
| Tutorium zur österreichischen Bürgerkarte | | Konvention |
|---|---|---|
| | | 1.2.2 |
| | | Empfehlung |
| Kurzbeschreibung | Das vorliegende Dokument enthält ein Tutorium für die Bedienung der Applikationsschnittstelle Security-Layer für Entwickler von Applikationen. | |
| Autoren: | Arno Hollosi<br>Gregor Karlinger<br>Thomas Rössler<br>Martin Centner<br>et al. | Projektteam/Arbeitsgruppe<br>AG Bürgerkarte |
| Datum: | 1.8.2006 | |

**Inhaltsverzeichnis**

1. Allgemeines
2. Schnittstellenbefehle
- 2.1. Signatur erstellen
- 2.2. Signaturen prüfen
- 2.3. Daten verschlüsseln
- 2.4. Daten entschlüsseln
- 2.5. Hashwert berechnen
- 2.6. Hashwert verifizieren
- 2.7. Infoboxen
- 2.8. Abfrage von Eigenschaften
- 2.9. Null-Operation
- 2.10. Fehlerbehandlung

3. Transportprotokolle
- 3.1. TCP/IP bzw. TLS
- 3.2. HTTP bzw. HTTPS

4. Standard-Anzeigeformat SLXHTML
- 4.1. Ein erstes Beispiel
- 4.2. Ein umfangreiches Beispiel
- 4.3. Signieren von SLXHTML-Dokumenten mit Bildern

Glossar

## 1. Allgemeines

Dieses Tutorium ist für Entwickler von *Applikationen* im Bereich E-Government und E-Commerce konzipiert. Es bietet einen praxisorientierten Überblick, wie die Funktionen der *Bürgerkarte* in solche *Applikationen* integriert werden können.

Im Abschnitt 2 werden für alle Funktionen der *Bürgerkarte* beispielhafte Schnittstellenbefehle besprochen.

In Abschnitt 3 wird das Absetzen von Befehlen an die *Bürgerkarten-Umgebung* über die spezifizierten Transportprotokolle TCP/IP bzw. TLS sowie HTTP bzw. HTTPS gezeigt. Insbesondere werden die vielfältigen Möglichkeiten der beiden letzten Protokolle besprochen, den Ablauf der Befehlsabarbeitung zwischen *Bürger*, *Bürgerkarten-Umgebung* und *Applikation* zu steuern. Schließlich werden Funktionsaufrufe der *Bürgerkarte* zu Sequenzen zusammengefügt, wie sie für typische Abläufe in *Applikationen* benötigt werden.

Abschnitt 4 beschäftigt sich mit dem Standard-Anzeigeformat der *Bürgerkarte*. Es wird dort der grundsätzliche Aufbau eines entsprechenden XHTML-Dokuments besprochen sowie ein umfangreiches Beispiel gegeben, das die Möglichkeiten zur Dokumentstrukturierung und Formatierung demonstriert. Schließlich wird noch auf das Signieren von Dokumenten im Standard-Anzeigeformat eingegangen, die Bilder enthalten.

Das Dokument ist derzeit noch nicht vollständig und als Work in Progress zu betrachten. Anregungen wenden Sie sich bitte per Email an die Autoren.

## 2. Schnittstellenbefehle

Dieser Abschnitt stellt Beispiele für alle Schnittstellenbefehle vor. Sie können alle Beispiele mit Ausnahme von „2.7.4.1.2 Lesen der Personenbindung durch die Privatwirtschaft“ direkt ausprobieren, in dem Sie das Formular verwenden.

## 2.1. Signatur erstellen

### 2.1.1. Signatur im Format CMS erstellen

#### 2.1.1.1. Ein erstes Beispiel

*Anfrage*

Zunächst soll ein einfaches Beispiel vorgestellt werden: Der Text `Ich bin ein einfacher Text.` soll mit jenem Signaturschlüssel signiert werden, der eine sichere Signatur bzw. eine Verwaltungssignatur leistet. Die dazupassende Anfrage sieht wie folgt aus:

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:CreateCMSSignatureRequest
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"
[04]   Structure="enveloping">
[05]   <sl:KeyboxIdentifier>SecureSignatureKeypair</sl:KeyboxIdentifier>
[06]   <sl:DataObject>
[07]     <sl:MetaInfo>
[08]       <sl:MimeType>text/plain</sl:MimeType>
[09]     </sl:MetaInfo>
[10]     <sl:Content>
[11]       <sl:Base64Content>SWNoIGJpbiBlaW4gZWluZmFjaGVyIFRleHQu</sl:Base64Content>
[12]     </sl:Content>
[13]   </sl:DataObject>
[14] </sl:CreateCMSSignatureRequest>
```

Zeile 2 enthält den passenden Befehl der Schnittstelle *Security-Layer*.

Zeile 3 enthält die Namenraum-Deklaration für die XML-Elemente, aus denen die Anfrage besteht.

In Zeile 4 wird mittels des Attributs `Structure`, dessen Wert auf `enveloping` gesetzt ist, festgelegt, dass die zu erzeugende CMS-Signatur auch die signierten Daten enthält. Sollen die Daten selbst nicht in der Signatur kodiert werden, muss dieses Attribut auf den Wert `detached` gesetzt sein.

In Zeile 5 wird mit dem Inhalt des Elements `sl:KeyboxIdentifier` der zu verwendende Signaturschlüssel ausgewählt. Entsprechend der Spezifikation Standardisierte Key- und Infoboxen ist der Wert `SecureSignatureKeypair` anzugeben, wenn eine sichere Signatur bzw. Verwaltungssignatur erstellt werden soll.

Die Zeilen 6 bis 13 enthalten die Angaben zu den Daten, die signiert werden sollen:

Die Zeilen 7 bis 9 enthalten Metainformationen zu diesen Daten, wobei für dieses Beispiel lediglich der *Mime Type* dieser Daten von Interesse ist. Dieser ist in Zeile 8 als Inhalt des Elements `sl:MimeType` mit `text/plain` angegeben, da ja ein einfacher Text signiert werden soll. Die Angabe des *Mime Types* ist wesentlich, da sie von der *Bürgerkarten-Umgebung* für die Auswahl der passenden Anzeigekomponente ausgewertet wird.

Die Zeilen 10 bis 12 geben die zu signierenden Daten selbst an. Das Element `sl:Base64Content` beinhaltet die base64-Kodierung der Daten, die signiert werden sollen, also des Texts `Ich bin ein einfacher Text.`.

*Antwort*

Die entsprechende Antwort der *Bürgerkarten-Umgebung* sollte in etwa wie folgt aussehen. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert und gekürzt wurde, was natürlich die elektronische Signatur bricht.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:CreateCMSSignatureResponse xmlns="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[03]   <sl:CMSSignature>MIIHGQYJKoZIhvcNAQcCoIIHCjCCBwYCAQExCzAJBgUrDgMCGgUAMCoGCSqGSIb3DQEHAaAdBBtJ
[04] Y2ggYmluIGVpbiBl...Y0245Zi7CH83+/97dnOUZH9Ug2B+WAJGjAS9o97ZFSx+gowRc3FEG
[05] IivzaAL5ARJzHV7wwIizxTPWVzSk/i5Zq8qHvlv3mbJ4QF84ArxhHKVSfG7GgWFHuQVApQyk</sl:CMSSignature>
[06] </sl:CreateCMSSignatureResponse>
```

Zeile 2 zeigt die zur Anfrage passende Antwort der *Bürgerkarten-Umgebung*.

Die Zeilen 3 bis 5 enthalten die von der *Bürgerkarten-Umgebung* erzeugte CMS-Signatur. Diese ist als Textinhalt des Elements `sl:CMSSignature` in base64-kodierter Form (oben verkürzt dargestellt) abgelegt.

*Downloads zu diesem Beispiel*

- examples/interface/createCMSSignature/First.xml
- examples/interface/createCMSSignature/First.Response.xml

*Weiterführende Informationen*

- Abschnitt 2.1, „Signatur nach CMS“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 2, „Keyboxen“ in Die österreichische Bürgerkarte - Standardisierte Key- und Infoboxen
- Abschnitt 9, „Anzeigeformate und Zeichensätze“ in Die österreichische Bürgerkarte - Minimale Umsetzung des Security-Layers

#### 2.1.1.2. Zu signierende Daten als Referenz angeben

*Anfrage*

Dieses Beispiel soll im Gegensatz zum vorigen ein XHTML-Dokument signieren, das dem Standard-Anzeigeformat der Bürgerkarte entspricht. Dieses zu signierende Dokument soll in der Signaturerstellungs-Anfrage nicht direkt inkludiert, sondern per URL referenziert werden. Nachfolgend die dazupassende Anfrage:

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:CreateCMSSignatureRequest
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"
[04]   Structure="enveloping">
[05]   <sl:KeyboxIdentifier>CertifiedKeypair</sl:KeyboxIdentifier>
[06]   <sl:DataObject>
[07]     <sl:MetaInfo>
[08]       <sl:MimeType>application/xhtml+xml</sl:MimeType>
[09]     </sl:MetaInfo>
[10]     <sl:Content
[11]       Reference="http://www.buergerkarte.at/konzept/securitylayer/spezifikation/
[12]         20040514/tutorial/examples/viewerformat/Simple.xhtml"/>
[13]   </sl:DataObject>
[14] </sl:CreateCMSSignatureRequest>
```

In Zeile 5 wird festgelegt, dass dieses Mal mit dem zweiten standardisierten Signaturschlüssel der Bürgerkarte signiert werden soll (Wert `CertifiedKeypair`).

In Zeile 8 wird der passende *Mime Type* für das XHTML-Dokument angegeben, das signiert werden soll. Der Wert für XHTML-Dokumente ist `application/xhtml+xml`.

In den Zeilen 10 bis 12 wird das zu signierend XHTML-Dokument per Referenzangabe ausgewählt. Das Attribut `Reference` muss dafür als Inhalt eine URL haben, die von der *Bürgerkarten-Umgebung* aufgelöst werden kann. Bitte beachten Sie, dass der oben angegebene Wert zur besseren Lesbarkeit einen Zeilenumbruch aufweist.

#### *Antwort*

Die entsprechende Antwort der *Bürgerkarten-Umgebung* sollte in etwa wie folgt aussehen. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert und gekürzt wurde, was natürlich die elektronische Signatur bricht.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:CreateCMSSignatureResponse xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[03]   <sl:CMSSignature>MIIH7wYJKoZIhvcNAQcCoIIH4DCCB9wCAQExCzAJBgUrDgMCGgUAMIIBMwYJKoZIhvcNAQcBoIIB
[04] JASCASA8P3htbCB2ZXJzaW9u...xLjAiIGVuY29kaW5nPSJVVEYtOCI/Pg0KPGh0bWwgeG1sbnM9
[05] mP+qV0lB2W21q2LL3eMldPKsJVgArkXEduw01jNybLeyU8SE+LTe6D1tR0B/PGFHbC6Yeu0bvD2c
[06] qq06QG+hKX+11S8bZHq3EB81lMTcrSP4foYsf6aqxhBpYkHZcncEBoA==</sl:CMSSignature>
[07] </sl:CreateCMSSignatureResponse>
```

Zeile 2 zeigt die zur Anfrage passende Antwort der *Bürgerkarten-Umgebung*.

Die Zeilen 3 bis 5 enthalten die von der *Bürgerkarten-Umgebung* erzeugte CMS-Signatur. Diese ist als Textinhalt des Elements `sl:CMSSignature` in base64-kodierter Form (oben verkürzt dargestellt) abgelegt.

#### *Downloads zu diesem Beispiel*

- examples/interface/createCMSSignature/DataPerReference.xml
- examples/viewerformat/Simple.xhtml
- examples/interface/createCMSSignature/DataPerReference.Response.xml

#### *Weiterführende Informationen*

- Abschnitt 2.1, „Signatur nach CMS“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 2, „Keyboxen“ in Die österreichische Bürgerkarte - Standardisierte Key- und Infoboxen
- Abschnitt 9, „Anzeigeformate und Zeichensätze“ in Die österreichische Bürgerkarte - Minimale Umsetzung des Security-Layers
- Standard-Anzeigeformat

### 2.1.2. Signatur im Format XMLDSIG erstellen

#### 2.1.2.1. Ein erstes Beispiel

#### *Anfrage*

Dieses Beispiel soll die gleichen Daten wie jenes aus Abschnitt Abschnitt 2.1.1.1, „Ein erstes Beispiel“ signieren, und zwar den Text `Ich bin ein einfacher Text.`. Dieses Mal soll jedoch eine Signatur im Format XMLDSIG erstellt werden. Die passende Anfrage sieht so aus:

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:CreateXMLSignatureRequest
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]   <sl:KeyboxIdentifier>SecureSignatureKeypair</sl:KeyboxIdentifier>
[05]   <sl:DataObjectInfo Structure="enveloping">
[06]     <sl:DataObject>
[07]       <sl:XMLContent>Ich bin ein einfacher Text.</sl:XMLContent>
[08]     </sl:DataObject>
[09]     <sl:TransformsInfo>
[10]       <sl:FinalDataMetaInfo>
[11]         <sl:MimeType>text/plain</sl:MimeType>
[12]       </sl:FinalDataMetaInfo>
[13]     </sl:TransformsInfo>
[14]   </sl:DataObjectInfo>
[15] </sl:CreateXMLSignatureRequest>
```

Zeile 2 enthält den passenden Befehl der Schnittstelle *Security-Layer*.

In Zeile 4 wird mit dem Inhalt des Elements `sl:KeyboxIdentifier` der zu verwendende Signaturschlüssel ausgewählt. Entsprechend der Spezifikation Standardisierte Key- und Infoboxen ist der Wert `SecureSignatureKeypair` anzugeben, wenn eine sichere Signatur bzw. Verwaltungssignatur erstellt werden soll.

Die Zeilen 5 bis 14 enthalten die Angaben zu den Daten, die signiert werden sollen:

In Zeile 5 wird mittels des Attributs `Structure` festgelegt, dass die zu signierenden Daten in die zu erzeugende XML-Signatur aufgenommen werden sollen. Dazu ist der Wert dieses Attributes auf `enveloping` zu setzen.

Die Zeilen 6 bis 8 geben die *Referenz-Eingangsdaten* an. Die Daten werden in diesem ersten Beispiel innerhalb des Containers `sl:XMLContent` angegeben, der eine beliebige Menge von XML-Knoten (Elemente, Text, Kommentare) aufnehmen kann. Im konkreten Fall wird lediglich der zu signierende Textknoten (`Ich bin ein einfacher Text.`) angegeben.

Die Zeilen 9 bis 13 enthalten Informationen darüber, wie die in den Zeilen 6 bis 8 angegebenen *Referenz-Eingangsdaten* in die *Hash-Eingangsdaten* transformiert werden sollen, bevor diese dann tatsächlich signiert werden. Im konkreten Fall sollen die *Referenz-Eingangsdaten* direkt signiert werden, daher fehlen in `sl:TransformsInfo` die Angaben zu den anzuwendenden Transformationen. Lediglich der *Mime Type* der *Referenz-Eingangsdaten* muss festgelegt werden, und zwar in Zeile 11 für den zu signierenden Text mit `text/plain`.

#### *Antwort*

Die entsprechende Antwort der *Bürgerkarten-Umgebung* sollte in etwa wie folgt aussehen. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert und gekürzt wurde, was natürlich die elektronische Signatur bricht.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:CreateXMLSignatureResponse
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]   <dsig:Signature
```

```
[05]     Id="signature-1084461392813"
[06]     xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
[07]     <dsig:SignedInfo>
[08]       <dsig:CanonicalizationMethod
[09]         Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
[10]       <dsig:SignatureMethod
[11]         Algorithm="http://www.buergerkarte.at/namespaces/ecdsa/20020630#"/>
[12]       <dsig:Reference
[13]         Id="reference-0-1084461392813"
[14]         URI="#xpointer(id('signed-data-0-1084461392813')/node())">
[15]         <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[16]         <dsig:DigestValue>7Dp/5KcvUfCnkohkOOzvFaeAIRc=</dsig:DigestValue>
[17]       </dsig:Reference>
[18]       <dsig:Reference
[19]         Type="http://uri.etsi.org/01903/v1.1.1#SignedProperties"
[20]         URI="#xpointer(id('etsi-signed-1084461392813')/*/*)">
[21]         <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[22]         <dsig:DigestValue>uUonRdJqrlWLyWfEVzz7uOXvFD8=</dsig:DigestValue>
[23]       </dsig:Reference>
[24]     </dsig:SignedInfo>
[25]     <dsig:SignatureValue>
[26]       OXxSRlOBBH84Psc3MRgVpEWZgzsAQa4bDz/01RrtoWWH8iJPImco9yn/kFMdfIvO
[27]     </dsig:SignatureValue>
[28]     <dsig:KeyInfo><!-- ... --></dsig:KeyInfo>
[29]     <dsig:Object
[30]       Id="signed-data-0-1084461392813">Ich bin ein einfacher Text.</dsig:Object>
[31]     <dsig:Object Id="etsi-signed-1084461392813"><!-- ... --></dsig:Object>
[32]   </dsig:Signature>
[33] </sl:CreateXMLSignatureResponse>
```

Zeile 2 zeigt die zur Anfrage passende Antwort der *Bürgerkarten-Umgebung*.

Die Zeilen 4 bis 32 enthalten die von der *Bürgerkarten-Umgebung* erzeugte XML-Signatur:

Die Zeilen 12 bis 17 zeigen die `dsig:Reference`, die für die in der Anfrage angegebenen, zu signierenden Daten erzeugt wurde. Man erkennt, dass keine Transformationen in die `dsig:Reference` aufgenommen wurden.

Die Zeilen 25 bis 27 enthalten den eigentlich berechneten Signaturwert.

Die Zeile 30 zeigt einen `dsig:Object` Container, den die Bürgerkarten-Umgebung angelegt hat, um darin die in der Anfrage angegebenen, zu signierenden Daten zu kodieren. Auf diese Daten wird dann aus der `dsig:Reference` heraus durch das Attribut `URI` verwiesen.

#### *Downloads zu diesem Beispiel*

- examples/interface/createXMLSignature/First.xml
- examples/viewerformat/Simple.xhtml
- examples/interface/createXMLSignature/First.Response.xml

#### *Weiterführende Informationen*

- Abschnitt 2.2, „Signatur nach XMLDSIG“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 2, „Keyboxen“ in Die österreichische Bürgerkarte - Standardisierte Key- und Infoboxen
- Abschnitt 9, „Anzeigeformate und Zeichensätze“ in Die österreichische Bürgerkarte - Minimale Umsetzung des Security-Layers

### 2.1.2.2. Möglichkeiten, die zu signierenden Daten anzugeben

Der Schnittstellenbefehl zur Erstellung einer Signatur sieht insgesamt drei verschiedene Arten vor, wie die zu signierenden Daten angegeben werden können:

- direkte Einbettung im Request in base64-kodierter Form (Verwendung des Elements `sl:DataObject/sl:Base64Content`);
- direkte Einbettung im Request als XML-Fragment (Verwendung des Elements `sl:DataObject/sl:XMLContent`);
- Referenzierung mittels URL, die von der *Bürgerkarten-Umgebung* aufgelöst werden kann (Verwendung des Attributs `sl:DataObject/@Reference` im Falle einer *Enveloping Signature* bzw. des Elements `sl:LocRefContent` bei einer *Detached Signature*).

#### *Anfrage*

Die folgende Anfrage zeigt beispielhaft die Verwendung aller drei oben erwähnten Möglichkeiten:

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:CreateXMLSignatureRequest xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[03]   <sl:KeyboxIdentifier>SecureSignatureKeypair</sl:KeyboxIdentifier>
[04]   <sl:DataObjectInfo Structure="detached">
[05]     <sl:DataObject Reference="http://www.example.com/Simple.txt">
[06]       <sl:Base64Content>SWNoIGJpbiBlaW4gZWluZmFjaGVyIFRleHQu</sl:Base64Content>
[07]     </sl:DataObject>
[08]     <sl:TransformsInfo>
[09]       <sl:FinalDataMetaInfo>
[10]         <sl:MimeType>text/plain</sl:MimeType>
[11]       </sl:FinalDataMetaInfo>
[12]     </sl:TransformsInfo>
[13]   </sl:DataObjectInfo>
[14]   <sl:DataObjectInfo Structure="enveloping">
[15]     <sl:DataObject>
[16]       <sl:XMLContent>
[17]         <html xmlns="http://www.w3.org/1999/xhtml">
[18]           <head>
[19]             <title>Ein einfaches SLXHTML-Dokument</title>
[20]             <style type="text/css">p { color: red; }</style>
[21]           </head>
[22]           <body>
[23]             <p>Ich bin ein einfacher Text in rot.</p>
[24]           </body>
[25]         </html>
[26]       </sl:XMLContent>
[27]     </sl:DataObject>
```

```
[28]     <sl:TransformsInfo>
[29]       <sl:FinalDataMetaInfo>
[30]         <sl:MimeType>application/xhtml+xml</sl:MimeType>
[31]       </sl:FinalDataMetaInfo>
[32]     </sl:TransformsInfo>
[33]   </sl:DataObjectInfo>
[34]   <sl:DataObjectInfo Structure="detached">
[35]     <sl:DataObject Reference="http://www.buergerkarte.at/konzept/securitylayer/spezifikation/
[36]                               20040514/tutorial/examples/viewerformat/Simple.xhtml"/>
[37]     <sl:TransformsInfo>
[38]       <sl:FinalDataMetaInfo>
[39]         <sl:MimeType>application/xhtml+xml</sl:MimeType>
[40]       </sl:FinalDataMetaInfo>
[41]     </sl:TransformsInfo>
[42]   </sl:DataObjectInfo>
[43]   <sl:DataObjectInfo Structure="detached">
[44]     <sl:DataObject Reference="http://www.example.com/Simple.xhtml">
[45]       <sl:LocRefContent>http://www.buergerkarte.at/konzept/securitylayer/spezifikation/
[46]         20040514/tutorial/examples/viewerformat/Simple.xhtml</sl:LocRefContent>
[47]     </sl:DataObject>
[48]     <sl:TransformsInfo>
[49]       <sl:FinalDataMetaInfo>
[50]         <sl:MimeType>application/xhtml+xml</sl:MimeType>
[51]       </sl:FinalDataMetaInfo>
[52]     </sl:TransformsInfo>
[53]   </sl:DataObjectInfo>
[54] </sl:CreateXMLSignatureRequest>
```

Mit Hilfe der Zeilen 4 bis 13 werden Daten für eine sogenannte *Detached Signature* angegeben (*Detached Signature* bedeutet hier vereinfacht, dass die signierten Daten nicht im gleichen Dokument kodiert sind wie die XML-Signatur, für mehr Infos siehe Beispiel Abschnitt 2.1.2.4, „Signaturtypen (enveloping, enveloped, detached)“). Für eine *Detached Signature* ist es notwendig, in Zeile 4 das Attribut `Structure` auf den Wert `detached` zu setzen. Weiters ist in Zeile 5 das Attribut `sl:DataObject/@Reference` anzugeben; sein Wert enthält eine URI, so wie sie in `dsig:Reference/@URI` der zu erzeugenden Signatur aufzunehmen ist. In diesem Beispiel ist diese URL jedoch bewusst so gewählt, dass sie von der *Bürgerkarten-Umgebung* nicht aufgelöst werden kann. Damit dieser Umstand nicht zu einem Fehler führt, müssen die Daten zusätzlich explizit angegeben werden: Dazu wird in diesem Beispiel das Element `sl:Base64Content` in Zeile 6 verwendet; es enthält die zu signierenden Daten in base64-kodierter Form. Die *Bürgerkarten-Umgebung* verhält sich nun so, dass sie die in `sl:DataObject/@Reference` angegebene URL nicht auflöst, sondern stattdessen die explizit in `sl:Base64Content` angegebenen Daten verwendet. Die erzeugte Signatur sieht aber so aus, als hätte die *Bürgerkarten-Umgebung* tatsächlich die URL aufgelöst. Sinnvoll ist diese Vorgangsweise beispielsweise in Fällen, in denen eine URL beim Erzeugen der Signatur nicht verfügbar ist, jedoch sehrwohl beim zukünftigen Prüfen. Die Zeilen 8 bis 12 enthalten Metadaten zu den zu signierenden Daten, vergleiche dazu Abschnitt 2.1.2.1, „Ein erstes Beispiel“).

Die Zeilen 14 bis 33 enthalten Angaben zu weiteren Daten, die signiert werden sollen. Es soll dazu wie in Abschnitt 2.1.2.1, „Ein erstes Beispiel“ eine *Enveloping Signature* (vergleiche den Wert des Attributs `Structure` in Zeile 14) angefertigt werden, das bedeutet, dass die zu signierenden Daten als Teil der XML-Struktur der Signatur kodiert werden. Die zu signierenden Daten, ein einfaches XHTML-Dokument, werden in den Zeilen 16 bis 26 explizit als XML-Daten als Inhalt des Elements `sl:XMLContent` angegeben. Das Attribut `sl:DataObject/@Reference` darf hier nicht angegeben werden, da die Daten ja in die XML-Struktur der Signatur aufgenommen, und nicht wie oben per URL referenziert werden. Die Zeilen 28 bis 32 enthalten Metadaten zu den zu signierenden Daten, vergleiche dazu Abschnitt 2.1.2.1, „Ein erstes Beispiel“).

Die dritten Daten, die signiert werden sollen, werden mit Hilfe der Zeilen 34 bis 42 spezifiziert. Es soll wie für die ersten Daten eine *Detached Signature* erstellt werden (vergleiche den Wert des Attributs `Structure` in Zeile 34). Im Unterschied zu den ersten Daten werden die Daten hier direkt per auflösbarer URL (angegeben im Attribut `sl:DataObject/@Reference` in Zeile 34 bis 35) referenziert. Die *Bürgerkarten-Umgebung* kann von dort die zu signierenden Daten beziehen; eine explizite Angabe der Daten in der Anfrage unterbleibt daher. Die Zeilen 37 bis 41 enthalten Metadaten zu den zu signierenden Daten, vergleiche dazu Abschnitt 2.1.2.1, „Ein erstes Beispiel“).

Schließlich sollen mit Hilfe der Zeilen 43 bis 53 noch weitere Daten als *Detached Signature* unterschrieben werden. Das Attribut `Structure` in Zeile 43 wird daher wieder auf den Wert `detached` gesetzt. In Zeile 44 ist das Attribut `sl:DataObject/@Reference` anzugeben; sein Wert enthält eine URI, so wie sie in `dsig:Reference/@URI` der zu erzeugenden Signatur aufzunehmen ist. Wie auch bei den zweiten Daten wurde eine URL gewählt, die von der *Bürgerkarten-Umgebung* nicht aufgelöst werden kann. Es muss daher in der Anfrage zusätzlich eine Quelle spezifiziert werden, von der die *Bürgerkarten-Umgebung* die Daten beziehen kann. Im Gegensatz zu den zweiten Daten wird nicht die explizite Kodierung in der Anfrage gewählt, sondern es wird mittels des Elements `sl:LocRefContent` in Zeile 44 bis 45 eine URL angegeben, welche die *Bürgerkarten-Umgebung* auflösen kann. Die Anwendungsfälle für diese Variante sind die gleichen wie für die ersten Daten. Die Zeilen 48 bis 52 enthalten Metadaten zu den zu signierenden Daten, vergleiche dazu Abschnitt 2.1.2.1, „Ein erstes Beispiel“).

#### *Antwort*

Die entsprechende Antwort der *Bürgerkarten-Umgebung* sollte in etwa wie folgt aussehen. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert und gekürzt wurde, was natürlich die elektronische Signatur bricht.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:CreateXMLSignatureResponse xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[03]   <dsig:Signature Id="signature-1084532870164" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
[04]     <dsig:SignedInfo>
[05]       <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
[06]       <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
[07]       <dsig:Reference Id="reference-0-1084532870164" URI="http://www.example.com/Simple.txt">
[08]         <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[09]         <dsig:DigestValue>7Dp/5KcvUfCnkohkOOzvFaeAIRc=</dsig:DigestValue>
[10]       </dsig:Reference>
[11]       <dsig:Reference Id="reference-1-1084532870164"
[12]         URI="#xpointer(id('signed-data-1-1084532870164')/node())">
[13]         <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[14]         <dsig:DigestValue>W9Gsw+M74YGV2iUhMG0cvLQk0NA=</dsig:DigestValue>
[15]       </dsig:Reference>
[16]       <dsig:Reference Id="reference-2-1084532870164"
[17]         URI="http://www.buergerkarte.at/konzept/securitylayer/spezifikation/20040514/tutorial/
[18]              examples/viewerformat/Simple.xhtml">
[19]         <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[20]         <dsig:DigestValue>DW8CLYLic6czUS32ZCDyyc1akdY=</dsig:DigestValue>
[21]       </dsig:Reference>
[22]       <dsig:Reference Id="reference-3-1084532870164" URI="http://www.example.com/Simple.xhtml">
[23]         <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[24]         <dsig:DigestValue>DW8CLYLic6czUS32ZCDyyc1akdY=</dsig:DigestValue>
[25]       </dsig:Reference>
[26]       <dsig:Reference Type="http://uri.etsi.org/01903/v1.1.1#SignedProperties"
[27]         URI="#xpointer(id('etsi-signed-1084532870164')/*/*)">
[28]         <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[29]         <dsig:DigestValue>fcz4azQhn5Zo7E7h/h/8W/oRDe0=</dsig:DigestValue>
[30]       </dsig:Reference>
```

```
[31]     </dsig:SignedInfo>
[32]     <dsig:SignatureValue><!-- ... --></dsig:SignatureValue>
[33]     <dsig:KeyInfo><!-- ... --></dsig:KeyInfo>
[34]     <dsig:Object Id="signed-data-1-1084532870164">
[35]       <html xmlns="http://www.w3.org/1999/xhtml">
[36]         <head>
[37]           <title>Ein einfaches SLXHTML-Dokument</title>
[38]           <style type="text/css">p { color: red; }</style>
[39]         </head>
[40]         <body>
[41]           <p>Ich bin ein einfacher Text in rot.</p>
[42]         </body>
[43]       </html>
[44]     </dsig:Object>
[45]     <dsig:Object Id="etsi-signed-1084532870164"><!-- ... --></dsig:Object>
[46]   </dsig:Signature>
[47] </sl:CreateXMLSignatureResponse>
```

Zeile 1 zeigt die zur Anfrage passende Antwort der *Bürgerkarten-Umgebung*.

