über Regelungen zur Erleichterung des elektronischen Verkehrs mit öffentlichen Stellen (E-Government-Gesetz – E-GovG) http://reference.e-government.gv.at/E-Government-Gesetz.394.0.html		27.02.2004
[MOA-WID-OA]			