Die Zeilen 2 bis 47 enthalten die von der *Bürgerkarten-Umgebung* erzeugte XML-Signatur:

Die Zeilen 7 bis 10 zeigen jene `dsig:Reference`, die von der *Bürgerkarten-Umgebung* für die ersten zu signierenden Daten erzeugt worden ist. Man erkennt, dass es sich dabei um eine *Detached Signature* handelt, denn das Attribut `URI` enthält eine Referenz aus dem Signaturdokument hinaus. Es handelt sich dabei um die in der Anfrage angegebene, nicht auflösbare URL `http://www.example.com/Simple.txt`.

Die Zeilen 11 bis 15 stellen jene `dsig:Reference` dar, die für die zweiten zu signierenden Daten von der *Bürgerkarten-Umgebung* erstellt wurde. Wie in der Anfrage festgelegt, wurde eine *Enveloping Signature* erzeugt: Das Attribut `URI` enthält eine interne Referenz (`#xpointer(id('signed-data-1-1084532870164')/node())`), die auf die entsprechend in die XML-Struktur der Signatur integrierten Daten (vergleiche die Zeilen 35 bis 43) verweist.

Die Zeilen 16 bis 21 repräsentieren die den dritten zu signierenden Daten entsprechende `dsig:Reference`. Erzeugt wurde hier gemäß der Anfrage eine *Detached Signature* über jene Daten, die von der *Bürgerkarten-Umgebung* von der in der Anfrage spezifizierten URL bezogen wurden. Man erkennt in den Zeilen 17 bis 18, dass spezifizierte URL auch in das Attribut `URI` der `dsig:Reference` übernommen wurde.

Die Zeilen 22 bis 25 schließlich zeigen jene `dsig:Reference`, welche den vierten in der Angabe spezifizierten Daten entspricht. Die Daten wurden von der *Bürgerkarten-Umgebung* von der im Element `sl:LocRefContent` der Anfrage spezifizierten URL aufgelöst, als Wert des Attributs `URI` der `dsig:Reference` in Zeile 22 wurde jedoch der Wert aus dem Attribut `sl:DataObject/@Reference` der Anfrage (vergleiche Zeile 44 der Anfrage) übernommen.

#### *Downloads zu diesem Beispiel*

- examples/interface/createXMLSignature/SpecifyData.xml
- examples/interface/createXMLSignature/SpecifyData.Response.xml

#### *Weiterführende Informationen*

- Abschnitt 2.2, „Signatur nach XMLDSIG“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 2, „Keyboxen“ in Die österreichische Bürgerkarte - Standardisierte Key- und Infoboxen
- Abschnitt 9, „Anzeigeformate und Zeichensätze“ in Die österreichische Bürgerkarte - Minimale Umsetzung des Security-Layers
- Standard-Anzeigeformat

### 2.1.2.3. Möglichkeiten, die zu signierenden Daten anzugeben

#### *Anfrage*

Dieses Beispiel stellt die wichtigsten Transformationen vor, die bei der Erstellung einer XML-Signatur verwendet werden können. Eine Transformation bzw. eine Kette mehrerer hintereinander geschalteter Transformationen werden auf die Referenz-Eingangsdaten (also jene Daten, die in `sl:DataObjectInfo/sl:DataObject` angegeben werden) angewendet; über das Ergebnis der letzten Transformation wird dann der Hashwert berechnet.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:CreateXMLSignatureRequest
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"
[04]   xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
[05]   <sl:KeyboxIdentifier>SecureSignatureKeypair</sl:KeyboxIdentifier>
```

Zeile 5 enthält die Angabe, dass eine sichere Signatur bzw. Verwaltungssignatur erstellt werden soll (`SecureSignatureKeypair`).

```
[06]   <sl:DataObjectInfo Structure="detached">
[07]     <sl:DataObject
[08]       Reference="http://www.buergerkarte.at/konzept/securitylayer/\
[09]         spezifikation/20040514/tutorial/examples/interface/common/SimpleText.txt.b64"/>
[10]     <sl:TransformsInfo>
[11]       <dsig:Transforms>
[12]         <dsig:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#base64"/>
[13]       </dsig:Transforms>
[14]       <sl:FinalDataMetaInfo>
[15]         <sl:MimeType>text/plain</sl:MimeType>
[16]       </sl:FinalDataMetaInfo>
[17]     </sl:TransformsInfo>
[18]   </sl:DataObjectInfo>
```

Für das erste zu signierende Datenobjekt werden die Referenz-Eingangsdaten mittels `DataObject/@Reference` referenziert (Zeile 7 - 9), d. h. die *Bürgerkarten-Umgebung* löst die darin enthaltene URL auf, um zu den Daten zu gelangen. Es handelt sich dabei um einen base64 kodierten Text.

Unterschrieben werden soll nun aber nicht dieser base64-kodierte Text, sondern der entsprechend dekodierte Text. Dies lässt sich elegant durch die Angabe einer *Base64 Decoding* Transformation bewerkstelligen. Dazu wird als erstes Kindelement von `CreateTransformsInfo` ein `dsig:Transforms` Element angegeben (Zeile 11 - 13). Dieses `dsig:Transforms` Element nimmt ein oderer mehrere `dsig:Transform` Elemente auf, wobei jedes `dsig:Transform` Element für eine Transformation steht. In unserem Fall wird nur eine einzige Transformation benötigt; die Angabe, um welche Transformation es sich handelt, wird durch das Attribut `dsig:Transform/@Algorithm` angegeben. Für die *Base64 Decoding* Transformation muss der Wert auf `http://www.w3.org/2000/09/xmldsig#base64` gesetzt werden. Sie ist eine parameterlose Transformation, d. h. `dsig:Transform` hat keine Kindelemente.

Der *Mime Type* der zu signierenden Daten wird als `text/plain` angegeben, da ja tatsächlich nach der durchgeführten Transformation dekodierter Text vorliegt, über den dann der Hashwert berechnet wird.

```
[19]   <sl:DataObjectInfo Structure="detached">
[20]     <sl:DataObject
[21]       Reference="http://www.buergerkarte.at/konzept/securitylayer/\
[22]         spezifikation/20040514/tutorial/examples/interface/common/XMLDocument.xml"/>
[23]     <sl:TransformsInfo>
[24]       <dsig:Transforms>
[25]         <dsig:Transform Algorithm="http://www.w3.org/2002/06/xmldsig-filter2">
[26]           <xp2:XPath xmlns:xp2="http://www.w3.org/2002/06/xmldsig-filter2"
[27]             xmlns:doc="urn:document" Filter="subtract">/doc:XMLDocument/doc:Paragraph[2]
[28]           </xp2:XPath>
[29]         </dsig:Transform>
[30]         <dsig:Transform Algorithm="http://www.w3.org/TR/1999/REC-xslt-19991116">
[31] <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
[32] xmlns:doc="urn:document">
[33] <xsl:output encoding="UTF-8" method="xml" indent="yes"/>
[34] <xsl:template match="/doc:XMLDocument">
[35]   <html xmlns="http://www.w3.org/1999/xhtml">
[36]     <head>
[37]       <title>HTML-Dokument</title>
[38]     </head>
[39]     <body>
[40]       <xsl:for-each select="doc:Paragraph">
[41]         <p>
[42]           <xsl:value-of select="child::text()"/>
[43]         </p>
[44]       </xsl:for-each>
[45]     </body>
[46]   </html>
[47] </xsl:template>
[48] </xsl:stylesheet></dsig:Transform>
[49]         <dsig:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
[50]       </dsig:Transforms>
[51]       <sl:FinalDataMetaInfo>
[52]         <sl:MimeType>application/xhtml+xml</sl:MimeType>
[53]       </sl:FinalDataMetaInfo>
[54]     </sl:TransformsInfo>
[55]   </sl:DataObjectInfo>
[56] </sl:CreateXMLSignatureRequest>
```

Für das zweite zu signierende Datenobjekt werden die Referenz-Eingangsdaten wiederum mittels `DataObject/@Reference` referenziert (Zeile 20 - 22), d. h. die *Bürgerkarten-Umgebung* löst die darin enthaltene URL auf, um zu den Daten zu gelangen. Es handelt sich dabei um ein XML-Dokument.

Zunächst soll von diesem XML-Dokument jedoch ein Teil weggeschnitten werden, da er nicht mitsigniert werden soll. Für diesen Zweck bietet sich die XPath Filter 2 Transformation an (Zeile 25 - 29). Das Attribut `dsig:Transform/@Algorithm` ist dazu auf den Wert `http://www.w3.org/2002/06/xmldsig-filter2` zu setzen. Diese Transformation benötigt weiters Transformationsparameter. Diese werden als Kindelement `xp2:XPath` in `dsig:Transform` angegeben. Das Attribut `Filter` selektiert den Filtermodus; für das Bespiel wird den Modus `subtract` benötigt, da ein Teil weggefiltert werden soll. Der Textinhalt von `xp2:XPath` ist ein XPath-Ausdruck, der den Wurzelknoten jenes Teilbaums selektiert, der weggefiltert werden soll. Für das Beispiel soll das zweite `doc:Paragraph` Element des XML-Dokuments weggefiltert werden. Beachten Sie, dass das im XPath-Ausdruck verwendete Namespace-Prefix `doc` im Kontext des `xp2:XPath` Elements deklariert sein muss.

Als nächstes soll nun das XML-Dokument mit Hilfe eines Stylesheets in ein XHTML-Dokument übergeführt werden. Dazu kann die XSLT Transformation verwendet werden (Zeile 30 - 48). Das Attribut `dsig:Transform/@Algorithm` ist dazu auf den Wert `http://www.w3.org/TR/1999/REC-xslt-19991116` zu setzen. Auch diese Transformation benötigt Transformationsparameter: Als Kindelement von `dsig:Transform` wird jener Stylesheet angegeben, mit dem die Stylesheet-Transformation ausgeführt werden soll.

Abschließend soll, wie in der Spezifikation der *XSLT-Transformation* empfohlen, eine Kanonisierungstransformation angewendet werden. Damit können Unterschiede im Output unterschiedlicher XSLT-Engines, wie sie in der Praxis vorkommen, abgefangen werden. Beachten Sie, dass als Voraussetzung dazu die Output-Methode im Stylesheet auf `xml` festgelegt werden muss (`<xsl:output method="xml">`), denn nur XML-Output kann anschließend kanonisiert werden. Das Attribut `dsig:Transform/@Algorithm` ist für die *Canonical XML Transformation* auf den Wert `http://www.w3.org/TR/2001/REC-xml-c14n-20010315` zu setzen. Die Transformation benötigt keine Transformationsparameter.

Das Ergebnis der drei hintereinandergeschalteten Transformationen, welches der Hashwert-Berechnung zufließt, finden Sie in den Downloads zu diesem Beispiel.

### *Antwort*

Die entsprechende Antwort der *Bürgerkarten-Umgebung* sollte in etwa wie folgt aussehen. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert und gekürzt wurde, was natürlich die elektronische Signatur bricht.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:CreateXMLSignatureResponse
[03]   xmlns:sl11="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]   <dsig:Signature Id="signature-0912200410273681"
[05]     xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
[06]     <dsig:SignedInfo>
[07]       <dsig:CanonicalizationMethod
[08]         Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments"/>
[09]       <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
```

Die Antwort enthält in `sl:CreateXMLSignatureResponse` das Ergebnis der Signaturerstellung. Nachdem die Signatur in kein bestehendes Dokument eingefügt werden sollte, enhält das Element direkt die erzeugte XML-Signatur (`dsig:Signature`).

```
[10]       <dsig:Reference Id="reference-0-0912200410273681"
[11]         URI="http://www.buergerkarte.at/konzept/securitylayer/spezifikation/20040514/\
[12]         tutorial/examples/interface/common/SimpleText.txt.b64">
[13]         <dsig:Transforms>
[14]           <dsig:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#base64"/>
[15]         </dsig:Transforms>
[16]         <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[17]         <dsig:DigestValue>7Dp/5KcvUfCnkohkOOzvFaeAIRc=</dsig:DigestValue>
[18]       </dsig:Reference>
```

Die erste `dsig:Reference` (Zeile 10 - 18) wurde auf Grund des ersten `DataObjectInfo` im Request erstellt. Man erkennt, dass die URL auf die Referenz-Eingangsdaten (Wert des Attributs `dsig:Reference/@URI`) aus `DataObject/@Reference` übernommen und eine *Base64 Decoding* Transformation eingefügt würde. Die im Request spezifizierten Transformationen werden also eins zu eins in die XML-Signatur übernommen.

```
[19]       <dsig:Reference Id="reference-1-0912200410273681"
[20]         URI="http://www.buergerkarte.at/konzept/securitylayer/spezifikation/20040514/\
```

```
[21]          tutorial/examples/interface/common/XMLDocument.xml">
[22]          <dsig:Transforms>
[23]            <dsig:Transform Algorithm="http://www.w3.org/2002/06/xmldsig-filter2">
[24]              <xpf:XPath Filter="subtract" xmlns:doc="urn:document"
[25]                xmlns:xpf="http://www.w3.org/2002/06/xmldsig-filter2">
[26]                /doc:XMLDocument/doc:Paragraph[2]
[27]              </xpf:XPath>
[28]            </dsig:Transform>
[29]            <dsig:Transform Algorithm="http://www.w3.org/TR/1999/REC-xslt-19991116">
[30]              <xsl:stylesheet version="1.0"
[31]                xmlns:doc="urn:document" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
[32]                <xsl:output encoding="UTF-8" indent="yes" method="xml"/>
[33]                <xsl:template match="/doc:XMLDocument"><!--...--></xsl:template>
[34]              </xsl:stylesheet>
[35]            </dsig:Transform>
[36]            <dsig:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
[37]          </dsig:Transforms>
[38]          <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[39]          <dsig:DigestValue>fIPwneCpjVqTXwHMN9DFfx6tJIU=</dsig:DigestValue>
[40]        </dsig:Reference>
```

Die zweite `dsig:Reference` (Zeile 19 - 40) wurde auf Grund des zweiten `DataObjectInfo` im Request erstellt. Man erkennt auch hier gut, dass die URL auf die Referenz-Eingangsdaten (Wert des Attributs `dsig:Reference/@URI`) aus `DataObject/@Reference` übernommen und die drei Transformationen wie im Request angegeben eingefügt wurden.

```
[41]        <dsig:Reference Id="etsi-its-0-0912200410273681"
[42]          Type="http://uri.etsi.org/01903/v1.1.1#SignedProperties" URI="">
[43]          <!--...-->
[44]        </dsig:Reference>
[45]      </dsig:SignedInfo>
[46]      <dsig:SignatureValue><!--...--></dsig:SignatureValue>
[47]      <dsig:KeyInfo><!--...--></dsig:KeyInfo>
[48]      <dsig:Object Id="etsi-signed-2-0912200410273681"
[49]        xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
[50]        <!--...-->
[51]      </dsig:Object>
[52]    </dsig:Signature>
[53] </sl:CreateXMLSignatureResponse>
```

Es folgt eine weitere Referenz auf die Signatureigenschaften (Zeile 41 - 44), sowie Signaturwert (Zeile 46), Schlüsselinformationen (Zeile 47) und Signatureigenschaften (Zeile 48 - 52).

#### *Downloads zu diesem Beispiel*

- examples/interface/createXMLSignature/Transforms.xml
- examples/interface/common/SimpleText.txt.b64
- examples/interface/common/XMLDocument.xml
- examples/interface/createXMLSignature/Transforms.Hashinput2.xhtml
- examples/interface/createXMLSignature/Transforms.Response.xml

#### *Weiterführende Informationen*

- Abschnitt 2.2, „Signatur nach XMLDSIG“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 2, „Keyboxen“ in Die österreichische Bürgerkarte - Standardisierte Key- und Infoboxen
- Abschnitt 9, „Anzeigeformate und Zeichensätze“ in Die österreichische Bürgerkarte - Minimale Umsetzung des Security-Layers
- Standard-Anzeigeformat
- Abschnitt 5.1.4, „Transformationsalgorithmen“ in Die österreichische Bürgerkarte - Minimale Umsetzung des Security-Layers

### 2.1.2.4. Signaturtypen (enveloping, enveloped, detached)

Der Standard für XML-Signaturen spricht, je nach dem in welchem örtlichen Verhältnis die XML-Signaturstruktur (`dsig:Signature`) und die signierten Daten zueinander stehen, von einer *Enveloping*, *Enveloped* oder *Detached Signature*:

- *Enveloping Signature*: Die XML-Signatur umschließt die signierten Daten.
- *Enveloped Signature*: Die XML-Signatur wird von den signierten Daten umschlossen.
- *Detached Signature*: Die XML-Signature und die signierten Daten sind voneinander unabhängig.

Betrachtet man diese Definition genauer, erkennt man, das sie eigentlich zu kurz greift. Eine XML-Signatur kann ja bekanntlich mehrere Dateneinheiten auf einmal signieren; es ist daher durchaus möglich, dass man ein und dieselbe Signatur mehreren der oben definierten Klassen zuordnen kann. Das folgende Beispiel erzeugt eine XML-Signatur, die insgesamt vier Dateneinheiten signiert; für die erste Dateneinheit entspricht sie einer *Enveloping Signature*, für die zweite Dateneinheit einer *Enveloped Singnature*, für die dritte und vierte Dateneinheit einer *Detached Signature*.

#### *Anfrage*

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:CreateXMLSignatureRequest
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"
[04]   xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
[05]   <sl:KeyboxIdentifier>CertifiedKeypair</sl:KeyboxIdentifier>
```

Zeile 5 enthält die Angabe, dass eine Signatur mit dem kombinierten Signatur- und Verschlüsselungsschlüssel erstellt werden soll (`CertifiedKeypair`).

```
[06]   <sl:DataObjectInfo Structure="enveloping">
[07]     <sl:DataObject>
[08]       <sl:XMLContent>Von der Signatur umschlossene Daten.</sl:XMLContent>
[09]     </sl:DataObject>
[10]     <sl:TransformsInfo>
[11]       <sl:FinalDataMetaInfo>
[12]         <sl:MimeType>text/plain</sl:MimeType>
[13]       </sl:FinalDataMetaInfo>
[14]     </sl:TransformsInfo>
```

```
[15]   </sl:DataObjectInfo>
```

Für die erste Dateneinheit (Zeile 6 - 15) wird die resultierende XML-Signatur einer *Enveloping Signature* entsprechen, d. h. die zu signierenden Daten werden von der XML-Signatur umschlossen. Genauer gesagt, werden sie in ein `dsig:Object` als Kind von `dsig:Signature` eingebettet werden.

Dafür ist zunächst in Zeile 6 das Attribut `Structure` mit dem Wert `enveloping` zu belegen. Die in das `dsig:Object` einzubettenden Daten, in diesem Fall eine einfache Zeichenkette, werden in Zeile 8 als Inhalt von `sl:XMLContent` angegeben.

```
[16]   <sl:DataObjectInfo Structure="detached">
[17]     <sl:DataObject Reference=""/>
[18]     <sl:TransformsInfo>
[19]       <dsig:Transforms>
[20]         <dsig:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
[21]       </dsig:Transforms>
[22]       <sl:FinalDataMetaInfo>
[23]         <sl:MimeType>application/xml</sl:MimeType>
[24]       </sl:FinalDataMetaInfo>
[25]     </sl:TransformsInfo>
[26]   </sl:DataObjectInfo>
```

Für die zweite Dateneinheit (Zeile 16 - 26) wird die resultierende XML-Signatur einer *Enveloping Signature* entsprechen, d. h. die XML-Signatur wird von den zu signierenden Daten umschlossen. Das mag zunächst nach einem Widerspruch klingen, denn die XML-Signatur würde ja dann versuchen, sich selbst zu signieren, was nicht funktionieren kann. Wie das trotzdem funktioniert, wird etwas weiter unten erläutert.

Um eine *Enveloping Signature* zu erstellen, muss zunächst in Zeile 16 das Attribut `Structure` auf den Wert `detached` gesetzt werden.

> **Anmerkung**
>
> Das ist kein Druckfehler, sondern hat vielmehr historische Gründe; wenn eine Enveloping Signature erzeugt werden soll, ist trotzdem der Wert `detached` zu verwenden.

Unterzeichnet werden soll das gesamte XML-Dokument, in das die zu erstellende Signatur eingefügt werden soll (vergleiche weiter unten die Zeilen 52 - 56). Dazu ist der Wert des Attributs `Reference` in Zeile 17 mit dem leeren String zu belegen.

Damit sich die Signatur nicht selbst unterschreibt, muss sie aus den zu signierenden Daten mit Hilfe einer Transformation herausgeschnitten werden. Dazu dient die speziell dafür erfundene *Enveloping Signature* Transformation (vergleiche Zeile 20); diese Transformation filtert aus den Transformationseingangsdaten genau das `dsig:Signature` Element heraus (*Anmerkung: Man könnte für diesen Zweck natürlich genau so gut selbst eine passende* XPath Filter 2 *Transformation definieren, die vordefinierte* Enveloping Signature *Transformation erleichtert nur die Arbeit*).

Nachdem die resultierenden Daten eine XML-Struktur sind, wird in Zeile 23 der *Mime Type* dementsprechend auf `application/xml` gesetzt.

```
[27]   <sl:DataObjectInfo Structure="detached">
[28]     <sl:DataObject Reference="http://www.buergerkarte.at/konzept/securitylayer/\
[29]       spezifikation/20040514/tutorial/examples/interface/common/SimpleText.txt"/>
[30]     <sl:TransformsInfo>
[31]       <sl:FinalDataMetaInfo>
[32]         <sl:MimeType>text/plain</sl:MimeType>
[33]       </sl:FinalDataMetaInfo>
[34]     </sl:TransformsInfo>
[35]   </sl:DataObjectInfo>
```

Für die dritte Dateneinheit (Zeile 27 - 35) wird die resultierende XML-Signatur einer *Detached Signature* entsprechen, d. h. die XML-Signatur und die signierten Daten stehen in keinem besonderen örtlichen Verhältnis zu einander (oder anders ausgedrückt, sind von einander getrennt und unabhängig).

Zunächst ist in Zeile 27 der Wert des Attributs `Structure` mit `detached` zu belegen.

> **Anmerkung**
>
> Diesmal ergibt das wieder Sinn ;-)

Unterzeichnet werden soll ein externes Textdokument, auf das mit Hilfe des Attributs Reference in Zeile 28 verwiesen wird. Die *Bürgerkarten-Umgebung* wird diese Referenz auflösen, um so zu den zu signierenden Daten zu gelangen; andererseits wird der Wert des Attributs auch im `dsig:Reference/@URI` Attribut der XML-Signatur verwendet werden.

Der *Mime Type* in Zeile 32 wird wiederum passend auf `text/plain` gesetzt.

```
[36]   <sl:DataObjectInfo Structure="detached">
[37]     <sl:DataObject Reference=""/>
[38]     <sl:TransformsInfo>
[39]       <dsig:Transforms>
[40]         <dsig:Transform Algorithm="http://www.w3.org/2002/06/xmldsig-filter2">
[41]           <xp2:XPath xmlns:xp2="http://www.w3.org/2002/06/xmldsig-filter2"
[42]           xmlns:doc="urn:document" Filter="intersect">/doc:Whole/doc:Part2</xp2:Xpath>
[43]         </dsig:Transform>
[44]       </dsig:Transforms>
[45]       <sl:FinalDataMetaInfo>
[46]         <sl:MimeType>application/xml</sl:MimeType>
[47]       </sl:FinalDataMetaInfo>
[48]     </sl:TransformsInfo>
[49]   </sl:DataObjectInfo>
```

Für die vierte Dateneinheit (Zeile 36 - 49) wird die resultierende XML-Signatur ebenfalls einer *Detached Signature* entsprechen, d. h. die XML-Signatur und die signierten Daten stehen auch hier in keinem besonderen örtlichen Verhältnis zu einander. Das stimmt zwar hier ein bisschen weniger als für die dritte Dateneinheit, wesentlich ist es jedoch, dass weder die Signatur die Daten noch die Daten die Sigantur beinhaltet: Signiert wird in diesem Fall zwar ein Teil jenes XML-Dokuments, in das die zu erstellende Signatur integriert werden soll (vergleiche weiter unten die Zeilen 52 - 56), aber eben nur ein Teil, und dieser Teil beinhaltet nicht die XML-Signatur.

Konkret ist in Zeile 36 wiederum das Attribut `Structure` mit dem Wert `detached` zu belegen. In Zeile 37 wird zunächst einmal ähnlich wie bei der zweiten Dateneinheit das gesamte Dokument selektiert, in das die Signatur eingefügt werden wird. In Zeile 40 - 43 wird jedoch mit Hilfe einer *XPath Filter 2* Transformation ein Großteil des Dokuments verworfen; übrig bleibt nur der Teilbaum des Dokuments, der vom Element `/doc:Whole/doc:Part2` aufgespannt wird.

Der übrig gebliebene Teilbaum ist eine XML-Struktur, daher wird der *Mime Type* hier auf `application/xml` gesetzt.

```
[50]   <sl:SignatureInfo>
[51]     <sl:SignatureEnvironment>
[52]       <sl:XMLContent>
[53]         <doc:Whole xmlns:doc="urn:document">
[54]           <doc:Part1>Text in Teil 1</doc:Part1>
[55]           <doc:Part2>Text in Teil 2</doc:Part2>
[56]         </doc:Whole>
```

```
[57]       </sl:XMLContent>
[58]     </sl:SignatureEnvironment>
[59]     <sl:SignatureLocation
[60]       xmlns:doc="urn:document" Index="4">/doc:Whole</sl:SignatureLocation>
[61]   </sl:SignatureInfo>
[62] </sl:CreateXMLSignatureRequest>
```

Das Element `sl:SignatureInfo` in Zeile 52 - 61 muss dann angegeben werden, wenn - wie hier der Fall - die XML-Signatur in ein bereits bestehendes XML-Dokument eingebettet werden soll.

`sl:SignatureEnvironment` (Zeile 51 - 58) enhält genau dieses bestehende XML-Dokument. Grundsätzlich stehen dafür die gleichen Methoden wie für die Angabe von Datenobjekten (vergleiche Abschnitt 2.1.2.2, „Möglichkeiten, die zu signierenden Daten anzugeben“) zur Verfügung. Hier wird das Dokument direkt in `sl:XMLContent` als XML-Struktur angegeben (Zeile 54 - 55).

Die Position, an der die XML-Signatur in das bestehende XML-Dokument eingefügt werden soll, wird mit Hilfe des Elements `sl:SignatureLocation` (Zeile 59 - 61) angegeben. Der Textinhalt dieses Elements besteht dabei aus einem XPath-Ausdruck, der - angewendet auf den Root-Knoten des XML-Dokuments - jenes Element selektiert, als dessen Kind das `dsig:Signature` Element aufgenommen werden soll.

> **Anmerkung**
>
> Das im XPath-Ausdruck verwendete Namenraum-Präfix `doc` muss im Kontext des Elements `sl:SignatureLocation` bekannt sein; deshalb findet sich in Zeile 60 auch die entsprechende Namenraum-Deklaration.

Das Attribut `Index` gibt den Offset für `dsig:Signature` innerhalb dieses Elements an, wobei mit `0` zu zählen begonnen wird.

> **Anmerkung**
>
> Der Wert `4` für dieses Beispiel mag auf den ersten Blick irritieren, da die Signatur unmittelbar nach `doc:Part2` eingefügt werden soll; es müssen jedoch auch die durch die Einrückung entstandenen Whitespace-Textknoten vor `doc:Part1` und `doc:Part2` berücksichtigt werden.

### *Antwort*

Die entsprechende Antwort der *Bürgerkarten-Umgebung* sollte in etwa wie folgt aussehen. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert und gekürzt wurde, was natürlich die elektronische Signatur bricht.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:CreateXMLSignatureResponse
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]   <doc:Whole xmlns:doc="urn:document">
[05]     <doc:Part1>Text in Teil 1</doc:Part1>
[06]     <doc:Part2>Text in Teil 2</doc:Part2>
[07]     <dsig:Signature Id="signature-09122004155437545"
[08]       xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
[09]       <dsig:SignedInfo>
[10]         <dsig:CanonicalizationMethod
[11]           Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments"/>
[12]         <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
```

Man erkennt, dass das Antwortelement `sl:CreateXMLSignatureResponse` dieses Mal nicht direkt ein `dsig:Signature` Element beinhaltet, sondern vielmehr das in der Anfrage angegebene XML-Dokument (Zeile 5ff), wobei die XML-Signatur an der in der Anfrage spezifizierten Stelle eingefügt worden ist (unmittelbar nach `doc:Part2`, vergleiche Zeile 6 - 7).

```
[13]         <dsig:Reference Id="reference-0-09122004155437545"
[14]           URI="#signed-data-0-09122004155437545">
[15]           <dsig:Transforms>
[16]             <dsig:Transform Algorithm="http://www.w3.org/2002/06/xmldsig-filter2">
[17]               <xpf:XPath Filter="intersect"
[18]                 xmlns:xpf="http://www.w3.org/2002/06/xmldsig-filter2">
[19]                 //*[@Id='signed-data-0-09122004155437545']/node()</xpf:XPath>
[20]             </dsig:Transform>
[21]           </dsig:Transforms>
[22]           <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[23]           <dsig:DigestValue>G+UQZGU9uY8B7rW0yGvCLpo55ek=</dsig:DigestValue>
[24]         </dsig:Reference>
```

Die erste `dsig:Reference` (Zeile 13 - 24) entspricht der ersten Dateneinheit aus der Anfrage (siehe dort, Zeile 6 - 15). Das Attribut `URI` in Zeile 14 verweist mittels *ID-Reference* auf das `dsig:Object`, in das der in der Anfrage angegebene Text eingebettet wurde. Zusätzlich wurde von der *Bürgerkarten-Umgebung* eine *XPath Filter 2* Transformation selbständig eingefügt (Zeile 16 - 20), damit nicht das gesamte `dsig:Object` (siehe Zeile 62 - 63 weiter unten), sondern lediglich der darin enthaltene Text (siehe Zeile 62 weiter unten) tatsächlich signiert wird.

```
[25]         <dsig:Reference Id="reference-1-09122004155437545" URI="">
[26]           <dsig:Transforms>
[27]             <dsig:Transform
[28]               Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
[29]           </dsig:Transforms>
[30]           <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[31]           <dsig:DigestValue>dZoplbJsn4ooI/7oC8733wwFvTY=</dsig:DigestValue>
[32]         </dsig:Reference>
```

Die zweite `dsig:Reference` (Zeile 25 - 32) entspricht der zweiten Dateneinheit aus der Anfrage (siehe dort, Zeile 16 - 26).

```
[33]         <dsig:Reference Id="reference-2-09122004155437545"
[34]           URI="http://www.buergerkarte.at/konzept/securitylayer/spezifikation/\
[35]           20040514/tutorial/examples/interface/common/SimpleText.txt">
[36]           <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[37]           <dsig:DigestValue>7Dp/5KcvUfCnkohkOOzvFaeAIRc=</dsig:DigestValue>
[38]         </dsig:Reference>
```

Die dritte `dsig:Reference` (Zeile 33 - 38) entspricht der dritten Dateneinheit aus der Anfrage (siehe dort, Zeile 27 - 35). Man erkennt, dass der Wert von `sl:DataObject/@Reference` (Anfrage, Zeile 28 - 29) direkt in `dsig:Reference/@URI` übernommen wurde (Zeile 34 - 35).

```
[39]         <dsig:Reference Id="reference-3-09122004155437545" URI="">
[40]           <dsig:Transforms>
[41]             <dsig:Transform Algorithm="http://www.w3.org/2002/06/xmldsig-filter2">
[42]               <xpf:XPath Filter="intersect" xmlns:doc="urn:document"
[43]                 xmlns:xpf="http://www.w3.org/2002/06/xmldsig-filter2">
[44]                 /doc:Whole/doc:Part2</xpf:XPath>
```

```
[45]             </dsig:Transform>
[46]           </dsig:Transforms>
[47]           <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[48]           <dsig:DigestValue>7xBtEW4LIBb/UCX/YhZKZ/kMGPo=</dsig:DigestValue>
[49]         </dsig:Reference>
```

Die vierte `dsig:Reference` (Zeile 39 - 49) entspricht der vierten Dateneinheit aus der Anfrage (siehe dort, Zeile 36 - 49).

```
[50]         <dsig:Reference Id="etsi-its-0-09122004155437545"
[51]           Type="http://uri.etsi.org/01903/v1.1.1#SignedProperties" URI="">
[52]           <!--...-->
[53]       <dsig:SignatureValue><!--...--></dsig:SignatureValue>
[54]       <dsig:KeyInfo><!--...--></dsig:KeyInfo>
[55]       <dsig:Object Id="etsi-signed-2-09122004155437545"
[56]         xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
[57]         <etsi:QualifyingProperties Target="#signature-09122004155437545"
[58]           xmlns:etsi="http://uri.etsi.org/01903/v1.1.1#">
[59]           <!--...-->
[60]         </etsi:QualifyingProperties>
[61]       </dsig:Object>
[62]       <dsig:Object Id="signed-data-0-09122004155437545">
[63]         Von der Signatur umschlossene Daten.</dsig:Object>
[64]     </dsig:Signature>
[65]   </doc:Whole>
[66] </sl:CreateXMLSignatureResponse>
```

In Zeile 62 - 64 erkennt man das `dsig:Object` Element, in dem die zu signierenden Daten aus der ersten Dateneinheit der Anfrage (siehe dort, Zeile 8) eingebettet wurden.

#### *Downloads zu diesem Beispiel*

- examples/interface/createXMLSignature/SignatureTypes.xml
- examples/interface/common/SimpleText.txt
- examples/interface/createXMLSignature/SignatureTypes.Response.xml

#### *Weiterführende Informationen*

- Abschnitt 2.2, „Signatur nach XMLDSIG“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 2, „Keyboxen“ in Die österreichische Bürgerkarte - Standardisierte Key- und Infoboxen

### 2.1.2.5. Ergänzungsobjekte, Signaturmanifest

Dieses Beispiel stellt die Verwendung von Ergänzungsobjekten vor. Ein Ergänzungsobjekt betrifft entweder ein zu signierendes Datum (Zusammenhang mit einem `sl:DataObject`) oder jenes Dokument, in das eine zu erzeugende Signatur eingefügt werden soll (Zusammenhang mit `sl:SignatureEnvironment`). Es muss dann angegeben werden, wenn in einem zu signierenden Datum bzw. im Einfügedokument auf Daten per Referenz verwiesen wird, diese referenzierten Daten aber von der *Bürgerkarten-Umgebung* nicht aufgelöst werden können. Das Ergänzungsobjekt enthält dann genau diese Daten, die nicht von der *Bürgerkarten-Umgebung* aufgelöst werden können.

Weiters soll in diesem Beispiel erklärt werden, in welchen Fällen die *Bürgerkarten-Umgebung* automatisch ein sogenanntes Signaturmanifest in die zu erstellende XML-Signatur integriert.

#### *Anfrage*

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:CreateXMLSignatureRequest xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"
[03]   xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
[04]   <sl:KeyboxIdentifier>CertifiedKeypair</sl:KeyboxIdentifier>
```

Zeile 4 enthält die Angabe, dass eine Signatur mit dem kombinierten Signatur- und Verschlüsselungsschlüssel erstellt werden soll (`CertifiedKeypair`).

```
[05]   <sl:DataObjectInfo Structure="detached">
[06]     <sl:DataObject Reference="#Para2"/>
```

Das zu signierende Datum in diesem Beispiel ist ein Teil des Dokuments, in das die zu erstellende Signatur eingefügt werden soll. Der Wert des `Reference` Attributs in Zeile 6 ist `#Para2`, das bedeutet, dass jenes Element des Einfügedokuments signiert werden soll, das ein ID-Attribut mit dem Wert `#Para2` aufweist. Damit die *Bürgerkarten-Umgebung* diesen Hinweis auswerten kann, muss sie das Einfügedokument validierend parsen, denn sonst wüsste sie ja nicht, welche Attribute überhaupt ID-Attribute sind. Das zum validierenden Parsen notwendige XML-Schema wird der *Bürgerkarten-Umgebung* in diesem Beispiel über ein Ergänzungsobjekt zum Einfügedokument mitgeteilt (siehe Zeile 31 - 36 weiter unten).

```
[07]     <sl:TransformsInfo>
[08]       <dsig:Transforms>
[09]         <dsig:Transform Algorithm="http://www.w3.org/TR/1999/REC-xslt-19991116">
[10]           <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
[11]             <xsl:include href="XMLDocument.Para.xsl"/>
[12]           </xsl:stylesheet>
[13]         </dsig:Transform>
[14]         <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
[15]       </dsig:Transforms>
[16]       <sl:FinalDataMetaInfo>
[17]         <sl:MimeType>application/xhtml+xml</sl:MimeType>
[18]       </sl:FinalDataMetaInfo>
```

Das Beispiel enthält als erste Transformation eine *XSLT-Transformation*. Der als Kindelement von `dsig:Transform` angegebene Stylesheet verweist dabei mittels `xsl:include` auf einen weiteren Stylesheet. Dieser weitere Stylesheet kann jedoch von der *Bürgerkarten-Umgebung* nicht direkt aufgelöst werden, da er als relative Referenz angegeben wird. Deshalb ist es notwendig, diesen weiteren Stylesheet als Ergänzungsobjekt zu den signierenden Daten anzugeben.

Bei diesem weiteren Stylesheet handelt es sich auch um einen sogenannten *impliziten Transformationsparameter*, d. h. einen Transformationsparameter, der nicht direkt als Teil der Transformationsbeschreibung in die `dsig:Reference` der XML-Signatur aufgenommen wird, und damit auch nicht mitsigniert wird. Das spielt in vielen Anwendungsfällen zwar keine Rolle, wird aber wichtig, wenn eine *Applikation* später die XML-Signatur prüfen möchte, und für diese *Applikation* nicht die resultierenden *Hash-Eingangsdaten* (hier das resultierende XHTML-Dokument), sondern die ursprünglich referenzierten Daten (hier das XML-Dokument) von Bedeutung sind. Auf den korrekten Zusammenhang kann die *Applikation* nämlich in solchen Fällen nur dann vertrauen, wenn die gesamte Information mitsigniert ist, die für die Transformation von den ursprünglich referenzierten Daten in die *Hash-Eingangsdaten* notwendig ist. Sobald ein *impliziter Transformationsparameter* verwendet wird, ist diese Bedingung jedoch nicht mehr gegeben. Deshalb fügt die *Bürgerkarten-Umgebung* bei Verwendung von *impliziten Transformationsparametern* automatisch eine weitere `dsig:Reference` in die XML-Signatur ein, welche die

*impliziten Transformationsparameter* referenziert, die somit dann doch mitsigniert werden (vergleiche die Analyse der Antwort weiter unten).

```
[19]      </sl:TransformsInfo>
[20]      <sl:Supplement>
[21]        <sl:Content Reference="XMLDocument.Para.xsl">
[22]          <sl:LocRefContent>http://www.buergerkarte.at/konzept/securitylayer/spezifikation/\
[23]          20040514/tutorial/examples/interface/common/XMLDocument.Para.xsl</sl:LocRefContent>
[24]        </sl:Content>
[25]      </sl:Supplement>
[26]    </sl:DataObjectInfo>
```

Ein Ergänzungsobjekt für zu signierende Daten wird im entsprechenden `sl:DataObjectInfo` Element angegeben, und zwar als Inhalt des Elements `sl:Supplement`. Das verpflichtend zu verwendende Attribut `sl:Content/@Reference` in Zeile 21 enthält dabei die Referenz auf das Ergänzungsobjekt in exakt jener Schreibweise, wie sie in der `xsl:include` Direktive in Zeile 11 vorkommt, hier also `XMLDocument.Para.xsl`. Das Element `sl:Content` beinhaltet das Ergänzungsobjekt so, wie es die *Bürgerkarten-Umgebung* verwenden soll (Elemente `sl:Base64Content` oder `sl:XMLContent`, bzw. enthält eine von der *Bürgerkarten-Umgebung* auflösbare Referenz auf das Ergänzungsobjekt (Element `sl:LocRefContent`). Im konkreten Beispiel wird `sl:LocRefContent` verwendet.

```
[27]    <sl:SignatureInfo>
[28]      <sl:SignatureEnvironment Reference="http://www.buergerkarte.at/konzept/securitylayer/\
[29]      spezifikation/20040514/tutorial/examples/interface/common/XMLDocument.withSchemaHint.xml"/>
[30]      <sl:SignatureLocation Index="4" xmlns:doc="urn:document">/doc:XMLDocument</sl:SignatureLocation>
```

Eingefügt werden soll die zu erzeugende Signatur in ein bestehendes Dokument, das MOA SS durch Auflösen der in `sl:SignatureEnvironment/@Reference` angegebenen URL (Zeil 28 - 29) erhält. Eingefügt werden soll die Signatur als fünfter Kindknoten des Wurzelelements `doc:XMLDocument`. Beachten Sie wiederum die Hinweise zur ??? für das Attribut `Index` bzw. zur ??? der im XPath-Ausdruck verwendeten Namespace-Deklarationen (hier `doc`).

```
[31]      <sl:Supplement>
[32]        <sl:Content Reference="urn:XMLDocument.xsd">
[33]          <sl:LocRefContent>http://www.buergerkarte.at/konzept/securitylayer/spezifikation/20040514/\
[34]          tutorial/examples/interface/common/XMLDocument.xsd</sl:LocRefContent>
[35]        </sl:Content>
[36]      </sl:Supplement>
[37]    </sl:SignatureInfo>
[38] </sl:CreateXMLSignatureRequest>
```

Wie oben bereits angemerkt, muss die *Bürgerkarten-Umgebung* zur Auflösung der ID-Referenz `#Para2` das Einfügedokument validierend parsen. Im Einfügedokument ist zwar nun ein Hinweis auf die dazu notwendige Grammatikinformation in Form eines XML-Schemas enthalten (Attribut `xsi:schemaLocation` enthält den Wert `"urn:document urn:XMLDocument.xsd"`). Nachdem die darin angegebene URI `urn:XMLDocument.xsd` jedoch von der *Bürgerkarten-Umgebung* nicht direkt aufgelöst werden kann, wird im Request ein Ergänzungsobjekt zum Einfügedokument angegeben (`sl:Supplement`, Zeile 31 - 36). Das Attribut `sl:Content/@Reference` in Zeile 32 enthält die Referenz auf das Ergänzungsobjekt in exakt jener Schreibweise, wie sie im Attribut `xsi:schemaLocation` des XML-Dokuments angegeben wurde (`urn:XMLDocument.xsd`). Das Element `sl:Content` beinhaltet das Ergänzungsobjekt so, wie es die *Bürgerkarten-Umgebung* verwenden soll (Elemente `sl:Base64Content` oder `sl:XMLContent`), bzw. enthält eine von der *Bürgerkarten-Umgebung* auflösbare Referenz auf das Ergänzungsobjekt (Element `sl:LocRefContent`). Im konkreten Beispiel wird `sl:LocRefContent` verwendet (Zeile 33 - 34).

Beachten Sie bitte, dass die Verwendung von Ergänzungsobjekten für die Mitteilung von XML-Schemata mit den meisten *Bürgerkarten-Umgebungen* nur dann funktioniert, wenn die Referenz auf das XML-Schema in der `xsi:schemaLocation` eine absolute URI ist (also z.B. wie hier `urn:XMLDocument.xsd` oder auch `http://example.org/XMLDocument.xsd`, nicht aber z.B. `XMLDocument.xsd` oder `../schemas/XMLDocument.xsd`).

Auch für das Auflösen eines Verweises in einer DTD kann in analoger Weise von Ergänzungsobjekten Gebrauch gemacht werden.

### *Antwort*

Die entsprechende Antwort der *Bürgerkarten-Umgebung* sollte in etwa wie folgt aussehen. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert und gekürzt wurde, was natürlich die elektronische Signatur bricht.

Bezüglich der Ergänzungsobjekte enthält die Antwort keine besonders erwähnenswerten Besonderheiten. Jedoch soll die Behandlung des oben erwähnten impliziten Transformationsparameters (weiterer, referenzierter Stylesheet in Zeile 11der Anfrage) durch die *Bürgerkarten-Umgebung* analysiert werden.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:CreateXMLSignatureResponse
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]   <doc:XMLDocument xmlns="http://reference.e-government.gv.at/namespace/moa/20020822#"
[05]      xmlns:doc="urn:document" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
[06]      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[07]      xsi:schemaLocation="urn:document urn:XMLDocument.xsd">
[08]     <doc:Paragraph>Ich bin der erste Absatz in diesem Dokument.</doc:Paragraph>
[09]     <doc:Paragraph ParaId="Para2">Und ich bin der zweite Absatz in diesem Dokument.
[10] Ich habe weiters ein eigenens ID-Attribut bekommen.</doc:Paragraph>
[11]     <dsig:Signature Id="HS_signature" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
[12]       <dsig:SignedInfo>
[13]         <dsig:CanonicalizationMethod
[14]           Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
[15]         <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
[16]         <dsig:Reference Id="reference-data-0" URI="#Para2">
[17]           <dsig:Transforms>
[18]             <dsig:Transform Algorithm="http://www.w3.org/TR/1999/REC-xslt-19991116">
[19]               <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
[20]                 <xsl:include href="XMLDocument.Para.xsl"/>
[21]               </xsl:stylesheet>
[22]             </dsig:Transform>
[23]             <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
[24]           </dsig:Transforms>
[25]           <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[26]           <dsig:DigestValue>luM3wUmedTvkMHVedQkA/8otXUE=</dsig:DigestValue>
[27]         </dsig:Reference>
[28]         <dsig:Reference
[29]           Type="http://www.buergerkarte.at/specifications/Security-Layer/20020225#SignatureManifest"
[30]           URI="#Manifest">
[31]           <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[32]           <dsig:DigestValue>qCv1pfxB4ahrKCEPHRMMr5PhEAc=</dsig:DigestValue>
[33]         </dsig:Reference>
[34]         <dsig:Reference Type="http://uri.etsi.org/01903/v1.1.1#SignedProperties" URI="#refetsi">
[35]           <!--...-->
[36]         </dsig:Reference>
```

```
[37]       </dsig:SignedInfo>
```

Man erkennt, dass die *Bürgerkarten-Umgebung* im Element `dsig:SignedInfo` (Zeile 12 - 37) neben der Referenz auf die eigentlich zu signierenden Daten (Zeile 16 - 27 ) und der Referenz auf die Signatureigenschaften (Zeile 34 - 36) eine weitere Referenz auf das sogenannte *Signaturmanifest* eingefügt hat (Zeile 28 - 33). Man erkennt diese besondere Referenz an dem speziellen Wert des Attributs `Type` in Zeile 29.

```
[38]       <dsig:SignatureValue><!--...--></dsig:SignatureValue>
[39]       <dsig:KeyInfo><!--...--></dsig:KeyInfo>
[40]       <dsig:Object>
[41]         <dsig:Manifest Id="Manifest">
[42]           <dsig:Reference URI="XMLDocument.Para.xsl">
[43]             <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[44]             <dsig:DigestValue>Ihk/yhx06oOI0Qwi57Xs9dbMW0I=</dsig:DigestValue>
[45]           </dsig:Reference>
[46]         </dsig:Manifest>
[47]       </dsig:Object>
[48]       <dsig:Object Id="refetsi"><!--...--></dsig:Object>
[49]     </dsig:Signature>
[50]   </doc:XMLDocument>
[51] </sl:CreateXMLSignatureResponse>
```

Das *Signaturmanifest* selbst finden Sie einem `dsig:Object` Container in den Zeilen 40 - 47. Der Container enthält ein `dsig:Manifest`, das für jeden verwendeten *impliziten Transformationsparameter* der XML-Signatur ein `dsig:Reference`-Element enthält (im konkreten Beispiel eben eine `dsig:Reference` für den einen *impliziten Transformationsparameter*, den weiteren Stylesheet - vergleiche Zeile 11 der Anfrage bzw. Zeile 20 der Antwort). Der Wert des Attributs `URI` in Zeile 42 entspricht dabei genau jenem des Attributs `href` in Zeile 11 der Anfrage bzw. Zeile 20 der Antwort).

#### *Downloads zu diesem Beispiel*

- examples/interface/createXMLSignature/Supplements.xml
- examples/interface/common/XMLDocument.withSchemaHint.xml
- examples/interface/createXMLSignature/Supplements.Response.xml

#### *Weiterführende Informationen*

- Abschnitt 2.2, „Signatur nach XMLDSIG“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 2.2.2.2, „Implizite Transformationsparameter“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 2, „Keyboxen“ in Die österreichische Bürgerkarte - Standardisierte Key- und Infoboxen

## 2.2. Signaturen prüfen

### 2.2.1. Signatur im Format CMS prüfen

#### 2.2.1.1. Ein erstes Beispiel

#### *Anfrage*

Dieses Beispiel ist ein einfacher Request zur Prüfung einer CMS-Signatur. Sein Aufbau wird nachfolgend analysiert. Bitte beachten Sie, dass der nachfolgende Ausschnitt aus dem Request aus Gründen der Übersichtlichkeit gekürzt wurde.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:VerifyCMSSignatureRequest
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]   <sl:CMSSignature>MIIHsAYJKoZIh...6kpOPJaLg==</sl:CMSSignature>
[05] </sl:VerifyCMSSignatureRequest>
```

Der Request enthält in `sl:CMSSignature` in Zeile 4 die zu prüfende CMS-Signatur, und zwar in base64 kodierter Form. In diesem Beispiel wird davon ausgegangen, dass es sich dabei um eine *Enveloping* Signature handelt, d. h. dass die signierten Daten als Teil der CMS-Struktur vorhanden sind. Für die Behandlung einer *Detached* Signature sei auf

Abschnitt 2.2.1.2, „Erweitertes Beispiel“

verwiesen.

#### *Antwort*

Die entsprechende Antwort der *Bürgerkarten-Umgebung* sollte in etwa wie folgt aussehen. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert und gekürzt wurde.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:VerifyCMSSignatureResponse
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"
[04]   xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
[05]   <sl:SignerInfo>
[06]     <dsig:X509Data>
[07]       <dsig:X509SubjectName>C=AT,CN=Gregor Karlinger,S=Karlinger,G=Gregor,SN=913895552911
[08]       </dsig:X509SubjectName>
[09]       <dsig:X509IssuerSerial>
[10]         <dsig:X509IssuerName>C=AT,O=A-Trust Ges. f. Sicherheitssysteme im elektr. Datenverkehr\
[11]           GmbH,OU=a-sign-Premium-Sig-01,CN=a-sign-Premium-Sig-01</dsig:X509IssuerName>
[12]         <dsig:X509SerialNumber>6218
[13]         </dsig:X509SerialNumber>
[14]       </dsig:X509IssuerSerial>
[15]       <dsig:X509Certificate>
[16] MIIE4DCCA8igAwIBAgICGEowDQYJKoZIhvcNAQEFBQAwgZcxCzAJBgNVBAYTAkFUMUgwRgYDVQQK
[17] Ez9BLVRydXN0IEdlcy4gZi4gU...Rpz2MP3nU9H2IfKk36n6hhVpc3EC6aF02RdIBD+x8VxVsA==
[18]       </dsig:X509Certificate>
[19]       <sl:QualifiedCertificate/>
[20]     </dsig:X509Data>
[21]   </sl:SignerInfo>
```

Die Response enthält zunächst in `sl:SignerInfo/dsig:X509Data` (Zeile 5 - 21) Informationen über den Signator, die aus dem in der CMS-Signatur enthaltenen Signatorzertifikat entnommen sind.

`dsig:X509SubjectName` ist immer vorhanden und enthält den Namen des Signators. `dsig:X509IssuerSerial` ist ebenfalls immer vorhanden und

enthält den Namen des Austellers des Signatorzertifikats (`dsig:X509IssuerName`) sowie die Seriennummer des Zertifikats (`dsig:X509SerialNumber`).

Ob darüber hinaus in `dsig:X509Data` weitere Angaben zum Signator geliefert werden (wie in diesem Beispiel das Signatorzertifikat selbst in base64 kodierter Form, Zeile 15 - 18), ist abhängig von der verwendeten *Bürgerkarten-Umgebung*.

Optional vorhanden ist das inhaltslose Element `QualifiedCertificate`, und zwar dann, wenn es sich beim Signatorzertifikat um ein qualifiziertes Zertifikat handelt. Dies ist in diesem Beispiel der Fall (vergleiche Zeile 19).

```
[22]   <sl:SignatureCheck>
[23]     <sl:Code>0</sl:Code>
[24]     <sl:Info>Die Überprüfung des Werts der Signatur konnte erfolgreich durchgeführt\
[25]        werden.</sl:Info>
[26]   </sl:SignatureCheck>
```

Anschließend an `sl:SignerInfo` enthält die Response mit `sl:SignatureCheck` das Resultat der kryptographischen Prüfung der Signatur. `sl:Code` ist jedenfalls vorhanden und enthält einen Abschnitt 3.1.2, „Antwort“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer; in unserem Beispiel ist dort der Wert `0` enthalten, d. h. die Signatur konnte erfolgreich validiert werden. `sl:Info` enthält optional eine textuelle Erklärung des Ganzzahl-Kodes.

```
[27]   <sl:CertificateCheck>
[28]     <sl:Code>0</sl:Code>
[29]     <sl:Info>Jedes Zertifikat dieser Kette ist zum in der Anfrage angegebenen\
[30]        Prüfzeitpunkt gültig.</sl:Info>
[31]   </sl:CertificateCheck>
[32] </sl:VerifyCMSSignatureResponse>
```

Abschließend enthält die Response mit `sl:CertificateCheck` das Resultat der Prüfung des Signatorzertifikats. Zunächst prüft die *Bürgerkarten-Umgebung*, ob ausgehend vom Signatorzertifikat eine Zertifikatskette zu einem als vertrauenswürdig konfigurierten sog. *Trust Anchor* gebildet werden kann. Gelingt dies, wird die Gültigkeit jedes Zertifikats dieser Kette überprüft. `sl:Code` ist jedenfalls vorhanden und enthält einen Abschnitt 3.1.2.2, „Prüfung der Signaturprüfdaten“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer; in unserem Beispiel ist dort der Wert `0` enthalten, d. h. alle Prüfungen konnten erfolgreich durchgeführt werden. `sl:Info` enthält optional eine textuelle Erklärung des Ganzzahl-Kodes.

#### *Downloads zu diesem Beispiel*

- examples/interface/verifyCMSSignature/First.xml
- examples/interface/verifyCMSSignature/First.Response.xml

#### *Weiterführende Informationen*

- Abschnitt 3.1, „Signatur nach CMS“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 3.1.2, „Antwort“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 3.1.2.2, „Prüfung der Signaturprüfdaten“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer

### 2.2.1.2. Erweitertes Beispiel

#### *Anfrage*

Dieses erweiterte Beispiel zur Prüfung einer CMS-Signatur demonstriert die Prüfung mehrerer Signatoren einer CMS-Signatur, die Angabe des Prüfzeitpunkts sowie die Prüfung einer *Detached Signature*, d. h. einer Signatur, in der die signierten Daten nicht enthalten sind und daher extra angegeben werden müssen.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:VerifyCMSSignatureRequest Signatories="1"
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]   <sl:DateTime>2004-08-17T08:00:00+02:00</sl:DateTime>
[05]   <sl:CMSSignature>MIIHiwYJKoZIhvcNAQcCoII...ccNd5OLqgkfiwsvqSk48lou</sl:CMSSignature>
[06]   <sl:DataObject>
[07]     <sl:Content>
[08]       <sl:Base64Content>RGllc2UgRGF0ZW4gd2FyZW4gYmFzZTY0IGtvZGllcnQu</sl:Base64Content>
[09]     </sl:Content>
[10]   </sl:DataObject>
[11] </sl:VerifyCMSSignatureRequest>
```

Liegt eine zu prüfende CMS-Signatur vor, die von mehreren Unterzeichnenden signiert worden ist, kann das Attribut `sl:VerifyCMSSignatureRequest/@Signatories` verwendet werden, um jene Unterzeichnenden auszuwählen, deren Unterschriften von der *Bürgerkarten-Umgebung* geprüft werden sollen (vergleiche Zeile 2). Der Default-Wert für dieses optionale Attribut ist `1`. Soll nicht die Unterschrift allein des ersten Unterzeichnenden geprüft werden, muss das Attribut explizit angegeben werden. Es enthält dann eine oder mehrere Ganzzahlwerte, getrennt durch Leerzeichen. Jede Ganzzahl bezeichnet einen Unterzeichnenden, wobei die Reihenfolge der Auflistung der Unterzeichner in der CMS-Signatur entspricht. Der Wert `"1 3"` würde beispielsweise aussagen, dass die *Bürgerkarten-Umgebung* die Unterschrift des ersten sowie des dritten Unterzeichnenden prüfen soll.

Mit dem optionalen Element `sl:DateTime` in Zeile 4 kann der Zeitpunkt der Signaturprüfung explizit vorgegeben werden. Inhalt dieses Elements ist die Angabe von Datum und Uhrzeit entsprechend dem XML-Schema Datentyp *dateTime* . Enthält der angegebene Zeitpunkt keinen Zeitzonen-Offset zur UTC, wird der Zeitpunkt als lokale Zeit des Rechners interpretiert, auf dem die *Bürgerkarten-Umgebung* läuft. Wird `sl:DateTime` nicht angegeben, versucht die *Bürgerkarten-Umgebung*, den Zeitpunkt der Signaturerstellung aus der Signatur zu ermitteln (anhand des Signaturattributs `SigningTime`). Enthält die Signatur keinen Zeitpunkt der Signaturerstellung, verwendet die *Bürgerkarten-Umgebung* die aktuelle Systemzeit des Rechners, auf dem sie läuft.

Das optionale Element `sl:DataObject` muss dann angegeben werden, wenn eine *Detached Signature* geprüft werden soll, d. h. wenn in der CMS-Signatur die signierten Daten nicht mitkodiert sind. In `sl:DataObject/sl:Content/sl:Base64Content` sind in einem solchen Fall diese Daten in base64 kodierter Form bereit zu stellen (vgl. Zeile 6 - 10).

#### *Antwort*

Die Antwort der *Bürgerkarten-Umgebung* wird an dieser Stelle nicht näher analysiert, da sie keine für das Thema des Beispiels relevanten Besonderheiten aufweist.

#### *Downloads zu diesem Beispiel*

- examples/interface/verifyCMSSignature/Extended.xml
- examples/interface/verifyCMSSignature/Extended.Response.xml

#### *Weiterführende Informationen*

- Abschnitt 3.1, „Signatur nach CMS“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Definition des XML-Schema Datentyps *dateTime*
- Definition des Signaturattributs `SigningTime` in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer

### 2.2.2. Signatur im Format XML prüfen

#### 2.2.2.1. Ein erstes Beispiel

*Anfrage*

Nachfolgend finden Sie einen einfachen XML-Request zur Prüfung einer XML-Signatur. Bitte beachten Sie, dass der dargestellte Request zur bessernen Lesbarkeit eingerückt und gekürzt wurde.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:VerifyXMLSignatureRequest
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"
[04]   xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
[05]   <sl:SignatureInfo>
[06]     <sl:SignatureEnvironment>
[07]       <sl:XMLContent>
[08]         <dsig:Signature Id="signature-13122004171228295"
[09]           xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
[10]           <dsig:SignedInfo>
[11]             <dsig:CanonicalizationMethod
[12]               Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments"/>
[13]             <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
[14]             <dsig:Reference Id="reference-0-13122004171228295"
[15]               URI="#signed-data-0-13122004171228295">
[16]               <dsig:Transforms>
[17]                 <dsig:Transform Algorithm="http://www.w3.org/2002/06/xmldsig-filter2">
[18]                   <xpf:XPath Filter="intersect"
[19]                     xmlns:xpf="http://www.w3.org/2002/06/xmldsig-filter2">
[20]                     //*[@Id='signed-data-0-13122004171228295']/node()</xpf:XPath>
[21]                 </dsig:Transform>
[22]               </dsig:Transforms>
[23]               <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[24]               <dsig:DigestValue>pId+CwfdF8+4RdWSfPHIXeIJejA=</dsig:DigestValue>
[25]             </dsig:Reference>
[26]             ...
[27]           </dsig:SignedInfo>
[28]           <dsig:SignatureValue>...</dsig:SignatureValue>
[29]           <dsig:KeyInfo>...</dsig:KeyInfo>
[30]           <dsig:Object Id="etsi-signed-2-13122004171228295"
[31]             xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">...</dsig:Object>
[32]           <dsig:Object Id="signed-data-0-13122004171228295">Signiere mich bitte.</dsig:Object>
[33]         </dsig:Signature>
[34]       </sl:XMLContent>
[35]     </sl:SignatureEnvironment>
```

Das Element `sl:SignatureInfo` (Zeile 5 - 37) enthält einerseits das zu prüfende XML-Dokument, andererseits Angaben zur Position der XML-Signatur innerhalb des zu prüfenden XML-Dokuments.

Das Element `sl:VerifySignatureEnvironment` (Zeile 6 - 35) enthält jenes XML-Dokument, das die zu prüfende XML-Signatur enthält. Auch hier stehen eine Reihe von Möglichkeiten zur Verfügung, dieses XML-Dokument anzugeben. Im Beispiel wurde das Element `sl:XMLContent` verwendet; alternativ stehen die Elemente `sl:Base64Content` und `sl:LocRefContent` bzw. gleichwertig zu `sl:LocRefContent` das Attribut `sl:Reference` zur Verfügung.

Im konkreten Beispiel enthält das angegebene XML-Dokument direkt als Root-Element die zu prüfende Signatur (`dsig:Signature`). Es handelt sich dabei um eine *Enveloping Signature*, d. h. die signierten Daten sind in einem `dsig:Object` als Teil der XML-Struktur der Signatur kodiert. Tatsächlich signiert ist hier die Zeichenkette `Signiere mich bitte.`

```
[36]     <sl:SignatureLocation>/dsig:Signature</sl:SignatureLocation>
[37]   </sl:SignatureInfo>
[38] </sl:VerifyXMLSignatureRequest>
```

Das Element `sl:SignatureLocation` enthält als Text den XPath-Ausdruck zur Selektion der XML-Signatur innerhalb des zu prüfenden XML-Dokuments. Die Auswertung des XPath-Ausdrucks muss genau ein Element `dsig:Signature` ergeben. Bitte beachten Sie, dass im Kontext des Elements `sl:SignatureLocation` alle im XPath-Ausdruck verwendeten Namespace-Präfixe bekannt sein müssen (hier das Präfix `dsig`, deklariert in Zeile 4 des Requests).

*Antwort*

Die entsprechende Antwort der *Bürgerkarten-Umgebung* sollte in etwa wie folgt aussehen. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert und gekürzt wurde.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:VerifyXMLSignatureResponse
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"
[04]   xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
[05]   <sl:SignerInfo>
[06]     <dsig:X509Data>
[07]       <dsig:X509SubjectName>C=AT,CN=Gregor Karlinger,S=Karlinger,G=Gregor,SN=913895552911
[08]       </dsig:X509SubjectName>
[09]       <dsig:X509IssuerSerial>
[10]         <dsig:X509IssuerName>C=AT,O=A-Trust Ges. f. Sicherheitssysteme im elektr. \
[11]           Datenverkehr GmbH,OU=a-sign-Premium-Sig-01,CN=a-sign-Premium-Sig-01
[12]         </dsig:X509IssuerName>
[13]         <dsig:X509SerialNumber>6218</dsig:X509SerialNumber>
[14]       </dsig:X509IssuerSerial>
[15]       <dsig:X509Certificate>MIIE4DCCA8ig...c3EC6aF02RdIBD+x8VxVsA==</dsig:X509Certificate>
[16]       <sl:QualifiedCertificate/>
[17]     </dsig:X509Data>
[18]   </sl:SignerInfo>
```

Die Response enthält zunächst in `sl:SignerInfo` (Zeile 5 - 18) Informationen über den Signator.

Konnte die *Bürgerkarten-Umgebung* im Rahmen der Signaturprüfung ein dem öffentlichen Schlüssel entsprechendes Signatorzertifikat nach X.509 identifizieren, enthält `sl:SignerInfo` genau ein `dsig:X509Data` Element, welches wiederum wie folgt aufgebaut ist: `dsig:X509SubjectName` ist immer vorhanden und enthält den Namen des Signators. `dsig:X509IssuerSerial` ist ebenfalls immer vorhanden und enthält den Namen des Austellers des Signatorzertifikats (`dsig:X509IssuerName`) sowie die Seriennummer des Zertifikats (`dsig:X509SerialNumber`). Optional vorhanden ist das inhaltslose Element `QualifiedCertificate`, und zwar dann, wenn es sich beim Signatorzertifikat um ein qualifiziertes Zertifikat handelt. Dies ist in diesem Beispiel der Fall (vergleiche Zeile 16). Ob darüber hinaus in `dsig:X509Data` weitere Angaben zum Signator geliefert werden (wie in diesem Beispiel das Signatorzertifikat selbst in base64 kodierter Form, Zeile 15), ist abhängig von der verwendeten *Bürgerkarten-Umgebung*.

```
[19]   <sl:SignatureCheck>
[20]     <sl:Code>0</sl:Code>
[21]     <sl:Info>Die Überprüfung der Hash-Werte und des Werts der Signatur konnte erfolgreich \
[22]       durchgeführt werden.</sl:Info>
[23]   </sl:SignatureCheck>
```

Anschließend an `sl:SignerInfo` enthält die Response mit `sl:SignatureCheck` das Resultat der kryptographischen Prüfung der Signatur. `sl:Code` ist jedenfalls vorhanden und enthält einen Abschnitt 3.2.2, „Antwort“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer; in unserem Beispiel ist dort der Wert `0` enthalten, d. h. die Signatur konnte erfolgreich validiert werden. `sl:Info` enthält optional eine textuelle Erklärung des Ganzzahl-Kodes.

```
[24]   <sl:SignatureManifestCheck>
[25]     <sl:Code>0</sl:Code>
[26]     <sl:Info>Für diese Signatur ist kein Signaturmanifest notwendig.</sl:Info>
[27]   </sl:SignatureManifestCheck>
```

Danach enthält die Response mit `sl:SignatureManifestCheck` das Resultat der Überprüfung des Abschnitt 2.2.2.2, „Implizite Transformationsparameter“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer. `sl:Code` ist jedenfalls vorhanden und enthält einen Abschnitt 3.2.2, „Antwort“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer; in unserem Beispiel ist dort der Wert `0` enthalten, d. h. die zu prüfende Signatur ist kein Signaturmanifest notwendig. `sl:Info` enthält optional eine textuelle Erklärung des Ganzzahl-Kodes.

```
[28]   <sl:CertificateCheck>
[29]     <sl:Code>0</sl:Code>
[30]     <sl:Info>Jedes Zertifikat dieser Kette ist zum in der Anfrage angegebenen Prüfzeitpunkt \
[31]       gültig.</sl:Info>
[32]   </sl:CertificateCheck>
[33] </sl:VerifyXMLSignatureResponse>
```

Abschließend enthält die Response mit `sl:CertificateCheck` das Resultat der Prüfung des Signatorzertifikats. Zunächst prüft die *Bürgerkarten-Umgebung*, ob ausgehend vom Signatorzertifikat eine Zertifikatskette zu einem als vertrauenswürdig konfigurierten sog. *Trust Anchor* gebildet werden kann. Gelingt dies, wird die Gültigkeit jedes Zertifikats dieser Kette überprüft. `sl:Code` ist jedenfalls vorhanden und enthält einen Abschnitt 3.1.2.2, „Prüfung der Signaturprüfdaten“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer; in unserem Beispiel ist dort der Wert `0` enthalten, d. h. alle Prüfungen konnten erfolgreich durchgeführt werden. `sl:Info` enthält optional eine textuelle Erklärung des Ganzzahl-Kodes.

#### *Downloads zu diesem Beispiel*

- examples/interface/verifyXMLSignature/First.xml
- examples/interface/verifyXMLSignature/First.Response.xml

#### *Weiterführende Informationen*

- Abschnitt 3.2, „Signatur nach XMLDSIG“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 3.2.2, „Antwort“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 2.2.2.2, „Implizite Transformationsparameter“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 2.2.2.2, „Implizite Transformationsparameter“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 3.1.2.2, „Prüfung der Signaturprüfdaten“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer

### 2.2.2.2. Erweitertes Beispiel

#### *Anfrage*

Dieses erweiterte Beispiel zur Prüfung einer XML-Signatur demonstriert die explizite Angabe des Prüfzeitpunkts sowie die Spezifikation des XML-Dokuments mit der zu prüfenden Signatur per Referenz.

```
[01]<?xml version="1.0" encoding="UTF-8"?>
[02] <sl:VerifyXMLSignatureRequest
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"
[04]   xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
[05]   <sl:DateTime>2004-12-13T17:00:00</sl:DateTime>
```

Mit dem optionalen Element `sl:DateTime` in Zeile 5 kann der Zeitpunkt der Signaturprüfung explizit vorgegeben werden. Inhalt dieses Elements ist die Angabe von Datum und Uhrzeit entsprechend dem XML-Schema Datentyp *dateTime* . Enthält der angegebene Zeitpunkt keinen Zeitzonen-Offset zur UTC, wird der Zeitpunkt als lokale Zeit des Rechners interpretiert, auf dem die *Bürgerkarten-Umgebung* läuft. Wird `sl:DateTime` nicht angegeben, versucht die *Bürgerkarten-Umgebung*, den Zeitpunkt der Signaturerstellung aus der Signatur zu ermitteln (anhand des Signaturattributs ETSIXML in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer). Enthält die Signatur keinen Zeitpunkt der Signaturerstellung, verwendet die *Bürgerkarten-Umgebung* die aktuelle Systemzeit des Rechners, auf dem sie läuft.

```
[06]   <sl:SignatureInfo>
[07]     <sl:SignatureEnvironment Reference="http://www.buergerkarte.at/konzept/securitylayer/ \
[08]       spezifikation/20040514/tutorial/examples/interface/common/XMLDocument.signed.xml"/>
[09]     <sl:SignatureLocation
[10]       xmlns:doc="urn:Document">/doc:Document/dsig:Signature</sl:SignatureLocation>
[11]   </sl:SignatureInfo>
[12] </sl:VerifyXMLSignatureRequest>
```

Das XML-Dokument, welches die zu prüfende XML-Signatur beinhaltet, wird in diesem Fall nicht direkt als Inhalt in `sl:SignatureEnvironment/sl:XMLContent` oder `sl:SignatureEnvironment/sl:Base64Content` angegeben, sondern es wird lediglich per URL auf dieses Dokument referenziert (vergleiche Wert des Attributs `Reference` in Zeile 7 - 8). Die *Bürgerkarten-Umgebung* wird diese URL auflösen, um zum XML-Dokument mit der zu prüfenden XML-Signatur zu gelangen.

Das Element `sl:SignatureLocation` (Zeile 9 - 10) enthält als Text den XPath-Ausdruck zur Selektion der XML-Signatur innerhalb des zu prüfenden XML-Dokuments. Die Auswertung des XPath-Ausdrucks muss genau ein Element `dsig:Signature` ergeben. Bitte beachten Sie, dass im Kontext des Elements `sl:SignatureLocation` alle im XPath-Ausdruck verwendeten Namespace-Präfixe bekannt sein müssen (hier die Präfixe `dsig`, deklariert in Zeile 4 des Requests bzw. `doc`, deklariert in Zeile 10 des Requests).

#### *Antwort*

Die Antwort der *Bürgerkarten-Umgebung* wird an dieser Stelle nicht näher analysiert, da sie keine für das Thema des Beispiels relevanten Besonderheiten aufweist.

#### *Downloads zu diesem Beispiel*

- examples/interface/verifyXMLSignature/Extended.xml
- Referenziertes XML-Dokument der Anfrage
- examples/interface/verifyXMLSignature/Extended.Response.xml

#### *Weiterführende Informationen*

- Abschnitt 3.2, „Signatur nach XMLDSIG“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Definition des XML-Schema Datentyps *dateTime*
- ETSIXML in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer

### 2.2.2.3. Prüfung von XMLDSIG-Manifesten

#### *Anfrage*

Dieses Beispiel zur Prüfung einer XML-Signatur demonstriert die Prüfung eines in der XML-Signatur vorhandenden Manifests nach XMLDSIG. Bitte beachten Sie, dass der dargestellte Request zur besseren Lesbarkeit eingerückt und gekürzt wurde.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <VerifyXMLSignatureRequest
[03]   xmlns="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"
[04]   xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
[05]   <SignatureInfo>
[06]     <SignatureEnvironment>
[07]       <XMLContent>
[08]         <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#" Id="signature-1-1">
[09]           <dsig:SignedInfo>
[10]             <dsig:CanonicalizationMethod
[11]               Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
[12]             <dsig:SignatureMethod
[13]               Algorithm="http://www.buergerkarte.at/namespaces/ecdsa/200206030#ecdsa-sha1"/>
[14]             <dsig:Reference Type="http://www.w3.org/2000/09/xmldsig#Manifest"
[15]               URI="#dsig-manifest-1-1">
[16]               <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[17]               <dsig:DigestValue>nUUaW6OtcsNvV/QhqmkU2QXT1Mw=</dsig:DigestValue>
[18]             </dsig:Reference>
[19]           </dsig:SignedInfo>
[20]           <dsig:SignatureValue>...</dsig:SignatureValue>
[21]           <dsig:KeyInfo>...</dsig:KeyInfo>
[22]           <dsig:Object Id="signed-data-1-1-1">Diese Daten sind signiert.</dsig:Object>
[23]           <dsig:Object>
[24]             <dsig:Manifest Id="dsig-manifest-1-1">
[25]               <dsig:Reference Id="reference-1-1"
[26]                 URI="#xpointer(id('signed-data-1-1-1')/node())">
[27]                 <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[28]                 <dsig:DigestValue>EYxznGxNRAIcHQeUsj+zsK+uaHA=</dsig:DigestValue>
[29]               </dsig:Reference>
[30]             </dsig:Manifest>
[31]           </dsig:Object>
[32]         </dsig:Signature>
[33]       </XMLContent>
[34]     </SignatureEnvironment>
```

Das Element `sl:SignatureEnvironment` (Zeile 6 - 349 enthält als `sl:XMLContent` die zu prüfende XML-Signatur. Man erkennt, dass sich die einzige `dsig:Reference` im `dsig:SignedInfo` der XML-Signatur (Zeile 14 - 18) auf ein Manifest nach XMLDSig bezieht (erkennbar am Attribut `Type` in Zeile 14, das auf den Wert `http://www.w3.org/2000/09/xmldsig#Manifest` gesetzt ist). Im Response (siehe unten) werden wir deshalb ein eigenes Resultat für die Manifest-Prüfung erhalten.

Das Manifest selbst ist in einem `dsig:Object`, also innerhalb der XML-Struktur der XML-Signatur kodiert (Zeile 24 - 30). Es enthält eine `dsig:Reference` (Zeile 25 - 29), welche sich auf die Zeichenkette `Diese Daten sind signiert.` (Zeile 22) bezieht.

```
[35]     <SignatureLocation>//dsig:Signature</SignatureLocation>
[36]   </SignatureInfo>
[37] </VerifyXMLSignatureRequest>
```

Das Element `sl:SignatureLocation` (Zeile 35) enthält als Text den XPath-Ausdruck zur Selektion der XML-Signatur innerhalb des zu prüfenden XML-Dokuments. Die Auswertung des XPath-Ausdrucks muss genau ein Element `dsig:Signature` ergeben. Bitte beachten Sie, dass im Kontext des Elements `sl:SignatureLocation` alle im XPath-Ausdruck verwendeten Namespace-Präfixe bekannt sein müssen (hier das Präfix `dsig`, deklariert in Zeile 4 des Requests).

#### *Antwort*

Die entsprechende Antwort der *Bürgerkarten-Umgebung* sollte in etwa wie folgt aussehen. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert und gekürzt wurde.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:VerifyXMLSignatureResponse
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"
[04]   xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
[05]   <sl:SignerInfo>
[06]     <dsig:X509Data>...</dsig:X509Data>
[07]   </sl:SignerInfo>
[08]   <sl:SignatureCheck>...</sl:SignatureCheck>
[09]   <sl:SignatureManifestCheck>...</sl:SignatureManifestCheck>
```

Die Response enthält zunächst in `sl:SignerInfo/dsig:X509Data` Informationen über den Signator, die aus dem in der XML-Signatur enthaltenen Signatorzertifikat entnommen sind. Das Element `sl:SignatureCheck` enthält das Resultat der kryptographischen Prüfung der Signatur; das Element `sl:SignatureManifestCheck` das Resultat der Überprüfung des Abschnitt 2.2.2.2, „Implizite Transformationsparameter“ in Die österreichische

Bürgerkarte - Applikationsschnittstelle Security-Layer.

```
[10]   <sl:XMLDSIGManifestCheck>
[11]     <sl:Code>0</sl:Code>
[12]     <sl:Info>Für jede dsig:Reference des mittels sl:Info näher gekennzeichneten Manifests \
[13]       konnte der Hash-Wert erfolgreich überprüft werden.
[14]       <sl:ReferringSigReference>1</sl:ReferringSigReference>
[15]     </sl:Info>
[16]   </sl:XMLDSIGManifestCheck>
```

Neu ist in dieser Response das an `sl:SignatureCheck` anschließende Element `sl:XMLDSIGManifestCheck` (Zeile 10 - 16). Ein oder mehrere solche Elemente werden immer dann zurückgeliefert, wenn in `dsig:SignedInfo` der XML-Signatur `dsig:Reference` Elemente existieren, die sich auf ein Manifest nach XMLDSIG beziehen (siehe oben). Je solcher `dsig:Reference` enthält die Antwort ein korrespondierendes Element `sl:XMLDSIGManifestCheck`, im konkreten Beispiel als eines.

Das Element `sl:Code` gibt das Ergebnis der durchgeführten Prüfung des XMLDSIG-Manifests an. `sl:Code` ist jedenfalls vorhanden und enthält einen Abschnitt 3.2.2, „Antwort“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer; in unserem Beispiel ist dort der Wert `0` enthalten, d. h. die Prüfung jeder `dsig:Reference` im `dsig:Manifest` (im konkreten Beispiel also genau einer `dsig:Reference`) konnte erfolgreich durchgeführt werden. `sl:Info` enthält optional eine textuelle Erklärung des Ganzzahl-Kodes.

Das Element `sl:Info/sl:ReferringSigReference` enthält als Textinhalt die Nummer jenes `dsig:Reference` Elements in `dsig:SignedInfo` der XML-Signatur, welches auf das untersuchte Manifest nach XMLDSIG verweist, wobei mit `1` zu zählen begonnen wird. In diesem Beispiel war es also die erste `dsig:Reference`.

```
[17]   <sl:CertificateCheck>...</sl:CertificateCheck>
[18] </sl:VerifyXMLSignatureResponse>
```

Das Element `sl:CertificateCheck` enthält das Resultat der Zertifikatsprüfung.

#### *Downloads zu diesem Beispiel*

- examples/interface/verifyXMLSignature/XMLDSIGManifest.xml
- examples/interface/verifyXMLSignature/XMLDSIGManifest.Response.xml

#### *Weiterführende Informationen*

- Abschnitt 3.2, „Signatur nach XMLDSIG“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Spezifikation des XMLDSIG-Manifests
- Abschnitt 3.2.2, „Antwort“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer Abschnitt 3.1.2.2, „Prüfung der Signaturprüfdaten“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer

### 2.2.2.4. Ergänzungsobjekte

#### *Anfrage*

Dieses Beispiel zur Prüfung einer XML-Signatur demonstriert die Verwendung von Ergänzungsobjekten. Ein Ergänzungsobjekt betrifft entweder ein signiertes Datum (Zusammenhang mit einem `dsig:Reference` der XML-Signatur) oder jenes Dokument, in dem sich die zu prüfende XML-Signatur befindet (Zusammenhang mit `sl:SignatureEnvironment`). Es muss dann angegeben werden, wenn auf ein signiertes Datum bzw. in einem signierten Datum bzw. in dem die XML-Signatur enthaltenden XML-Dokument auf weitere Daten per Referenz verwiesen wird, diese Referenz aber von der *Bürgerkarten-Umgebung* nicht aufgelöst werden kann. Das Ergänzungsobjekt enthält dann genau diese Daten, die nicht von der *Bürgerkarten-Umgebung* aufgelöst werden können.

Bitte beachten Sie, dass der dargestellte Request zur bessernen Lesbarkeit eingerückt und gekürzt wurde.

```
[001] <?xml version="1.0" encoding="UTF-8"?>
[002] <sl:VerifyXMLSignatureRequest
[003]   xmlns=sl:"http://www.buergerkarte.at/namespaces/securitylayer/1.2#"
[004]   xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
[005]   <sl:SignatureInfo>
[006]     <sl:SignatureEnvironment>
[007]       <sl:XMLContent>
[008]         <doc:XMLDocument
[009]           xmlns="http://reference.e-government.gv.at/namespace/moa/20020822#"
[010]           xmlns:doc="urn:document" xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
[011]           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
[012]           xsi:schemaLocation="urn:document urn:XMLDocument.xsd">
[013]           <doc:Paragraph>Ich bin der erste Absatz in diesem Dokument.</doc:Paragraph>
[014]           <doc:Paragraph ParaId="Para2">Und ich bin der zweite Absatz in diesem Dokument.
[015]             Ich habe weiters ein eigenens ID-Attribut bekommen.</doc:Paragraph>
[016]           <dsig:Signature Id="signature-1-1"
[017]             xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
```

Das Element `sl:SignatureEnvironment` (Zeile 6 ff.) enthält das XML-Dokument (`doc:XMLDocument`, Zeile 8 ff.) mit der zu prüfenden XML-Signatur (Zeile 16 ff.).

```
[018]             <dsig:SignedInfo>
[019]               ...
[020]               <dsig:Reference Id="reference-1-1" URI="SimpleText.txt">
[021]                 <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[022]                 <dsig:DigestValue>7Dp/5KcvUfCnkohkOOzvFaeAIRc=</dsig:DigestValue>
[023]               </dsig:Reference>
```

Die erste `dsig:Reference` der zu prüfenden XML-Signatur (Zeile 20 - 23) verweist mit Hilfe der URI `SimpleText.txt` auf ein externes Dokument. Nachdem es sich dabei jedoch um eine relative URI handelt, wird sie die *Bürgerkarten-Umgebung* nicht auflösen können; es wird also ein passendes Supplement angegeben werden müssen (siehe unten).

```
[024]               <dsig:Reference Id="reference-1-2" URI="#Para2">
[025]                 <dsig:Transforms>
[026]                   <dsig:Transform Algorithm="http://www.w3.org/TR/1999/REC-xslt-19991116">
[027]                     <xsl:stylesheet
[028]                       version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
[029]                       <xsl:include href="XMLDocument.Para.xsl"/>
[030]                     </xsl:stylesheet>
[031]                   </dsig:Transform>
[032]                   <dsig:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
[033]                 </dsig:Transforms>
```

```
[034]                 <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[035]                 <dsig:DigestValue>luM3wUmedTvkMHVedQkA/8otXUE=</dsig:DigestValue>
[036]               </dsig:Reference>
[037]               ...
[038]             </dsig:SignedInfo>
[039]             <dsig:SignatureValue>...</dsig:SignatureValue>
[040]             <dsig:KeyInfo>...</dsig:KeyInfo>
[041]             ...
[042]           </dsig:Signature>
[043]         </doc:XMLDocument>
[044]       </sl:XMLContent>
[045]     </sl:SignatureEnvironment>
[046]     <sl:SignatureLocation>//dsig:Signature</sl:SignatureLocation>
[047]   </sl:SignatureInfo>
```

Man erkennt, dass das Attribut `URI` der zweiten `dsig:Reference` der zu prüfenden Signatur (Zeile 24 - 36) eine interne Referenz auf das Element `doc:Paragraph` mit dem auf den Wert `Para2` gesetzten ID-Attribut `ParaId` (vergleiche Zeile 14 weiter oben) enthält. Die *Bürgerkarten-Umgebung* kann jedoch den Umstand, dass es sich bei `doc:Paragraph/@ParaId` um ein ID-Attribut handelt, nur dann erkennen, wenn es das XML-Dokument validierend parst. Der dazu nötige Verweis auf das passende XML-Schema ist zwar mit dem Attribut `xsi:schemaLocation` vorhanden (vergleiche Zeile 12 weiter oben), jedoch handelt es sich dabei mit `urn:XMLDocument.xsd` um eine nicht auflösbare Referenz. Deshalb wird im Request ein passendes Ergänzungsobjekt benötigt (siehe unten).

Weiters erkennt man, dass `dsig:Reference` ein XSLT-Transformation enthält (Zeile 26 - 31). Im darin kodierten Stylesheet-Parameter (`dsig:Transform/xsl:stylesheet`) wird ein weiterer Stylesheet inkludiert (`XMLDocument.Para.xsl`, vergleiche Zeile 29). Diese Referenz ist aber wiederum für die *Bürgerkarten-Umgebung* nicht auflösbar. Auch hier wird also ein passendes Ergänzungsobjekt benötigt (siehe unten).

```
[048]   <sl:Supplement>
[059]     <sl:Content Reference="SimpleText.txt">
[050]       <sl:XMLContent>Ich bin ein einfacher Text.</sl:XMLContent>
[051]     </sl:Content>
[052]   </sl:Supplement>
```

Das erste Element `sl:Supplement` enthält nun das Ergänzungsobjekt für das oben beschriebene externe Textdokument (vergleiche Zeile 20 - 23). `sl:Content/@Reference` enthält die Referenz genau so, wie sie oben im Attribut `URI` der `dsig:Reference` angegeben wurde. Im Inhalt von `sl:Content` werden entweder explizit jene Daten angegeben, die von der *Bürgerkarten-Umgebung* statt des Auflösens der Referenz verwendet werden sollen (`sl:Base64Content` oder - wie im konkreten Beispiel - `sl:XMLContent`), oder aber es wird mit `sl:LocRefContent` eine auflösbare Referenz für diese Daten an die *Bürgerkarten-Umgebung* übergeben.

```
[053]   <sl:Supplement>
[054]     <sl:Content Reference="XMLDocument.Para.xsl">
[055]       <sl:XMLContent>
[056]         <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
[057]           xmlns:doc="urn:document" xmlns="http://www.w3.org/1999/xhtml">
[058]           <xsl:output encoding="UTF-8" method="xml" indent="yes"/>
[059]           <xsl:template match="/">
[060]             <html xmlns="http://www.w3.org/1999/xhtml">
[061]               <head>
[062]                 <title>HTML-Dokument</title>
[063]               </head>
[064]               <body>
[065]                 <xsl:apply-templates/>
[066]               </body>
[067]             </html>
[068]           </xsl:template>
[069]           <xsl:template match="doc:Paragraph">
[070]             <p>
[071]               <xsl:value-of select="child::text()"/>
[072]             </p>
[073]           </xsl:template>
[074]         </xsl:stylesheet>
[075]       </sl:XMLContent>
[076]     </sl:Content>
[077]   </sl:Supplement>
```

Das zweite Element `sl:Supplement` enthält nun das Ergänzungsobjekt für den oben beschriebenen inkludierten Stylesheet (vergleiche Zeile 29). `sl:Content/@Reference` enthält die Referenz genau so, wie sie oben im Attribut `xsl:stylesheet/@href` angegeben wurde. Im Inhalt von `sl:Content` werden entweder explizit jene Daten angegeben, die von der *Bürgerkarten-Umgebung* statt des Auflösens der Referenz verwendet werden sollen (`sl:Base64Content` oder - wie im konkreten Beispiel - `sl:XMLContent`), oder aber es wird mit `sl:LocRefContent` eine auflösbare Referenz für diese Daten an die *Bürgerkarten-Umgebung* übergeben.

```
[078]   <sl:Supplement>
[079]     <sl:Content Reference="urn:XMLDocument.xsd">
[080]       <sl:XMLContent>
[081]         <xs:schema targetNamespace="urn:document"
[082]           xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="urn:document"
[083]           elementFormDefault="qualified" attributeFormDefault="unqualified">
[084]           <xs:element name="XMLDocument">
[085]             <xs:complexType>
[086]               <xs:sequence>
[087]                 <xs:element name="Paragraph" maxOccurs="unbounded">
[088]                   <xs:complexType mixed="true">
[089]                     <xs:attribute name="ParaId" type="xs:ID" use="optional"/>
[090]                   </xs:complexType>
[091]                 </xs:element>
[092]                 <xs:any namespace="##other" minOccurs="0" maxOccurs="unbounded"
[093]                   processContent="lax"/>
[094]               </xs:sequence>
[095]             </xs:complexType>
[096]           </xs:element>
[097]         </xs:schema>
[098]       </sl:XMLContent>
[099]     </sl:Content>
[100]   </sl:Supplement>
[101] </sl:VerifyXMLSignatureRequest>
```

Das dritte Element `sl:Supplement` enthält analog das Ergänzungsobjekt für das oben beschriebene XML-Schema. `Content/@Reference` in Zeile 79 enthält die Referenz genau so, wie sie in Zeile 12 im Attribut `xsi:schemaLocation` angegeben wurde.

#### *Antwort*

Die Antwort der *Bürgerkarten-Umgebung* wird an dieser Stelle nicht näher analysiert, da sie keine für das Thema des Beispiels relevanten Besonderheiten aufweist.

#### *Downloads zu diesem Beispiel*

- examples/interface/verifyXMLSignature/Supplements.xml
- examples/interface/verifyXMLSignature/Supplements.Response.xml

#### *Weiterführende Informationen*

- Abschnitt 3.2, „Signatur nach XMLDSIG“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer

## 2.3. Daten verschlüsseln

### 2.3.1. Daten im CMS Format verschlüsseln

#### 2.3.1.1. Übergabe der zu verschlüsselnden Daten im Base64 Format

#### *Anfrage*

Der Text `Ich bin ein einfacher Text.` soll für bestimmte Empfänger verschlüsselt werden. Für die Verschlüsselungsoperation werden die öffentlichen Schlüssel der Empfänger benötigt (Element `RecipientPublicKey`), welche in deren X509 Zertifikaten enthalten sind. Die Zertifikate werden base64 kodiert und jedes Zertifikat wird in einem `X509Certificate` Element gespeichert. Bei diesem Beispiel wird nur ein Empfänger angegeben.

Die zu verschlüsselnden Daten werden base64-kodiert im Element `sl:ToBeEncrytped/sl:Content/sl:Base64Content` übergeben.

Weiters werden im Element `sl:ToBeEncrypted` die Metainformationen zu den zu verschlüsselnden Daten angegeben. Dabei muss das Element `sl:ToBeEncrypted/sl:MetaInfo/sl:MimeType` angegeben werden. Diese Information wird von der *Bürgerkarte* benötigt, um die Anzeige der zu verschlüsselnden Daten zu ermöglichen. Optional kann im Element `sl:ToBeEncrypted/sl:MetaInfo/sl:Description` eine verbale Beschreibung der Daten angegeben werden.

```
[01]  <?xml version="1.0" encoding="UTF-8"?>
[02]    <sl:EncryptCMSRequest
[03]      xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]      <sl:RecipientPublicKey>
[05]        <sl:X509Certificate>
[06]          MIIFBDCCA+ygAwIBAgIDASpYMA0GCSqGSIb3DQEBBQUAMIGXMQswCQYDVQQGEwJBVDFIMEYGA1UE
[07]          Cgw/QS1UcnVzdCBHZXMuIGYuIFNpY2hlcmhlaXRzc3lzdGVtZSBpbSBlbGVrdHIuIERhdGVudmVy
[08]          a2VociBHbWJIMR4wHAYDVQQLDBVhLXNpZ24tUHJlbWl1bS1FbmMtMDIxHjAcBgNVBAMMFWEtc2ln
[09]          bi1QcmVtaXVtLUVuYy0wMjAeFw0wNTA0MjEwNjE0MzdaFw0wOTA0MjEwNjE0MzdaMFoxCzAJBgNV
[10]          BAYTAkFUMRQwEgYDVQQDDAtQZXRlciBUZXVmbDEOMAwGA1UEBAwFVGV1ZmwxDjAMBgNVBCoMBVBl
[11]          dGVyMRUwEwYDVQQFEww2NjcxOTI1NzA2MzQwgd8wDQYJKoZIhvcNAQEBBQADgc0AMIHJAoHBALzp
[12]          AKja0OI2iGC+ufp8hMYo/UliAjIY/HcOgIb+UN+0qL4RmGEt1CTYBcm6t3NIGi9+pVTat0nKmSsH
[13]          qs5NtlZJvahIHrs6q/Nvs6vzLTVHkRwl9CcgsF54MdKz/LzE41yZ+EE07HqW8l69OIXNSVrFS4kG
[14]          oEJUHFGWdke71Kpbfu4+2d2cfU9jMX/rUzBz/fcbeg2IMY3DhI4uH7R492eS5bEmbZnYlSuvK4Em
[15]          3Mx3TFrL8ZOjNOCnfJAuAbf9gwIDAQABo4IB1zCCAdMwEwYDVR0jBAwwCoAIRyFHjpdh4x4wewYI
[16]          KwYBBQUHAQEEbzBtMEIGCCsGAQUFBzAChjZodHRwOi8vd3d3LmEtdHJ1c3QuYXQvY2VydHMvYS1z
[17]          aWduLVByZW1pdW0tRW5jLTAyYS5jcnQwJwYIKwYBBQUHMAGGG2h0dHA6Ly9vY3NwLmEtdHJ1c3Qu
[18]          YXQvb2NzcDBNBgNVHSAERjBEMEIGBiooABEBDDA4MDYGCCsGAQUFBwIBFipodHRwOi8vd3d3LmEt
[19]          dHJ1c3QuYXQvZG9jcy9jcC9hLXNpZ24tdG9rZW4wgZoGA1UdHwSBkjCBjzCBjKCBiaCBhoaBg2xk
[20]          YXA6Ly9sZGFwLmEtdHJ1c3QuYXQvb3U9YS1zaWduLVByZW1pdW0tRW5jLTAyLG89QS1UcnVzdCxj
[21]          PUFUP2NlcnRpZmljYXRlcmV2b2NhdGlvbmxpc3Q/YmFzZT9vYmplY3RjbGFzcz1laWRDZXJ0aWZp
[22]          Y2F0aW9uQXV0aG9yaXR5MBEGA1UdDgQKBAhEc3YCuoW7uDAOBgNVHQ8BAf8EBAMCBLAwJQYDVR0R
[23]          BB4wHIEacGV0ZXIudGV1ZmxAaWFpay50dWdyYXouYXQwCQYDVR0TBAIwADANBgkqhkiG9w0BAQUF
[24]          AAOCAQEAJfg2cBaLMaXVqoZ23UA8qKxTxh+WeSlEveI4Ca43tu89uutJ3w/rXVFJ1EcSaA4OTAJt
[25]          icp5LstzJmrJoTcxbb3gC46x5MrgyvDbiTb/AiHBwl1C0GN3pjv1cqFfOMYvuWPbliVPgCdCYqva
[26]          sr5KNWbge9r0tEh6Oogx0zAVrsdSYN30eSECch6NKlptD6L5KRKoorlCIBq7n2U70DpSWFYQHegJ
[27]          ier2yeY5hG6ceIZKKJ/fjDLH2NzWidoXk3NWqc3X836YCnoNyQ0oqgkz6gPSyWTpWnJ+j/WNBouA
[28]          ImEAiehOOgnNBJgS72z5iJsDFcLfI6cX/WmibSp3nR/AMQ==
[29]        </sl:X509Certificate>
[30]      </sl:RecipientPublicKey>
[31]    <sl:ToBeEncrypted>
[32]    <sl:MetaInfo>
[33]      <sl:MimeType>
[34]        text/plain
[35]      </sl:MimeType>
[36]      <sl:Description>
[37]           Diese Daten liegen als reiner Text vor.
[38]       </sl:Description>
[39]    </sl:MetaInfo>
[40]    <sl:Content>
[41]      <sl:Base64Content>
[42]        SWNoIGJpbiBlaW4gZWluZmFjaGVyIFRleHQu
[43]      </sl:Base64Content>
[44]    </sl:Content>
[45]    </sl:ToBeEncrypted>
[46]  </sl:EncryptCMSRequest>
```

Zeile 2 enthält den passenden Befehl der Schnittstelle *Security-Layer*.

Zeile 3 enthält die Namenraum-Deklaration für die XML-Elemente, aus denen die Anfrage besteht.

Die Zeilen 4 bis 30 enthalten das base64-kodierte X509 Zertifikat des Empfängers.

Die Zeilen 32 bis 39 geben Metainformationen an, die für Anzeige der zu verschlüsselnden Daten benötigt werden.

Die Zeilen 40 bis 44 enthalten die zu verschlüsselnden Daten (der Text `Ich bin ein einfacher Text`) kodiert im Base64 Format.

#### *Antwort*

Die Antwort enthält den verschlüsselten Text als CMS Objekt im Element `CMSMessage` .

```
[01]  <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
[02]  <sl:EncryptCMSResponse
```

```
[03]    xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]    <sl:CMSMessage>
[05]      MIIB3AYJKoZIhvcNAQcDoIIBzTCCAckCAQExggF7MIIBdwIBATCBnzCBlzEL
[06]      MAkGA1UEBhMCQVQxSDBGBgNVBAoMP0EtVHJ1c3QgR2VzLiBmLiBTaWNoZXJo
[07]      ZWl0c3N5c3RlbWUgaW0gZWxla3RyLiBEYXRlbnZlcmtlaHIgR21iSDEeMBwG
[08]      A1UECwwVYS1zaWduLVByZW1pdW0tRW5jLTAyMR4wHAYDVQQDDBVhLXNpZ24t
[09]      UHJlbWl1bS1FbmMtMDICAwEqWDANBgkqhkiG9w0BAQEFAASBwBiwSIHzq6LK
[10]      4RcT6wrA6TuJAHgsVRtirQQhMkvgSWyozC5SJoyYDVuqFgci+0MwBioPp7H6
[11]      gv0m2RAp3p7MdyaUBY7WzC9X5anTcioCuI1E4EoQJGyg+rUD7PzrRl/HroP3
[12]      EEzGK7jZCJ9ToWmleMMYsLmtkMJ5MlnRdtyuReLU8ATfGCJOMSJlUDuiVqmU
[13]      UOSO/l8M6AyXz7ZJ7ntgf6IJtOo0G/f5Ph/smWWXltKD2nWxzJUUaXs75lfB
[14]      +/KfTjBFBgkqhkiG9w0BBwEwFAYIKoZIhvcNAwcECPEnrwVJxQ54oCIEIBcB
[15]      bbIEsKpuWsUxFFPBBjTJtV8rVFXfpTBFuC03ltTo
[16]    </sl:CMSMessage>
[17]  </sl:EncryptCMSResponse>
```

*Downloads zu diesem Beispiel*

- examples/interface/encryptCMS/encryptCMS_Base64.xml
- examples/interface/encryptCMS/encryptCMS_Base64_Answer.xml

*Weiterführende Information*

- Abschnitt 4.1, „Verschlüsselung als CMS-Nachricht“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer

#### 2.3.1.2. Übergabe der zu verschlüsselnden Daten als Referenz

*Anfrage*

Bei diesem Beispiel werden die zu verschlüsselnden Daten als Referenz im Attribut `sl:ToBeEncrypted/sl:Content/@Reference` angegeben. Sonst entspricht die Anfrage jener aus Abschnitt 2.3.1.1, „Übergabe der zu verschlüsselnden Daten im Base64 Format“.

```
[01]  <?xml version="1.0" encoding="UTF-8"?>
[02]    <sl:EncryptCMSRequest
[03]      xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]      <sl:RecipientPublicKey>
[05]        <sl:X509Certificate>
[06]          MIIFBDCCA+ygAwIBAgIDASpYMA0GCSqGSIb3DQEBBQUAMIGXMQswCQYDVQQGEwJBVDFIMEYGA1UE
[07]          Cgw/QS1UcnVzdCBHZXMuIGYuIFNpY2hlcmhlaXRzc3lzdGVtZSBpbSBlbGVrdHIuIERhdGVudmVy
[08]          a2VociBHbWJIMR4wHAYDVQQLDBVhLXNpZ24tUHJlbWl1bS1FbmMtMDIxHjAcBgNVBAMMFWEtc2ln
[09]          bi1QcmVtaXVtLUVuYy0wMjAeFw0wNTA0MjEwNjE0MzdaFw0wOTA0MjEwNjE0MzdaMFoxCzAJBgNV
[10]          BAYTAkFUMRQwEgYDVQQDDAtQZXRlciBUZXVmbDEOMAwGA1UEBAwFVGV1ZmwxDjAMBgNVBCoMBVBl
[11]          dGVyMRUwEwYDVQQFEww2NjcxOTI1NzA2MzQwgd8wDQYJKoZIhvcNAQEBBQADgc0AMIHJAoHBALzp
[12]          AKja0OI2iGC+ufp8hMYo/U1iAjIY/HcOgIb+UN+0qL4RmGEt1CTYBcm6t3NIGi9+pVTat0nKmSsH
[13]          qs5NtlZJvahIHrs6q/Nvs6vzLTVHkRwl9CcgsF54MdKz/LzE41yZ+EE07HqW8l69OIXNSVrFS4kG
[14]          oEJUHFGWdke71Kpbfu4+2d2cfU9jMX/rUzBz/fcbeg2IMY3DhI4uH7R492eS5bEmbZnYlSuvK4Em
[15]          3Mx3TFrL8ZOjNOCnfJAuAbf9gwIDAQABo4IB1zCCAdMwEwYDVR0jBAwwCoAIRyFHjpdh4x4wewYI
[16]          KwYBBQUHAQEEbzBtMEIGCCsGAQUFBzAChjZodHRwOi8vd3d3LmEtdHJ1c3QuYXQvY2VydHMvYS1z
[17]          aWduLVByZW1pdW0tRW5jLTAyYS5jcnQwJwYIKwYBBQUHMAGGG2h0dHA6Ly9vY3NwLmEtdHJ1c3Qu
[18]          YXQvb2NzcDBNBgNVHSAERjBEMEIGBioobABEBDDA4MDYGCCsGAQUFBwIBFipodHRwOi8vd3d3LmEt
[19]          dHJ1c3QuYXQvZG9jcy9jcC9hLXNpZ24tdG9rZW4wgZoGA1UdHwSBkjCBjzCBjKCBiaCBhoaBg2xk
[20]          YXA6Ly9sZGFwLmEtdHJ1c3QuYXQvb3U9YS1zaWduLVByZW1pdW0tRW5jLTAyLG89QS1UcnVzdCxj
[21]          PUFUP2NlcnRpZmljYXRlcmV2b2NhdGlvbmxpc3Q/YmFzZT9vYmplY3RjbGFzcz1laWRDZXJ0aWZp
[22]          Y2F0aW9uQXV0aG9yaXR5MBEGA1UdDgQKBAhEc3YCuoW7uDAOBgNVHQ8BAf8EBAMCBLAwJQYDVR0R
[23]          BB4wHIEacGV0ZXIudGV1ZmxAaWFpay50dWdyYXouYXQwCQYDVR0TBAIwADANBgkqhkiG9w0BAQUF
[24]          AAOCAQEAJfg2cBaLMaXVqoZ23UA8qKxTxh+WeSlEveI4Ca43tu89uutJ3w/rXVFJlEcSaA4OTAJt
[25]          icp5LstzJmrJoTcxbb3gC46x5MrgyvDbiTb/AiHBw11C0GN3pjv1cqFfOMYvuWPbliVPgCdCYqva
[26]          sr5KNWbge9r0tEh6Oogx0zAVrsdSYN30eSECch6NKlptD6L5KRKoorlCIBq7n2U70DpSWFYQHegJ
[27]          ier2yeY5hG6ceIZKKJ/fjDLH2NzWidoXk3NWqc3X836YCnoNyQ0oqgkz6gPSyWTpWnJ+j/WNBouA
[28]          ImEAiehOOgnNBJgS72z5iJsDFcLfI6cX/WmibSp3nR/AMQ==
[29]        </sl:X509Certificate>
[30]      </sl:RecipientPublicKey>
[31]    <sl:ToBeEncrypted>
[32]      <sl:MetaInfo>
[33]        <sl:MimeType>
[34]          text/html
[35]        </sl:MimeType>
[36]       <sl:Description>
[37]         Diese Daten liegen als reiner Text vor.
[38]       </sl:Description>
[39]      </sl:MetaInfo>
[40]      <sl:Content
[41]        Reference="http://www.buergerkarte.at/konzept/securitylayer/spezifikation/aktuell/tutorial/Tutorial.html">
[42]      </sl:Content>
[43]    </sl:ToBeEncrypted>
[44]  </sl:EncryptCMSRequest>
```

*Antwort*

Die Antwort entspricht bis auf die verschlüsselten Daten der Antwort aus Abschnitt 2.3.1.1, „Übergabe der zu verschlüsselnden Daten im Base64 Format“.

*Downloads zu diesem Beispiel*

- examples/interface/encryptCMS/encryptCMS_Reference.xml
- examples/interface/encryptCMS/encryptCMS_Reference_Answer.xml

*Weiterführende Information*

- Abschnitt 4.1, „Verschlüsselung als CMS-Nachricht“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer

### 2.3.2. Daten im XML Format verschlüsseln

#### 2.3.2.1. Verschlüsselung eines vollständigen XML Dokuments (`New`)

### *Anfrage*

Soll ein vollständiges XML Dokument oder beliebige Daten nach dem XMLENC Standard verschlüsselt werden, so wird das Element `New` verwendet. Es gibt drei verschiedene Möglichkeiten, wie die zu verschlüsselnden Daten angegeben werden können:

- direkte Einbettung im Request in base64-kodierter Form (Verwendung des Elements `sl:DataObject/sl:Base64Content`)
- direkte Einbettung im Request als XML-Fragment (Verwendung des Elements `sl:DataObject/sl:XMLContent`)
- Referenzierung mittels URL, die von der Bürgerkarten-Umgebung aufgelöst werden kann

Das folgende Beispiel verschlüsselt ein vollständiges XML Dokument, das als XML Fragment im Element `XMLContent` angegeben wird.

```
[01]  <?xml version="1.0" encoding="UTF-8"?>
[02]  <sl:EncryptXMLRequest
[03]    xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"
[04]    xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
[05]    xmlns:docns="http://www.w3.org/1999/xhtml">
[06]    <sl:RecipientPublicKey>
[07]      <sl:X509Certificate>
[08]        MIIFBDCCA+ygAwIBAgIDASpYMA0GCSqGSIb3DQEBBQUAMIGXMQswCQYDVQQGEwJBVDFIMEYGA1UE
[09]        Cgw/QS1UcnVzdCBHZXMuIGYuIFNpY2hlcmhlaXRzc3lzdGVtZSBpbSBlbGVrdHIuIERhdGVudmVy
[10]        a2VociBHbWJIMR4wHAYDVQQLDBVhLXNpZ24tUHJlbWl1bS1FbmMtMDIxHjAcBgNVBAMMFWEtc2ln
[11]        bi1QcmVtaXVtLUVuYy0wMjAeFw0wNTA0MjEwNjE0MzdaFw0wOTA0MjEwNjE0MzdaMFoxCzAJBgNV
[12]        BAYTAkFUMRQwEgYDVQQDDAtQZXRlciBUZXVmbDEOMAwGA1UEBAwFVGV1ZmwxDjAMBgNVBCoMBVBl
[13]        dGVyMRUwEwYDVQQFEww2NjcxOTI1NzA2MzQwgd8wDQYJKoZIhvcNAQEBBQADgc0AMIHJAoHBALzp
[14]        AKja0OI2iGC+ufp8hMYo/U1iAjIY/HcOgIb+UN+0qL4RmGEt1CTYBcm6t3NIGi9+pVTat0nKmSsH
[15]        qs5NtlZJvahIHrs6q/Nvs6vzLTVHkRwl9CcgsF54MdKz/LzE41yZ+EE07HqW8l69OIXNSVrFS4kG
[16]        oEJUHFGWdke71Kpbfu4+2d2cfU9jMX/rUzBz/fcbeg2IMY3DhI4uH7R492eS5bEmbZnYlSuvK4Em
[17]        3Mx3TFrL8ZOjNOCnfJAuAbf9gwIDAQABo4IB1zCCAdMwEwYDVR0jBAwwCoAIRyFHjpdh4x4wewYI
[18]        KwYBBQUHAQEEbzBtMEIGCCsGAQUFBzAChjZodHRwOi8vd3d3LmEtdHJ1c3QuYXQvY2VydHMvYS1z
[19]        aWduLVByZW1pdW0tRW5jLTAyYS5jcnQwJwYIKwYBBQUHMAGGG2h0dHA6Ly9vY3NwLmEtdHJ1c3Qu
[20]        YXQvb2NzcDBNBgNVHSAERjBEMEIGBiooABEBDDA4MDYGCCsGAQUFBwIBFipodHRwOi8vd3d3LmEt
[21]        dHJ1c3QuYXQvZG9jcy9jcC9hLXNpZ24tdG9rZW4wgZoGA1UdHwSBkjCBjzCBjKCBiaCBhoaBg2xk
[22]        YXA6Ly9sZGFwLmEtdHJ1c3QuYXQvb3U9YS1zaWduLVByZW1pdW0tRW5jLTAyLG89QS1UcnVzdCxj
[23]        PUFUP2NlcnRpZmljYXRlcmV2b2NhdGlvbmxpc3Q/YmFzZT9vYmplY3RjbGFzczllaWRDZXJ0aWZp
[24]        Y2F0aW9uQXV0aG9yaXR5MBEGA1UdDgQKBAhEc3YCuoW7uDAOBgNVHQ8BAf8EBAMCBLAwJQYDVR0R
[25]        BB4wHIEacGV0ZXIudGV1ZmxAaWFpay50dWdyYXouYXQwCQYDVR0TBAIwADANBgkqhkiG9w0BAQUF
[26]        AAOCAQEAJfg2cBaLMaXVqoZ23UA8qKxTxh+WeSlEveI4Ca43tu89uutJ3w/rXVFJ1EcSaA4OTAJt
[27]        icp5LstzJmrJoTcxbb3gC46x5MrgyvDbiTb/AiHBw11C0GN3pjv1cqFfOMYvuWPb1iVPgCdCYqva
[28]        sr5KNWbge9r0tEh6Oogx0zAVrsdSYN30eSECCh6NKlptD6L5KRKoorlCIBq7n2U70DpSWFYQHegJ
[29]        ier2yeY5hG6ceIZKKJ/fjDLH2NzWidoXk3NWqc3X836YCnoNyQ0oqgkz6gPSyWTpWnJ+j/WNBouA
[30]        ImEAiehOOgnNBJgS72z5iJsDFcLfI6cX/WmibSp3nR/AMQ==
[31]      </sl:X509Certificate>
[32]    </sl:RecipientPublicKey>
[33]    <sl:ToBeEncrypted>
[34]      <New ParentSelector="/" NodeCount="0">
[35]        <sl:MetaInfo>
[36]          <sl:MimeType>text/html</sl:MimeType>
[37]        </sl:MetaInfo>
[38]        <sl:Content>
[39]          <sl:XMLContent>
[40]            <html xmlns="http://www.w3.org/1999/xhtml">
[41]              <head>
[42]                <title>Ein einfaches SLXHTML-Dokument</title>
[43]                <style type="text/css">p { color: red; }</style>
[44]              </head>
[45]              <body>
[46]                <p>Ich bin ein einfacher Text in rot.</p>
[47]              </body>
[48]            </html>
[49]          </sl:XMLContent>
[50]        </sl:Content>
[51]      </sl:New>
[52]    </sl:ToBeEncrypted>
[53]  </sl:EncryptXMLRequest>
```

Zeile 2 enthält den passenden Befehl der Schnittstelle *Security-Layer*.

Die Zeilen 3 bis 5 enthalten die Namenraum-Deklarationen für die XML-Elemente, aus denen die Anfrage besteht.

Die Zeilen 6 bis 32 enthalten das base64-kodierte X509 Zertifikat des Empfängers.

Die Zeilen 35 bis 37 geben Metainformationen an, die für Anzeige der zu verschlüsselnden Daten benötigt werden.

Die Zeilen 40 bis 48 enthalten das zu verschlüsselnde XML Dokument.

### *Antwort*

Das angegebene Dokument wird verschlüsselt und folgende Antwort wird zurückgeliefert.

```
[01]  <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
[02]  <sl:EncryptXMLResponse
[03]    xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]    <sl:EncryptionEnvironment>
[05]      <xenc:EncryptedData
[06]        xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
[07]         Id="encrypted-data-0-1152523792-32960945-15207">
[08]        <xenc:EncryptionMethod
[09]          Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
[10]        <ds:KeyInfo
[11]          xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
[12]          <xenc:EncryptedKey
[13]            Id="encrypted-key-1-1152523792-32960975-11645-c0">
[14]            <xenc:EncryptionMethod
[15]              Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
[16]            <ds:KeyInfo>
[17]              <ds:KeyValue>
[18]                <ds:RSAKeyValue>
[19]                  <ds:Modulus>
[20]                    vOkAqNrQ4jaIYL65+nyExij9TWICMhj8dw6Ahv5Q37SovhGYYS3UJNgFybq3
```

```
[21]                    c0gaL36lVNq3ScqZKweqzk22Vkm9qEgeuzqr82+zq/MtNUeRHCX0JyCwXngx
[22]                    0rP8vMTjXJn4QTTsepbyXr04hc1JWsVLiQagQlQcUZZ2R7vUqlt+7j7Z3Zx9
[23]                    T2Mxf+tTMHP99xt6DYgxjcOEji4ftHj3Z5LlsSZtmdiVK68rgSbczHdMWsvx
[24]                    k6M04Kd8kC4Bt/2D
[25]                  </ds:Modulus>
[26]                  <ds:Exponent>
[27]                    AQAB
[28]                  </ds:Exponent>
[29]                </ds:RSAKeyValue>
[30]              </ds:KeyValue>
[31]            </ds:KeyInfo>
[32]            <xenc:CipherData>
[33]              <xenc:CipherValue>
[34]                FsLNfU5wpNNTyRjmAxWVGAnH1Qfj2/3LfffD1D3kleCQQtg3JjmrKlCp8NDr
[35]                JT7vP2Ymb+TTrm9NUKkJzo7B1cDX0uB86GefY+j9aX/L7PB2x9hu0/1kpvxn
[36]                xtWaqe6PnoEVZr6Y0Af5bAPOEcMnvi/lnxqjz2EsiF5AuP4MktkXuIWuKzEk
[37]                oc/bKWDJUuLzVQN+JVKPfuZJwB1yjaoFguA+4dW+17mdS3OLmde4sIfJxaMD
[38]                aUN9thYDAZR3pnU9
[39]              </xenc:CipherValue>
[40]            </xenc:CipherData>
[41]          </xenc:EncryptedKey>
[42]        </ds:KeyInfo>
[43]        <xenc:CipherData>
[44]          <xenc:CipherValue>
[45]            l7VSOyeHIbF/lRgLNUD7is97aSA7VQT239/B2ZZ1CfAs6UHUwbNugP/4ymbL
[46]            HAwIjvo1eWXit/WdEi1PhkiGRGEDLz7vX0xgkc2SauZ1f5e4/irvhUjYm1nb
[47]            nk/JoV8gaItGVY/ZVjtdXTKy48oJk7hgltIRi5Mnf7XZIevt0e1Z3eVdcam9
[48]            k+z+kBahsSJIL6bwaK6N6dtk38nA+4/1iVvm676Np06Uq42Q/rHPXm0IVcQ3
[49]            11yirh3idgQUx60J0+J0tGAOzpfElUnDwk3pwr86kjg7eyzUjYGHT+IFidjv
[50]            H0WR4MvlXA==
[51]          </xenc:CipherValue>
[52]        </xenc:CipherData>
[53]      </xenc:EncryptedData>
[54]    </sl:EncryptionEnvironment>
[55]  </sl:EncryptXMLResponse
```

Zeile 2 enthält die passende Antwort der Schnittstelle *Security-Layer*.

Die Zeilen 3 enthält die Namenraum-Deklaration für die XML-Elemente, aus denen die Anfrage besteht.

Die Zeilen 5 bis 15 enthalten Angaben zum verschlüsselten Dokument und den verwendeten Algorithmen

Die Zeilen 12 bis 41 enthalten den mit RSA verschlüsselten Session Key, der für die Verschlüsselung der Daten benötigt wird.

Die Zeilen 43 bis 52 enthalten das mit dem Session Key verschlüsselte XML Dokument (base64-kodiert).

#### *Downloads zu diesem Beispiel*

- examples/interface/encryptXML/encryptXML_New.xml
- examples/interface/encryptXML/encryptXML_New_Answer.xml

#### *Weiterführende Information*

- Abschnitt 4.2, „Verschlüsselung als XML-Dokument“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer

### 2.3.2.2. Verschlüsselung eines Elements in einem vorhandenen XML Dokument (`Element`)

#### *Anfrage*

In folgendem Beispiel wird ein XML Dokument angegeben, von dem nur ein einzelnes Element verschlüsselt werden soll. Dieses Element wird im Attribut `@selector` des Elements `Element` mit Hilfe von XPATH selektiert. Dabei muss dem `EncryptXMLRequest` jeder Namenraum, der vom angegeben Dokument verwendet wird, bekannt sein.

```
[01]  <?xml version="1.0" encoding="UTF-8"?>
[02]    <sl:EncryptXMLRequest
[03]      xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"
[04]      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
[05]      xmlns:docns="http://www.w3.org/1999/xhtml">
[06]      <sl:RecipientPublicKey>
[07]        <sl:X509Certificate>
[08]          MIIFBDCCA+ygAwIBAgIDASpYMA0GCSqGSIb3DQEBBQUAMIGXMQswCQYDVQQGEwJBVDFIMEYGA1UE
[09]          Cgw/QSlUcnVzdCBHZXMuIGYuIFNpY2hlcmhlaXRzc3lzdGVtZSBpbSBlbGVrdHIuIERhdGVudmVy
[10]          a2VociBHbWJIMR4wHAYDVQQLDBVhLXNpZ24tUHJlbWl1bS1FbmMtMDIxHjAcBgNVBAMMFWEtc2ln
[11]          bi1QcmVtaXVtLUVuYy0wMjAeFw0wNTA0MjEwNjE0MzdaFw0wOTA0MjEwNjE0MzdaMFoxCzAJBgNV
[12]          BAYTAkFUMRQwEgYDVQQDDAtQZXRlciBUZXVmbDEOMAwGA1UEBAwFVGV1ZmwxDjAMBgNVBCoMBVBl
[13]          dGVyMRUwEwYDVQQFEww2NjcxOTI1NzA2MzQwgd8wDQYJKoZIhvcNAQEBBQADgc0AMIHJAoHBALzp
[14]          AKja0OI2iGC+ufp8hMYo/U1iAjIY/HcOgIb+UN+0qL4RmGEt1CTYBcm6t3NIGi9+pVTat0nKmSsH
[15]          qs5NtlZJvahIHrs6q/Nvs6vzLTVHkRwl9CcgsF54MdKz/LzE41yZ+EE07HqW8l69OIXNSVrFS4kG
[16]          oEJUHFGWdke71Kpbfu4+2d2cfU9jMX/rUzBz/fcbeg2IMY3DhI4uH7R492eS5bEmbZnYlSuvK4Em
[17]          3Mx3TFrL8ZOjNOCnfJAuAbf9gwIDAQABo4IB1zCCAdMwEwYDVR0jBAwwCoAIRyFHjpdh4x4wewYI
[18]          KwYBBQUHAQEEbzBtMEIGCCsGAQUFBzAChjZodHRwOi8vd3d3LmEtdHJ1c3QuYXQvY2VydHMvYS1z
[19]          aWduLVByZW1pdW0tRW5jLTAyYS5jcnQwJwYIKwYBBQUHMAGGG2h0dHA6Ly9vY3NwLmEtdHJ1c3Qu
[20]          YXQvb2NzcDBNBgNVHSAERjBEMEIGBiooABEBDDA4MDYGCCsGAQUFBwIBFipodHRwOi8vd3d3LmEt
[21]          dHJ1c3QuYXQvZG9jcy9jcC9hLXNpZ24tdG9rZW4wgZoGA1UdHwSBkjCBjzCBjKCBiaCBhoaBg2xk
[22]          YXA6Ly9sZGFwLmEtdHJ1c3QuYXQvb3U9YS1zaWduLVByZW1pdW0tRW5jLTAyLG89QS1UcnVzdCxj
[23]          PUFUP2NlcnRpZmljYXRlcmV2b2NhdGlvbmxpc3Q/YmFzZT9vYmplY3RjbGFzcz1laWRDZXJ0aWZp
[24]          Y2F0aW9uQXV0aG9yaXR5MBEGA1UdDgQKBAhEc3YCuoW7uDAOBgNVHQ8BAf8EBAMCBLAwJQYDVR0R
[25]          BB4wHIEacGV0ZXIudGV1ZmxAaWFpay50dWdyYXouYXQwCQYDVR0TBAIwADANBgkqhkiG9w0BAQUF
[26]          AAOCAQEAJfg2cBaLMaXVqoZ23UA8qKxTxh+WeSlEveI4Ca43tu89uutJ3w/rXVFJ1EcSaA4OTAJt
[27]          icp5LstzJmrJoTcxbb3gC46x5MrgyvDbiTb/AiHBw11C0GN3pjv1cqFfOMYvuWPbliVPgCdCYqva
[28]          sr5KNWbge9r0tEh6Oogx0zAVrsdSYN30eSECch6NKlptD6L5KRKoorlCIBq7n2U70DpSWFYQHegJ
[29]          ier2yeY5hG6ceIZKKJ/fjDLH2NzWidoXk3NWqc3X836YCnoNyQ0oqgkz6gPSyWTpWnJ+j/WNBouA
[30]          ImEAiehOOgnNBJgS72z5iJsDFcLfI6cX/WmibSp3nR/AMQ==
[31]        </sl:X509Certificate>
[32]      </sl:RecipientPublicKey>
[33]      <sl:ToBeEncrypted>
[34]        <sl:Element
[35]          Selector="/docns:html/docns:head/docns:title">
[36]        </sl:Element>
```

```
[37]      </sl:ToBeEncrypted>
[38]      <sl:EncryptionInfo>
[39]        <sl:EncryptionEnvironment>
[40]          <sl:XMLContent>
[41]            <html xmlns="http://www.w3.org/1999/xhtml">
[42]              <head>
[43]                <title>Ein einfaches SLXHTML-Dokument</title>
[44]                <style type="text/css">p { color: red; }</style>
[45]              </head>
[46]              <body>
[47]                <p>Ich bin ein einfacher Text in rot.</p>
[48]              </body>
[49]            </html>
[50]          </sl:XMLContent>
[51]        </sl:EncryptionEnvironment>
[52]      </sl:EncryptionInfo>
[53]    </sl:EncryptXMLRequest>
```

Zeile 2 enthält den passenden Befehl der Schnittstelle *Security-Layer*.

Die Zeilen 3 bis 5 enthalten die Namenraum-Deklaration für die XML-Elemente, aus denen die Anfrage besteht.

Zeile 5 deklariert dabei den Namenraum, der vom angegebenen XML Dokument (Zeilen 41 bis 49) verwendet wird. Dieser Namenraum wird für den XPATH Ausdruck in Zeile 35 benötigt.

Die Zeilen 6 bis 32 enthalten das base64-kodierte X509 Zertifikat des Empfängers.

Die Zeilen 34 bis 36 geben an, dass nur das Element `title` vom angegebenen XML Dokument (Zeilen 41 bis 49) verschlüsselt werden soll.

Die Zeilen 41 bis 49 enthalten das XML Dokument. Von diesem Dokument soll nur das Element `title` verschlüsselt werden.

### *Antwort*

```
[01]  <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
[02]    <sl:EncryptXMLResponse
[03]      xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]      <sl:EncryptionEnvironment>
[05]        <html
[06]          xmlns="http://www.w3.org/1999/xhtml">
[07]          <head>
[08]            <xenc:EncryptedData
[09]              xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
[10]              Id="encrypted-data-0-1152534185-43290818-11809"
[11]              Type="http://www.w3.org/2001/04/xmlenc#Element">
[12]                <xenc:EncryptionMethod
[13]                  Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
[14]                 <ds:KeyInfo
[15]                    xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
[16]                    <xenc:EncryptedKey
[17]                      Id="encrypted-key-1-1152534185-43290838-5497-c0">
[18]                      <xenc:EncryptionMethod
[19]                        Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
[20]                      <ds:KeyInfo>
[21]                        <ds:KeyValue>
[22]                          <ds:RSAKeyValue>
[23]                            <ds:Modulus>
[24]                              vOkAqNrQ4jaIYL65+nyExij9TWICMhj8dw6Ahv5Q37SovhGYYS3UJNgFybq3
[25]                              c0gaL36lVNq3ScqZKweqzk22Vkm9qEgeuzqr82+zq/MtNUeRHCX0JyCwXngx
[26]                              0rP8vMTjXJn4QTTsepbyXr04hc1JWsVLiQagQlQcUZZ2R7vUqlt+7j7Z3Zx9
[27]                              T2Mxf+tTMHP99xt6DYgxjcOEji4ftHj3Z5LlsSZtmdiVK68rgSbczHdMWsvx
[28]                              k6M04Kd8kC4Bt/2D
[29]                            </ds:Modulus>
[30]                            <ds:Exponent>
[31]                              AQAB
[32]                            </ds:Exponent>
[33]                          </ds:RSAKeyValue>
[34]                        </ds:KeyValue>
[35]                      </ds:KeyInfo>
[36]                    <xenc:CipherData>
[37]                  <xenc:CipherValue>
[38]                    Yqc8dmj2/NgF6He3nhD36NyxkJ5bndlrXsI1M54Fgqeq6B16F02odj/6YYyH
[39]                    ygpmeU6sY9adbnab9Iq3Sbsa/YT7W239F9BWaMd2f2lnVzkro22A2e6xu4sd
[40]                    xGuHbfHdCQeIc8qlDoJK5tQadA4lS8nM37jCG+/Gp8QCIC+UxbF0iz3th6xD
[41]                    +r+K8P+rqXTQcscegpFycOQ6Bjg11HMzslF7W+kx75jcCwKy6/CtRMAI+mRZ
[42]                    GrSrHh66rM11e1Fx
[43]                  </xenc:CipherValue>
[44]                </xenc:CipherData>
[45]              </xenc:EncryptedKey>
[46]            </ds:KeyInfo>
[47]            <xenc:CipherData>
[48]              <xenc:CipherValue>
[49]                k7glMunnGceKGGaG+Ru3+gj+5j7jtrdwb1Dy9ef01hNjTZh/03UhGBIHwGd7
[50]                sZZ+JPoUcBKdKv1/9mvX7i4obcJa3FO0LrGFrWyAMHyIPi2KBlA3Sh4kk/W8
[51]                IJdyiXYY
[52]              </xenc:CipherValue>
[53]            </xenc:CipherData>
[54]          </xenc:EncryptedData>
[55]          <style type="text/css">p { color: red; }</style>
[56]        </head>
[57]        <body>
[58]          <p>Ich bin ein einfacher Text in rot.</p>
[59]        </body>
[60]      </html>
[61]    </sl:EncryptionEnvironment>
[62]  </sl:EncryptXMLResponse>
```

Zeile 2 enthält die passende Antwort der Schnittstelle *Security-Layer*.

Die Zeilen 3 enthält die Namenraum-Deklaration für die XML-Elemente, aus denen die Antwort besteht.

Die Zeilen 5 bis 60 enthalten das in der Anfrage angegebene XML Dokument. Das Element `title` ist in verschlüsselter Form enthalten (Zeilen 8 bis 54).

*Downloads zu diesem Beispiel*

- examples/interface/encryptXML/encryptXML_Element.xml
- examples/interface/encryptXML/encryptXML_Element_Answer.xml

*Weiterführende Information*

- Abschnitt 4.2, „Verschlüsselung als XML-Dokument“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer

### 2.3.2.3. Verschlüsselung des Inhalts eines Elements in einem vorhandenen XML Dokument (`ElementContent`)

*Anfrage*

In folgendem Beispiel wird ein XML Dokument angegeben, von dem nur der Inhalt eines einzelnen Elements verschlüsselt werden soll. Dieses Element wird im Attribut `@selector` des Elements `ElementContent` mit Hilfe von XPATH selektiert. Dabei muss dem `EncryptXMLRequest` jeder Namenraum, der vom angegeben Dokument verwendet wird, bekannt sein. Die Anfrage entspricht bis auf die Verwendung von `ElementContent` anstelle von `Element` der Anfrage aus Abschnitt 2.3.2.2, „Verschlüsselung eines Elements in einem vorhandenen XML Dokument (`Element`)“

```
[01]  <?xml version="1.0" encoding="UTF-8"?>
[02]    <sl:EncryptXMLRequest
[03]      xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"
[04]      xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
[05]      xmlns:docns="http://www.w3.org/1999/xhtml">
[06]      <sl:RecipientPublicKey>
[07]        <sl:X509Certificate>
[08]          MIIFBDCCA+ygAwIBAgIDASpYMA0GCSqGSIb3DQEBBQUAMIGXMQswCQYDVQQGEwJBVDFIMEYGA1UE
[09]          Cgw/QS1UcnVzdCBHZXMuIGYuIFNpY2hlcmhlaXRzc3lzdGVtZSBpbSBlbGVrdHIuIERhdGVudmVy
[10]          a2VociBHbWJIMR4wHAYDVQQLDBVhLXNpZ24tUHJlbWl1bS1FbmMtMDIxHjAcBgNVBAMMFWEtc2ln
[11]          bi1QcmVtaXVtLUVuYy0wMjAeFw0wNTA0MjEwNjE0MzdaFw0wOTA0MjEwNjE0MzdaMFoxCzAJBgNV
[12]          BAYTAkFUMRQwEgYDVQQDDAtQZXRlciBUZXVmbDEOMAwGA1UEBAwFVGV1ZmwxDjAMBgNVBCoMBVBl
[13]          dGVyMRUwEwYDVQQFEww2NjcxOTI1NzA2MzQwgd8wDQYJKoZIhvcNAQEBBQADgc0AMIHJAoHBALzp
[14]          AKja0OI2iGC+ufp8hMYo/U1iAjIY/HcOgIb+UN+0qL4RmGEt1CTYBcm6t3NIGi9+pVTat0nKmSsH
[15]          qs5NtlZJvahIHrs6q/Nvs6vzLTVHkRwl9CcgsF54MdKz/LzE41yZ+EE07HqW8l69OIXNSVrFS4kG
[16]          oEJUHFGWdke71Kpbfu4+2d2cfU9jMX/rUzBz/fcbeg2IMY3DhI4uH7R492eS5bEmbZnYlSuvK4Em
[17]          3Mx3TFrL8ZOjNOCnfJAuAbf9gwIDAQABo4IB1zCCAdMwEwYDVR0jBAwwCoAIRyFHjpdh4x4wewYI
[18]          KwYBBQUHAQEEbzBtMEIGCCsGAQUFBzAChjZodHRwOi8vd3d3LmEtdHJ1c3QuYXQvY2VydHMvYS1z
[19]          aWduLVByZW1pdW0tRW5jLTAyYS5jcnQwJwYIKwYBBQUHMAGGG2h0dHA6Ly9vY3NwLmEtdHJ1c3Qu
[20]          YXQvb2NzcDBNBgNVHSAERjBEMEIGBioooABEBDDA4MDYGCCsGAQUFBwIBFipodHRwOi8vd3d3LmEt
[21]          dHJ1c3QuYXQvZG9jcy9jcC9hLXNpZ24tdG9rZW4wgZoGA1UdHwSBkjCBjzCBjKCBiaCBhoaBg2xk
[22]          YXA6Ly9sZGFwLmEtdHJ1c3QuYXQvb3U9YS1zaWduLVByZW1pdW0tRW5jLTAyLG89QS1UcnVzdCxj
[23]          PUFUP2NlcnRpZmljYXRlcmV2b2NhdGlvbmxpc3Q/YmFzZT9vYmplY3RjbGFzcz1laWRDZXJ0aWZp
[24]          Y2F0aW9uQXV0aG9yaXR5MBEGA1UdDgQKBAhEc3YCuoW7uDAOBgNVHQ8BAf8EBAMCBLAwJQYDVR0R
[25]          BB4wHIEacGV0ZXIudGV1ZmxAaWFpay50dWdyYXouYXQwCQYDVR0TBAIwADANBgkqhkiG9w0BAQUF
[26]          AAOCAQEAJfg2cBaLMaXVqoZ23UA8qKxTxh+WeSlEveI4Ca43tu89uutJ3w/rXVFJ1EcSaA4OTAJt
[27]          icp5LstzJmrJoTcxbb3gC46x5MrgyvDbiTb/AiHBw11C0GN3pjv1cqFfOMYvuWPb1iVPgCdCYqva
[28]          sr5KNWbge9r0tEh6Oogx0zAVrsdSYN30eSECch6NK1ptD6L5KRKoorlCIBq7n2U70DpSWFYQHegJ
[29]          ier2yeY5hG6ceIZKKJ/fjDLH2NzWidoXk3NWqc3X836YCnoNyQ0oqgkz6gPSyWTpWnJ+j/WNBouA
[30]          ImEAiehOOgnNBJgS72z5iJsDFcLfI6cX/WmibSp3nR/AMQ==
[31]        </sl:X509Certificate>
[32]      </sl:RecipientPublicKey>
[33]      <sl:ToBeEncrypted>
[34]        <sl:ElementContent
[35]          Selector="/docns:html/docns:head/docns:title">
[36]        </sl:ElementContent>
[37]      </sl:ToBeEncrypted>
[38]      <sl:EncryptionInfo>
[39]        <sl:EncryptionEnvironment>
[40]          <sl:XMLContent>
[41]            <html xmlns="http://www.w3.org/1999/xhtml">
[42]              <head>
[43]                <title>Ein einfaches SLXHTML-Dokument</title>
[44]                <style type="text/css">p { color: red; }</style>
[45]              </head>
[46]              <body>
[47]                <p>Ich bin ein einfacher Text in rot.</p>
[48]              </body>
[49]            </html>
[50]          </sl:XMLContent>
[51]        </sl:EncryptionEnvironment>
[52]      </sl:EncryptionInfo>
[53]    </sl:EncryptXMLRequest>
```

Zeile 2 enthält den passenden Befehl der Schnittstelle *Security-Layer*.

Die Zeilen 3 bis 5 enthalten die Namenraum-Deklaration für die XML-Elemente, aus denen die Anfrage besteht.

Zeile 5 deklariert dabei den Namenraum, der vom angegebenen XML Dokument (Zeilen 41 bis 49) verwendet wird. Dieser Namenraum wird für den XPATH Ausdruck in Zeile 35 benötigt.

Die Zeilen 6 bis 32 enthalten das base64-kodierte X509 Zertifikat des Empfängers.

Die Zeilen 34 bis 36 geben an, dass nur der Inhalt des Elements `title` vom angegebenen XML Dokument (Zeilen 41 bis 49) verschlüsselt werden soll.

Die Zeilen 41 bis 49 enthalten das XML Dokument. Von diesem Dokument soll nur der Inhalt des Elements `title` verschlüsselt werden.

*Antwort*

```
[01]  <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
[02]    <sl:EncryptXMLResponse
[03]      xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]      <sl:EncryptionEnvironment>
[05]        <html
[06]          xmlns="http://www.w3.org/1999/xhtml">
[07]          <head>
[08]            <title>
[09]              <xenc:EncryptedData
[10]                xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
[11]                Id="encrypted-data-0-1152532362-41467517-23174"
```

```
[12]                Type="http://www.w3.org/2001/04/xmlenc#Content">
[13]                <xenc:EncryptionMethod
[14]                  Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
[15]                <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
[16]                  <xenc:EncryptedKey
[17]                    Id="encrypted-key-1-1152532362-41467527-29158-c0">
[18]                    <xenc:EncryptionMethod
[19]                      Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
[20]                    <ds:KeyInfo>
[21]                      <ds:KeyValue>
[22]                        <ds:RSAKeyValue>
[23]                          <ds:Modulus>
[24]                            vOkAqNrQ4jaIYL65+nyExij9TWICMhj8dw6Ahv5Q37SovhGYYS3UJNgFybq3
[25]                            c0gaL36lVNq3ScqZKweqzk22Vkm9qEgeuzqr82+zq/MtNUeRHCX0JyCwXngx
[26]                            0rP8vMTjXJn4QTTsepbyXr04hc1JWsVLiQagQlQcUZZ2R7vUqlt+7j7Z3Zx9
[27]                            T2Mxf+tTMHP99xt6DYgxjcOEji4ftHj3Z5LlsSZtmdiVK68rgSbczHdMWsvx
[28]                            k6M04Kd8kC4Bt/2D
[29]                          </ds:Modulus>
[30]                          <ds:Exponent>
[31]                            AQAB
[32]                          </ds:Exponent>
[33]                        </ds:RSAKeyValue>
[34]                      </ds:KeyValue>
[35]                    </ds:KeyInfo>
[36]                    <xenc:CipherData>
[37]                      <xenc:CipherValue>
[38]                        MDA77eIm+HXZnVkMAl3Ox858BAG7fSeLGTIcmtm/dKjp8Sk2M12RN7xqvIoP
[39]                        LsYab8ddAJktE/s+Tk1MKzGPdlvfHZInu4OVjKQfHOReuic8ndmjc8K2kBot
[40]                        uNz0WTKAEOQ1l2zgNBVKnbeFzI2ozrO1uHBfeR2t+pu92mp1L8FvATW/+tD/
[41]                        7AAwRROxZut2jFrmmw/ajfUWMtNm+8gtpwqdx12N03tbW9tihKlYKgKspOL6
[42]                        GGPYBysIjl39KbTq
[43]                      </xenc:CipherValue>
[44]                    </xenc:CipherData>
[45]                  </xenc:EncryptedKey>
[46]                </ds:KeyInfo>
[47]                <xenc:CipherData>
[48]                  <xenc:CipherValue>
[49]                    NhUqASe+jJ0BHqTX4sayQLz7qUNbO8Wdj9qEI4wm+9Mbml3Agfjluw==
[50]                  </xenc:CipherValue>
[51]                </xenc:CipherData>
[52]              </xenc:EncryptedData>
[53]            </title>
[54]            <style type="text/css">p { color: red; }</style>
[55]          </head>
[56]          <body>
[57]            <p>Ich bin ein einfacher Text in rot.</p>
[58]          </body>
[59]        </html>
[60]      </sl:EncryptionEnvironment>
[61]    </sl:EncryptXMLResponse>
```

Zeile 2 enthält die passende Antwort der Schnittstelle *Security-Layer*.

Die Zeile 3 enthält die Namenraum-Deklaration für die XML-Elemente, aus denen die Antwort besteht.

Die Zeilen 5 bis 59 enthalten das in der Anfrage angegebene XML Dokument. Der Inhalt des Elements `title` ist in verschlüsselter Form enthalten (Zeilen 9 bis 52).

### *Downloads zu diesem Beispiel*

- examples/interface/encryptXML/encryptXML_ElementContent.xml
- examples/interface/encryptXML/encryptXML_ElementContent_Answer.xml

### *Weiterführende Information*

- Abschnitt 4.2, „Verschlüsselung als XML-Dokument“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer

### *Anfrage*

Daten, die im CMS Format verschlüsselt sind, werden unter Verwendung des Befehls DecryptCMSRequest entschlüsselt. Dabei werden die zu entschlüsselnden Daten base64-kodiert im Element `sl:CMSMessage` übergeben. Alternativ können die zu entschlüsselnden Daten im Attribut `sl:Content/@Reference` als Referenz übergeben werden.

Weiters werden im Element `sl:EncryptedContent` die Metainformationen zu den zu verschlüsselnden Daten angegeben. Dabei kann das Element `sl:EncryptedContent/sl:MetaInfo/sl:MimeType` angeben werden. Diese Information wird von der *Bürgerkarte* verwendet, um die Anzeige der entschlüsselten Daten zu ermöglichen. Optional kann im Element `sl:ToBeEncrypted/sl:MetaInfo/sl:Description` eine verbale Beschreibung der Daten angegeben werden.

```
[01]  <?xml version="1.0" encoding="UTF-8"?>
[02]    <sl:DecryptCMSRequest
[03]      xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]      <sl:CMSMessage>
[05]        MIIB3AYJKoZIhvcNAQcDoIIBzTCCAckCAQExggF7MIIBdwIBATCBnzCBlzEL
[06]        MAkGA1UEBhMCQVQxSDBGBgNVBAoMP0EtVHJ1c3QgR2VzLiBmLiBTaWNoZXJo
[07]        ZWl0c3N5c3RlbWUgaW0gZWxla3RyLiBEYXRlbnZlcmtlaHIgR21iSDEeMBwG
[08]        A1UECwwVYS1zaWduLVByZW1pdW0tRW5jLTAyMR4wHAYDVQQDDBVhLXNpZ24t
[09]        UHJlbWl1bS1FbmMtMDICAwEqWDANBgkqhkiG9w0BAQEFAASBwBiwSIHzq6LK
[10]        4RcT6wrA6TuJAHgsVRtirQQhMkvgSWyozC5SJoyYDVuqFgci+0MwBioPp7H6
[11]        gv0m2RAp3p7MdyaUBY7WzC9X5anTcioCuI1E4EoQJGyg+rUD7PzrRl/HroP3
[12]        EEzGK7jZCJ9ToWmleMMYsLmtkMJ5MlnRdtyuReLU8ATfGCJOMSJlUDuiVqmU
[13]        UOSO/l8M6AyXz7ZJ7ntgf6IJtOo0G/f5Ph/smWWXltKD2nWxzJUUaXs75lfB
[14]        +/KfTjBFBgkqhkiG9w0BBwEwFAYIKoZIhvcNAwcECPEnrwVJxQ54oCIEIBcB
[15]        bbIEsKpuWsUxFFPBBjTJtV8rVFXfpTBFuC03ltTo
[16]      </sl:CMSMessage>
[17]      <sl:EncryptedContent>
[18]        <sl:MetaInfo>
[19]          <sl:MimeType>text/plain</sl:MimeType>
[20]          <sl:Description>
```

```
[21]            Diese Daten liegen als reiner Text vor.
[22]          </sl:Description>
[23]        </MetaInfo>
[24]      </sl:EncryptedContent>
[25]    </sl:DecryptCMSRequest>
```

Zeile 2 enthält den passenden Befehl der Schnittstelle *Security-Layer*.

Zeile 3 enthält die Namenraum-Deklaration für die XML-Elemente, aus denen die Anfrage besteht.

Die Zeilen 4 bis 16 enthalten die base64-kodierte CMS Nachricht, die entschlüsselt werden soll.

Die Zeilen 17 bis 24 geben Metainformationen an, die für Anzeige der zu verschlüsselnden Daten verwendet werden.

### *Antwort*

Die Antwort enthält die entschlüsselten Daten im Base64 Format im Element `sl:DecryptedData` .

```
[01]  <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
[02]  <sl:DecryptCMSResponse
[03]    xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]    <sl:DecryptedData>
[05]      SWNoIGJpbiBlaW4gZWluZmFjaGVyIFRleHQu
[06]    </sl:DecryptedData>
[05]  </sl:DecryptCMSResponse>
```

### *Downloads zu diesem Beispiel*

- examples/interface/decryptCMS/decryptCMS_Base64.xml
- examples/interface/decryptCMS/decryptCMS_Base64_Answer.xml

### *Weiterführende Information*

- Abschnitt 5.1, „Entschlüsselung einer CMS-Nachricht“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer

### *Anfrage*

Daten, die nach dem XMLENC Standard verschlüsselt sind, werden unter Verwendung des Befehls DecryptXMLRequest entschlüsselt. Dabei werden die zu entschlüsselnden Daten entweder base64-kodiert im Element `sl:EncryptedContent/sl:Base64Content` oder als XML Dokument im Element `sl:EncryptedContent/sl:XMLContent` übergeben. Alternativ können die zu entschlüsselnden Daten im Attribut `sl:EncryptedContent/sl:Content/@Reference` als Referenz übergeben werden.

Im Element `sl:EncrElemeSelector` wird angegeben welche verschlüsselten Elemente von der Bürgerkarten-Umgebung tatsächlich entschlüsselt werden sollen. Sein Inhalt spezifiziert einen Ausdruck nach XPATH, dessen Auswertung eine Knotenmenge mit beliebig vielen Elementen `xenc:EncryptedData` ergeben muss.

```
[01]  <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
[02]    <sl:DecryptXMLRequest
[03]      xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]      <sl:EncryptedContent>
[05]        <sl:XMLContent>
[06]          <xenc:EncryptedData
[07]              xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
[08]              Id="encrypted-data-0-1152184915-1418099-32011">
[09]            <xenc:EncryptionMethod
[10]                Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
[11]            <ds:KeyInfo
[12]                xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
[13]            <xenc:EncryptedKey
[14]                Id="encrypted-key-1-1152184915-1418139-8806-c0">
[15]              <xenc:EncryptionMethod
[16]                  Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
[17]              <ds:KeyInfo>
[18]                  <ds:KeyValue>
[19]                  <ds:RSAKeyValue>
[20]                    <ds:Modulus>
[21]                      vOkAqNrQ4jaIYL65+nyExij9TWICMhj8dw6Ahv5Q37SovhGYYS3UJNgFybq3
[22]                      c0gaL36lVNq3ScqZKweqzk22Vkm9qEgeuzqr82+zq/MtNUeRHCX0JyCwXngx
[23]                      0rP8vMTjXJn4QTTsepbyXr04hc1JWsVLiQagQlQcUZZ2R7vUqlt+7j7Z3Zx9
[24]                      T2Mxf+tTMHP99xt6DYgxjcOEji4ftHj3Z5LlsSZtmdiVK68rgSbczHdMWsvx
[25]                      k6M04Kd8kC4Bt/2D
[26]                    </ds:Modulus>
[27]                    <ds:Exponent>AQAB</ds:Exponent>
[28]                  </ds:RSAKeyValue>
[29]                </ds:KeyValue>
[30]              </ds:KeyInfo>
[31]              <xenc:CipherData>
[32]                <xenc:CipherValue>
[33]                  HTwgNQCRkfvU7dqlV/83mfkTevsFn8HTek54nQD+Erno/4IWWTn83riXF5i+
[34]                  u1y53bqiwXDduOmMzPsQj/8q2EuqsWzvEQm+aKItVXyX1AqXt11NEVCoR62Q
[35]                  qV+WcH61GM35swC92Ohe0S8hXLsaQUCgQHq7klzBjkXeLRFLCchZjsgc3Miy
[36]                  lIZdsNcZvZYsMZK0kpLiscH/WRrklSZdTT3tJwQqilSyHAFOz9AOCFai5p3b
[37]                  gsWdblZWm65w8vJe
[38]                </xenc:CipherValue>
[39]              </xenc:CipherData>
[40]            </xenc:EncryptedKey>
[41]          </ds:KeyInfo>
[42]          <xenc:CipherData>
[43]            <xenc:CipherValue>
[44]              BU4x6VaAFS4g9SJwDhoFK7MfRnr7CqAEqOnh1j0FuN/fJA4p8OJWtw==
[45]            </xenc:CipherValue>
[46]          </xenc:CipherData>
[47]        </xenc:EncryptedData>
[48]      </sl:XMLContent>
[49]    </sl:EncryptedContent>
```

```
[50]    <sl:EncrElemsSelector>
[51]      //*[@Id='encrypted-data-0-1151396325-49021018-4645']
[52]    </sl:EncrElemsSelector>
[53]  </sl:DecryptXMLRequest>
```

Zeile 2 enthält den passenden Befehl der Schnittstelle *Security-Layer*.

Zeile 3 enthält die Namenraum-Deklaration für die XML-Elemente, aus denen die Anfrage besteht.

Die Zeilen 6 bis 47 enthalten die verschlüsselte Nachricht (entsprechend dem XMLENC Standard)

Die Zeilen 49 bis 51 verwenden das Element `EncrElemsSelector`, um anzugeben welche Daten entschlüsselt werden sollen.

*Antwort*

```
[01]  <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
[02]    <sl:DecryptXMLResponse
[03]      xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]      <sl:CandidateDocument>
[05]        <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
[06]          Id="encrypted-data-0-1152184915-1418099-32011">
[07]          <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
[08]            <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
[09]              <xenc:EncryptedKey Id="encrypted-key-1-1152184915-1418139-8806-c0">
[10]                <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
[11]                <ds:KeyInfo>
[12]                  <ds:KeyValue>
[13]                    <ds:RSAKeyValue>
[14]                      <ds:Modulus>
[15]                        vOkAqNrQ4jaIYL65+nyExij9TWICMhj8dw6Ahv5Q37SovhGYYS3UJNgFybq3
[16]                        c0gaL36lVNq3ScqZKweqzk22Vkm9qEgeuzqr82+zq/MtNUeRHCX0JyCwXngx
[17]                        0rP8vMTjXJn4QTTsepbyXr04hc1JWsVLiQagQlQcUZZ2R7vUqlt+7j7Z3Zx9
[18]                        T2Mxf+tTMHP99xt6DYgxjcOEji4ftHj3Z5LlsSZtmdiVK68rgSbczHdMWsvx
[19]                        k6M04Kd8kC4Bt/2D
[20]                      </ds:Modulus>
[21]                      <ds:Exponent>
[22]                        AQAB
[23]                      </ds:Exponent>
[24]                    </ds:RSAKeyValue>
[25]                  </ds:KeyValue>
[26]                </ds:KeyInfo>
[27]                <xenc:CipherData>
[28]                  <xenc:CipherValue>
[29]                    HTwgNQCRkfvU7dqlV/83mfkTevsFn8HTek54nQD+Erno/4IWWTn83riXF5i+
[30]                    u1y53bqiwXDduOmMzPsQj/8q2EuqsWzvEQm+aKItVXyX1AqXt11NEVCoR62Q
[31]                    qV+WcH61GM35swC92Ohe0S8hXLsaQUCgQHq7klzBjkXeLRFLCchZjsgc3Miy
[32]                    lIZdsNcZvZYsMZK0kpLiscH/WRrklSZdTT3tJwQqilSyHAFOz9AOCFai5p3b
[33]                    gsWdblZWm65w8vJe
[34]                  </xenc:CipherValue>
[35]                </xenc:CipherData>
[36]              </xenc:EncryptedKey>
[37]            </ds:KeyInfo>
[38]            <xenc:CipherData>
[39]              <xenc:CipherValue>
[40]                BU4x6VaAFS4g9SJwDhoFK7MfRnr7CqAEqOnh1j0FuN/fJA4p8OJWtw==
[41]              </xenc:CipherValue>
[42]            </xenc:CipherData>
[43]          </xenc:EncryptedData>
[44]        </sl:CandidateDocument>
[45]      <sl:DecryptedBinaryData
[46]        EncrElemSelector="//*[@Id='encrypted-data-0-1152184915-1418099-32011']">
[47]        SWNoIGJpbiBlaW4gZWluZmFjaGVyIFRleHQu
[48]      </sl:DecryptedBinaryData>
[49]    </sl:DecryptXMLResponse>
```

Zeile 2 enthält den passenden Befehl der Schnittstelle *Security-Layer*.

Zeile 3 enthält die Namenraum-Deklaration für die XML-Elemente, aus denen die Anfrage besteht.

Die Zeilen 6 bis 44 enthalten die verschlüsselte Nachricht (entsprechend dem XMLENC Standard)

Die Zeilen 45 bis 48 enthalten die entschlüsselten Daten im Base64 Format..

*Downloads zu diesem Beispiel*

- examples/interface/decryptXML/decryptXML_XML.xml
- examples/interface/decryptXML/decryptXML_XML_Answer.xml

*Weiterführende Information*

- Abschnitt 5.2, „Entschlüsselung eines XML-Dokuments“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer

## 2.4. Daten entschlüsseln

### 2.4.1. Daten im CMS Format entschlüsseln

### 2.4.2. Daten im XML Format entschlüsseln

## 2.5. Hashwert berechnen

### 2.5.1. Ein erstes Beispiel

Dieses Beispiel demonstriert die Erzeugung von Hashwerten für beliebige Daten. Mit einem Request können gleichzeitig auch Hashwerte für mehrere Daten angefordert werden. Die Angabe der Daten, über die ein Hashwert berechnet werden soll, kann ähnlich wie bei den zuvor behandelten Befehlen entweder explizit im Request oder per Referenz mittels URL erfolgen.

#### 2.5.1.1. Anfrage

Mit der folgenden Anfrage wird von der *Bürgerkarten-Umgebung* die Berechnung von zwei Hashwerten angefordert.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:CreateHashRequest xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[03]   <sl:HashInfo RespondHashData="true">
[04]     <sl:HashData>
[05]       <sl:MetaInfo>
[06]         <sl:MimeType>text/plain</sl:MimeType>
[07]       </sl:MetaInfo>
[08]       <sl:Content>
[09]         <sl:XMLContent>Hasch' mich, ich bin der Frühling!</sl:XMLContent>
[10]       </sl:Content>
[11]     </sl:HashData>
[12]     <sl:HashAlgorithm>http://www.w3.org/2000/09/xmldsig#sha1</sl:HashAlgorithm>
[13]     <sl:FriendlyName>Aktueller Stimmungsbericht :-)</sl:FriendlyName>
[14]   </sl:HashInfo>
```

Die Zeilen 3 - 14 enthalten alle Angaben zur Berechnung des ersten Hashwertes.

Mit dem Attribut `sl:RespondHashData` in Zeile 3 wird bestimmt, ob die Daten, über welche der Hashwert berechnet werden soll, in der Antwort zurückgeliefert werden sollen. In diesem Fall ist der Wert `true`, d. h. die XML-Struktur der Zeilen 4 - 11 wird sich ident in der Antwort wiederfinden.

Die Zeilen 4 - 11 enthalten die Angaben zu den Daten, über die der Hashwert berechnet werden soll. An Metainformationen zu diesen Daten muss - wie oben in Zeile 6 - jedenfalls der *Mime Type* angegeben werden. Damit kann die *Bürgerkarten-Umgebung* die richtige Anzeigekomponente auswählen, wenn der *Bürger* die zu hashenden Daten Abschnitt 3.5, „Hashwert-Berechnung“ in Die österreichische Bürgerkarte - Anforderungen an die Benutzer-Schnittstelle. Weiters müssen mit dem Element `sl:Content` die zu hashenden Daten selbst spezifiziert werden. Die Angabe kann dabei explizit als Menge von XML-Knoten - wie hier in den Zeilen 9 - 11 als ein einfacher Textknoten - im Element `sl:XMLContent` bzw. in base64-kodierter Form im Element `sl:Base64Content` erfolgen, oder aber per URL-Referenz mit dem Attribut `sl:Content/@Reference` (vergleiche weiter unten).

Mit dem Element `sl:HashAlgorithm` in Zeile 12 wird der von der *Bürgerkarten-Umgebung* zu verwendende Abschnitt 8.1.1, „Digest-Algorithmen“ in Die österreichische Bürgerkarte - Minimale Umsetzung des Security-Layers bestimmt. Der hier verwendete Wert identifiziert *SHA-1*.

Optional kann das Element `sl:FriendlyName` angegeben werden. Der darin enthaltene, beliebig wählbare Text wird von der *Bürgerkarten-Umgebung* verwendet, um dem *Bürger* die spätere Abschnitt 3.5, „Hashwert-Berechnung“ in Die österreichische Bürgerkarte - Anforderungen an die Benutzer-Schnittstelle eines im Laufe einer Sitzung der *Bürgerkarten-Umgebung* berechneten Hashwerts zu erleichtern. Es wird daher dringend empfohlen, dieses Element zu verwenden und einen prägnanten Text als Wert zu wählen.

```
[15]   <sl:HashInfo RespondHashData="false">
[16]     <sl:HashData>
[17]       <sl:MetaInfo>
[18]         <sl:MimeType>image/png</sl:MimeType>
[19]       </sl:MetaInfo>
[20]       <sl:Content Reference="http://www.buergerkarte.at/konzept/securitylayer/\
[21]         spezifikation/20040514/tutorial/examples/interface/common/ModellBuergerkarte.png"/>
[22]     </sl:HashData>
[23]     <sl:HashAlgorithm>http://www.w3.org/2000/09/xmldsig#sha1</sl:HashAlgorithm>
[24]     <sl:FriendlyName>Grafik Modell Bürgerkarte</sl:FriendlyName>
[25]   </sl:HashInfo>
[26] </sl:CreateHashRequest>
```

Die Zeilen 15 - 25 enthalten alle Angaben zur Berechnung des zweiten Hashwertes.

In diesem Fall soll der Hashwert über ein Bild im Format *PNG* berechnet werden. Dementsprechend wird der *Mime Type* in Zeile 18 gesetzt. Die Bilddaten werden im Gegensatz zum ersten Teil des Beispiels nicht direkt, sondern mittels des Attributs `sl:Content/@Reference` (Zeile 20 - 21) referenziert. Die *Bürgerkarten-Umgebung* wird diese Referenz auflösen, um zu den zu hashenden Daten zu gelangen.

Der Wert des Attributs `sl:RespondHashData` in Zeile 15 ist auf den Wert `false` gesetzt, d. h. in der Antwort wird die XML-Struktur der Zeilen 16 - 22 nicht gespiegelt werden.

#### 2.5.1.2. Antwort

Die entsprechende Antwort der *Bürgerkarten-Umgebung* sollte in etwa wie folgt aussehen. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert wurde.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:CreateHashResponse
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]   <sl:HashInfo>
[05]     <sl:HashData>
[06]       <sl:MetaInfo>
[07]         <sl:MimeType>text/plain</sl:MimeType>
[08]       </sl:MetaInfo>
[09]       <sl:Content>
[10]         <sl:XMLContent>Hasch' mich, ich bin der Frühling!</sl:XMLContent>
[11]       </sl:Content>
[12]     </sl:HashData>
[13]     <sl:HashAlgorithm>http://www.w3.org/2000/09/xmldsig#sha1</sl:HashAlgorithm>
[14]     <sl:FriendlyName>Aktueller Stimmungsbericht :-)</sl:FriendlyName>
[15]     <sl:HashValue>+pWF1F0/8NpbRqBkRpREjK4p7LE=</sl:HashValue>
[16]   </sl:HashInfo>
```

In der Anfrage wurde die Berechnung von zwei Hashwerten angefordert. Dementsprechend enthält die Antwort zwei `sl:HashInfo` Elemente, also eines je berechnetem Hashwert. Die Reihenfolge der `sl:HashInfo` Elemente entspricht dabei jener der `sl:HashInfo` Elemente der Anfrage.

Die Zeilen 4 - 16 enthalten die Informationen zum ersten berechneten Hashwert. Nachdem in der Anfrage in Zeile 3 das Attribut `RespondHashData` auf `true` gesetzt wurde, enthält `sl:HashInfo` als erstes Kind das Element `sl:HashData`, und zwar genau so, wie es auch in der Anfrage angegeben wurde. Die Elemente `sl:HashAlgorithm` in Zeile 13 sowie ggf. `sl:FriendlyName` in Zeile 14 werden ebenfalls exakt aus der Anfrage übernommen.

Das Element `sl:HashValue` in Zeile 15 enthält schließlich den von der *Bürgerkarten-Umgebung* über die zu hashenden Daten berechenten Hashwert.

> **Anmerkung**
>
> Nachdem die zu hashenden Daten in diesem Beispiel als XML-Struktur in `sl:XMLContent` spezifiziert wurden, hat die Bürgerkarten-Umgebung die XML-Struktur Abschnitt 6.1.1, „Anfrage“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer, um zu einer eindeutige Darstellung als Folge von Bytes zu gelangen.

```
[17]   <sl:HashInfo>
[18]     <sl:HashAlgorithm>http://www.w3.org/2000/09/xmldsig#sha1</sl:HashAlgorithm>
[19]     <sl:FriendlyName>Grafik Modell BÃ¼rgerkarte</sl:FriendlyName>
[10]     <sl:HashValue>BLMp3QLSXqO7qEO7N4pEnpzkqPo=</sl:HashValue>
```

```
[21]   </sl:HashInfo>
[22] </sl:CreateHashResponse>
```

Die Zeilen 17 - 21 enthalten die Informationen zum zweiten berechneten Hashwert. Nachdem in der Anfrage in Zeile 15 das Attribut `RespondHashData` auf `false` gesetzt wurde, fehlt in diesem Fall das Element `sl:HashData` als Kind von `sl:HashInfo`.

#### 2.5.1.3. Downloads zu diesem Beispiel

- examples/interface/createHash/First.xml
- examples/interface/common/ModellBuergerkarte.png
- examples/interface/createHash/First.Response.xml

#### 2.5.1.4. Weiterführende Informationen

- Abschnitt 6.1, „Hashwert-Berechnung“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 3.5, „Hashwert-Berechnung“ in Die österreichische Bürgerkarte - Anforderungen an die Benutzer-Schnittstelle
- Abschnitt 8.1.1, „Digest-Algorithmen“ in Die österreichische Bürgerkarte - Minimale Umsetzung des Security-Layers

## 2.6. Hashwert verifizieren

### 2.6.1. Ein erstes Beispiel

Dieses Beispiel demonstriert die Verifikation von Hashwerten über beliebige Daten. Mit einem Request können gleichzeitig auch Hashwerte für mehrere Daten überprüft werden. Die Angabe der Daten, über die ein Hashwert nachgerechnet werden soll, kann ähnlich wie bei den zuvor behandelten Befehlen entweder explizit im Request oder per Referenz mittels URL erfolgen.

#### 2.6.1.1. Anfrage

Mit der folgenden Anfrage wird von der *Bürgerkarten-Umgebung* die Verifikation der im Abschnitt 2.5.1, „Ein erstes Beispiel“ errechneten Hashwerte angefordert.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:VerifyHashRequest xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[03]   <sl:HashInfo>
[04]     <sl:HashData>
[05]       <sl:MetaInfo>
[06]         <sl:MimeType>text/plain</sl:MimeType>
[07]       </sl:MetaInfo>
[08]       <sl:Content>
[09]         <sl:XMLContent>Hasch' mich, ich bin der Frühling!</sl:XMLContent>
[10]       </sl:Content>
[11]     </sl:HashData>
[12]     <sl:HashAlgorithm>http://www.w3.org/2000/09/xmldsig#sha1</sl:HashAlgorithm>
[13]     <sl:FriendlyName>Aktueller Stimmungsbericht :-)</sl:FriendlyName>
[14]     <sl:HashValue>+pWF1F0/8NpbRqBkRpREjK4p7LE=</sl:HashValue>
[15]   </sl:HashInfo>
```

Die Zeilen 3 - 15 enthalten alle Angaben zur Verifikation des ersten Hashwertes.

Die Zeilen 4 - 11 enthalten die Angaben zu den Daten, über die der Hashwert nachgerechnet werden soll. An Metainformationen zu diesen Daten muss - wie oben in Zeile 6 - jedenfalls der *Mime Type* angegeben werden. Damit kann die *Bürgerkarten-Umgebung* die richtige Anzeigekomponente auswählen, wenn der *Bürger* die Daten, über die der Hashwert nachgerechnet werden soll, Abschnitt 3.6, „Hashwert-Verifikation“ in Die österreichische Bürgerkarte - Anforderungen an die Benutzer-Schnittstelle. Weiters müssen mit dem Element `sl:Content` die Daten selbst spezifiziert werden. Die Angabe kann dabei explizit als Menge von XML-Knoten - wie hier in den Zeilen 9 - 11 als ein einfacher Textknoten - im Element `sl:XMLContent` bzw. in base64-kodierter Form im Element `sl:Base64Content` erfolgen, oder aber per URL-Referenz mit dem Attribut `sl:Content/@Reference` (vergleiche weiter unten).

Mit dem Element `sl:HashAlgorithm` in Zeile 12 wird der von der *Bürgerkarten-Umgebung* zu verwendende Abschnitt 8.2.1, „Digest-Algorithmen“ in Die österreichische Bürgerkarte - Minimale Umsetzung des Security-Layers bestimmt. Der hier verwendete Wert identifiziert *SHA-1*.

Optional kann das Element `sl:FriendlyName` angegeben werden. Der darin enthaltene, beliebig wählbare Text wird von der *Bürgerkarten-Umgebung* verwendet, um dem *Bürger* die spätere Abschnitt 3.6, „Hashwert-Verifikation“ in Die österreichische Bürgerkarte - Anforderungen an die Benutzer-Schnittstelle eines im Laufe einer Sitzung der *Bürgerkarten-Umgebung* verifizierten Hashwerts zu erleichtern. Es wird daher dringend empfohlen, dieses Element zu verwenden und einen prägnanten Text als Wert zu wählen.

Das Element `sl:HashValue` in Zeile 14 enthält schließlich den Referenz-Hashwert, den die *Bürgerkarten-Umgebung* überprüfen soll.

```
[16]   <sl:HashInfo>
[17]     <sl:HashData>
[18]       <sl:MetaInfo>
[19]         <sl:MimeType>image/png</sl:MimeType>
[20]       </sl:MetaInfo>
[21]       <sl:Content Reference="http://www.buergerkarte.at/konzept/securitylayer/spezifikation/\
[22]         20040514/tutorial/examples/interface/common/ModellBuergerkarte.png"/>
[23]     </sl:HashData>
[24]     <sl:HashAlgorithm>http://www.w3.org/2000/09/xmldsig#sha1</sl:HashAlgorithm>
[25]     <sl:FriendlyName>Grafik Modell Bürgerkarte</sl:FriendlyName>
[26]     <sl:HashValue>BLMp3QLSXqO7qEO7N4pEnpzkqPo=</sl:HashValue>
[27]   </sl:HashInfo>
[28] </sl:VerifyHashRequest>
```

Die Zeilen 15 - 25 enthalten alle Angaben zur Verifikation des zweiten Hashwertes.

In diesem Fall soll der Hashwert über ein Bild im Format *PNG* nachgerechnet werden. Dementsprechend wird der *Mime Type* in Zeile 19 gesetzt. Die Bilddaten werden im Gegensatz zum ersten Teil des Beispiels nicht direkt, sondern mittels des Attributs `sl:Content/@Reference` (Zeile 21 - 22) referenziert. Die *Bürgerkarten-Umgebung* wird diese Referenz auflösen, um zu den Bilddaten zu gelangen.

#### 2.6.1.2. Antwort

Die entsprechende Antwort der *Bürgerkarten-Umgebung* sollte in etwa wie folgt aussehen. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert wurde.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:VerifyHashResponse xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[03]   <sl:VerificationResult>
[04]     <sl:FriendlyName>Aktueller Stimmungsbericht :-)</sl:FriendlyName>
```

```
[05]     <sl:Result>1</sl:Result>
[06]   </sl:VerificationResult>
[07]   <sl:VerificationResult>
[08]     <sl:FriendlyName>Grafik Modell BÃ¼rgerkarte</sl:FriendlyName>
[09]     <sl:Result>1</sl:Result>
[10]   </sl:VerificationResult>
[11] </sl:VerifyHashResponse>
```

Die Antwort enthält je Hashwert, der in der Anfrage zur Verifikation eingereicht worden ist, ein korrespondierendes Element `sl:VerificationResult`; im konkreten Fall sind das zwei solche Elemente. Die Reihenfolge korrespondiert mit jener der `sl:HashInfo` Elemente der Anfrage.

`sl:VerificationResult` enthält zunächst optional das Element `sl:FriendlyName`, und zwar genau dann, wenn dieses Element auch im korrespondierenden `sl:HashInfo` der Anfrage enthalten war. In `sl:Result` ist das Ergebnis der Überprüfung des Hashwertes enthalten. Konnte der Wert erfolgreich verifiziert werden, enthält es den Wert `true` (oder gleichwertig `1`), ansonsten `false` (oder gleichwertig `0`).

#### 2.6.1.3. Downloads zu diesem Beispiel

- examples/interface/verifyHash/First.xml
- examples/interface/common/ModellBuergerkarte.png
- examples/interface/verifyHash/First.Response.xml

#### 2.6.1.4. Weiterführende Informationen

- Abschnitt 6.2, „Hashwert-Verifikation“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 3.6, „Hashwert-Verifikation“ in Die österreichische Bürgerkarte - Anforderungen an die Benutzer-Schnittstelle
- Abschnitt 8.2.1, „Digest-Algorithmen“ in Die österreichische Bürgerkarte - Minimale Umsetzung des Security-Layers

## 2.7. Infoboxen

### 2.7.1. Auslesen verfügbarer Infoboxen

Dieses Beispiel demonstriert das Auslesen der Bezeichner für alle in der *Bürgerkarten-Umgebung* verfügbaren Infoboxen. Diese Bezeichner können dann in den Befehlen zum Lesen, Verändern und Löschen von Infoboxen verwendet werden.

#### 2.7.1.1. Anfrage

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:InfoboxAvailableRequest
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"/>
```

Die Anfrage besteht aus dem leeren Element `sl:InfoboxAvailableRequest`.

#### 2.7.1.2. Antwort

Die entsprechende Antwort der *Bürgerkarten-Umgebung* sollte in etwa wie folgt aussehen. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert wurde.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:InfoboxAvailableResponse
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]   <sl:InfoboxIdentifier>Certificates</sl:InfoboxIdentifier>
[05]   <sl:InfoboxIdentifier>IdentityLink</sl:InfoboxIdentifier>
[06]   <sl:InfoboxIdentifier>CompressedIdentityLink</sl:InfoboxIdentifier>
[07]   <sl:InfoboxIdentifier>Mandates</sl:InfoboxIdentifier>
[08] </sl:InfoboxAvailableResponse>
```

Je verfügbarer Infobox enthält die Antwort ein Element `sl:InfoboxIdentifier` mit dem Bezeichner der Infobox (vergleiche Zeile 4 - 7).

#### 2.7.1.3. Downloads zu diesem Beispiel

- examples/interface/infoboxAvailable/First.xml
- examples/interface/infoboxAvailable/First.Response.xml

#### 2.7.1.4. Weiterführende Informationen

- Abschnitt 7.2, „Abfrage der verfügbaren Infoboxen“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 3, „Infoboxen“ in Die österreichische Bürgerkarte - Standardisierte Key- und Infoboxen

### 2.7.2. Infobox anlegen

#### 2.7.2.1. Ein erstes Beispiel

Dieses Beispiel demonstriert den minimal notwendigen Befehl zum Anlegen einer neuen Infobox innerhalb der *Bürgerkarten-Umgebung*. Es wird damit eine neue, vorerst leere Infobox angelegt. Die Zugriffsrechte werden ausschließlich über die *Benutzer-Schnittstelle* festgelegt.

*Anfrage*

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:InfoboxCreateRequest
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]   <sl:InfoboxIdentifier>TutorialBinary</sl:InfoboxIdentifier>
[05]   <sl:InfoboxType>BinaryFile</sl:InfoboxType>
[06]   <sl:Creator>Tutorium zur österreichischen Bürgerkarte</sl:Creator>
[07]   <sl:Purpose>Demonstriert das Anlegen, Lesen, Verändern und Löschen einer\
[08]     Binärdatei.</sl:Purpose>
[09] </sl:InfoboxCreateRequest>
```

Die Anfrage enthält zunächst als Inhalt des Elements `sl:InfoboxIdentifier` in Zeile 4 den frei wählbaren Bezeichner für die neu anzulegende Infobox, in diesem Beispiel eben `TutorialBinary`.

In Zeile 5 wird der Typ der Infobox spezifiziert, mögliche Werte sind `BinaryFile` für eine Abschnitt 7.1.1, „Binärdatei“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer oder `AssocArray` für ein Abschnitt 7.1.2, „Assoziatives Array“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer. In diesem Beispiel soll eine Binärdatei angelegt werden.

`sl:Creator` in Zeile 6 enthält eine Freitextbeschreibung der *Applikation*, welche die Infobox anlegen möchte. Diese Information wird dem *Bürger* von der *Bürgerkarten-Umgebung* über die *Benutzer-Schnittstelle* angezeigt.

Schließlich enthält `sl:Purpose` in Zeile 7 eine Freitextbeschreibung über den Zweck der Infobox. Auch diese Information wird dem *Bürger* von der *Bürgerkarten-Umgebung* über die *Benutzer-Schnittstelle* angezeigt.

#### *Antwort*

Die entsprechende Antwort der *Bürgerkarten-Umgebung* sollte in etwa wie folgt aussehen. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert wurde.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:InfoboxCreateResponse
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"/>
```

Die Antwort besteht aus dem leeren Element `sl:InfoboxCreateResponse`.

#### *Downloads zu diesem Beispiel*

- examples/interface/infoboxCreate/First.xml
- examples/interface/infoboxCreate/First.Response.xml

#### *Weiterführende Informationen*

- Abschnitt 7.3, „Anlegen einer Infobox“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 7.1, „Typen von Infoboxen“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer

### 2.7.2.2. Erweitertes Beispiel

Dieses Beispiel demonstriert das Anlegen einer Infobox vom Typ assoziatives Array innerhalb der *Bürgerkarten-Umgebung*, wobei bereits in der Anfrage Vorschläge bzw. Vorgaben bezüglich der Zugriffsrechte und der Hinweispolitik durch die *Bürgerkarten-Umgebung* gemacht werden.

#### *Anfrage*

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:InfoboxCreateRequest
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]   <sl:InfoboxIdentifier>TutorialAssocArray</sl:InfoboxIdentifier>
[05]   <sl:InfoboxType>AssocArray</sl:InfoboxType>
[06]   <sl:Creator>Tutorium zur österreichischen Bürgerkarte</sl:Creator>
[07]   <sl:Purpose>Demonstriert das Anlegen, Lesen, Verändern und Löschen\
[08]     eines assoziativen Arrays.</sl:Purpose>
```

Die ersten vier Kindelemente wurden bereits im Abschnitt 2.7.2.1, „Ein erstes Beispiel“ zum Anlegen einer Infobox beschrieben.

```
[09]   <sl:ReadAccessAuthorization UserMayChange="true">
[10]     <sl:RequesterID AuthenticationClass="anonym">*</sl:RequesterID>
[11]   </sl:ReadAccessAuthorization>
[12]   <sl:UpdateAccessAuthorization UserMayChange="false">
[13]     <sl:RequesterID
[14]       AuthenticationClass="certified">meine.applikation.at</sl:RequesterID>
[15]   </sl:UpdateAccessAuthorization>
[16]   <sl:ReadUserConfirmation
[17]     UserMayChange="true">confirm</sl:ReadUserConfirmation>
[18]   <sl:UpdateUserConfirmation
[19]     UserMayChange="true">confirmWithSecret</sl:UpdateUserConfirmation>
[20] </sl:InfoboxCreateRequest>
```

Die übrigen vier Kindelemente beinhalten Vorschläge bzw. Vorgaben an die *Bürgerkarten-Umgebung* bezüglich der Zugriffsrechte auf die Infobox sowie Informationspolitik der *Bürgerkarten-Umgebung* gegenüber dem *Bürger* bei Zugriffen auf die Infobox.

`sl:ReadAccessAuthorization` in Zeile 9 - 11 macht einen Vorschlag, welche *Applikationen* auf die Infobox lesend zugreifen dürfen. Es handelt sich dabei deshalb (lediglich) um einen Vorschlag, weil mit dem auf den Wert `true` eingestellten Attribut `UserMayChange` der *Bürgerkarten-Umgebung* mitgeteilt wird, dass der *Bürger* diesen Vorschlag nach eigenem Ermessen abändern darf.

`sl:ReadAccessAuthorization` muss zumindest ein Element `sl:RequestID` enthalten. Jedes dieser Elemente gibt an, welche *Applikation* auf die Infobox zugreifen darf. Der Textinhalt des Elements gibt entweder eine IP-Adresse oder einen Domänennamen an, wobei die Wildcard `*` für eine IP-Adresse für einen oder mehrere Teilbereiche von rechts (z.B. `127.*`) bzw. für einen Domänennamen für einen oder mehrere Teilbereiche von links (`*.ac.at`) eingesetzt werden kann. Der Wert des Attributs `AuthenticationClass` gibt an, in welche Abschnitt 2.1, „Authentisierungsklassen“ in Die österreichische Bürgerkarte - Zugriffsschutz die Authentisierung der *Applikation* durch die *Bürgerkarten-Umgebung* mindestens fallen muss, damit die durch `sl:RequestID` ausgedrückte Regel überhaupt greift. Sind mehrere Elemente `sl:RequestID` angegeben, werden die Regeln nacheinander von der *Bürgerkarten-Umgebung* abgearbeitet. Die erste Regel, welche auf die konkrete Anfrage zutrifft, wird für die Entscheidung, ob der Zugriff erlaubt wird, herangezogen. Im konkreten Fall besagt die Regel, dass beliebige *Applikationen* (`*`) auf die Infobox lesend zugreifen dürfen, wobei keine Authentisierung der *Applikation* durch die *Bürgerkarten-Umgebung* notwendig ist (Klasse `anonym`).

`sl:UpdateAccessAuthorization` in Zeile 12 - 15 macht eine Vorgabe, welche *Applikationen* auf die Infobox verändernd zugreifen dürfen. Es handelt sich dabei deshalb um eine Vorgabe, weil mit dem auf den Wert `false` eingestellten Attribut `UserMayChange` der *Bürgerkarten-Umgebung* mitgeteilt wird, dass der *Bürger* diese Vorgabe nicht abändern darf (Anmerkung: Er kann jedoch sehr wohl das Anlegen der Infobox als Ganzes ablehnen). Analog zu oben enthält `sl:UpdateAccessAuthorization` zumindest ein Element `sl:RequestID`. Im konkreten Fall besagt die einzig vorhandene Regel, dass lediglich die *Applikation* `meine.applikation.at` auf die Infobox verändernd zugreifen darf, wobei eine Authentisierung der *Applikation* durch die *Bürgerkarten-Umgebung* entsprechend der Abschnitt 2.1, „Authentisierungsklassen“ in Die österreichische Bürgerkarte - Zugriffsschutz `certified` notwendig ist.

`sl:ReadUserConfirmation` in Zeile 16 - 17 macht einen Vorschlag, wie die *Bürgerkarten-Umgebung* den *Bürger* von einem nach den für die Infobox festgelegten Regeln grundsätzlich erlaubten Lesezugriff Abschnitt 3.1.3, „Benutzerinteraktion“ in Die österreichische Bürgerkarte - Zugriffsschutz soll. `confirm` bedeutet, dass der *Bürger* über die *Benutzer-Schnittstelle* die Erlaubnis für die Ausführung bestätigen muss.

`sl:UpdateUserConfirmation` in Zeile 18 - 19 macht einen Vorschlag, wie die *Bürgerkarten-Umgebung* den *Bürger* von einem nach den für die Infobox festgelegten Regeln grundsätzlich erlaubten Veränderungszugriff Abschnitt 3.1.3, „Benutzerinteraktion“ in Die österreichische Bürgerkarte - Zugriffsschutz soll. `confirmWithSecret` bedeutet, dass der *Bürger* über die *Benutzer-Schnittstelle* die Erlaubnis für die Ausführung durch Eingabe eines Passwortes bestätigen muss.

#### *Antwort*

Die Antwort besteht auch hier aus dem leeren Element `sl:InfoboxCreateResponse` und wird an dieser Stelle nicht nochmals dargestellt.

#### *Downloads zu diesem Beispiel*

- examples/interface/infoboxCreate/Extended.xml
- examples/interface/infoboxCreate/Extended.Response.xml

#### *Weiterführende Informationen*

- Abschnitt 7.3, „Anlegen einer Infobox“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 2.1, „Authentisierungsklassen“ in Die österreichische Bürgerkarte - Zugriffsschutz
- Abschnitt 3.1.3, „Benutzerinteraktion“ in Die österreichische Bürgerkarte - Zugriffsschutz

### 2.7.3. Infobox löschen

Dieses Beispiel demonstriert das Löschen einer Infobox. Dieser Befehl funktioniert für mit den Abschnitt 3, „Infoboxen“ in Die österreichische Bürgerkarte - Standardisierte Key- und Infoboxen.

#### 2.7.3.1. Anfrage

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:InfoboxDeleteRequest
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]   <sl:InfoboxIdentifier>TutorialBinary</sl:InfoboxIdentifier>
[05] </sl:InfoboxDeleteRequest>
```

Das Kindelement `sl:InfoboxIdentifier` enthält den Bezeichner für die zu löschende Infobox, in diesem Fall also `TutorialBinary`.

#### 2.7.3.2. Antwort

Die entsprechende Antwort der *Bürgerkarten-Umgebung* sollte in etwa wie folgt aussehen. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert wurde.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:InfoboxDeleteResponse
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"/>
```

Die Antwort besteht aus dem leeren Element `sl:InfoboxDeleteResponse`.

#### 2.7.3.3. Downloads zu diesem Beispiel

- examples/interface/infoboxDelete/First.xml
- examples/interface/infoboxDelete/First.Response.xml

#### 2.7.3.4. Weiterführende Informationen

- Abschnitt 7.4, „Löschen einer Infobox“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 3, „Infoboxen“ in Die österreichische Bürgerkarte - Standardisierte Key- und Infoboxen

### 2.7.4. Infobox lesen

#### 2.7.4.1. Lesen einer Binärdatei

Die folgenden Beispiele demonstrieren das Lesen einer Infobox vom Typ Binärdatei. Eine Binärdatei wird in ihrer Gesamtheit ausgelesen.

##### *2.7.4.1.1 Erstes Beispiel*

Dieses Beispiel demonstriert das einfache Lesen einer Infobox vom Typ Binärdatei. Wenn Sie dieses Beispiel nachvollziehen möchten, führen Sie bitte zuerst das Abschnitt 2.7.2.1, „Ein erstes Beispiel“ (Anlegen der Infobox) sowie das Abschnitt 2.7.5.1, „Verändern einer Binärdatei“ (Befüllen der Infobox) aus.

##### *2.7.4.1.2 Lesen der Personenbindung durch die Privatwirtschaft*

Dieses Beispiel demonstriert das Lesen der Infobox *Personenbindung* (`IdentityLink`) durch eine *Applikation* des privaten Bereichs. Entsprechend den Vorgaben aus dem *E-GovG* wurde in der Spezifikation zur österreichischen Bürgerkarte festgelegt, dass ein Auslesen der *Personenbindung* (und damit der *Stammzahl*) nur einer *Applikation* des öffentlichen Bereichs möglich sein darf. Für *Applikationen* des privaten Bereichs ist die *Personenbindung* hingegen nur in modifizierter Form auslesbar, was im Folgenden beschrieben werden soll.

Bitte beachten Sie, dass Sie dieses Beispiel nur ausführen können, wenn die *Bürgerkarten-Umgebung* eine Authentisierung der *Applikation* entsprechend der Abschnitt 2.1, „Authentisierungsklassen“ in Die österreichische Bürgerkarte - Zugriffsschutz *certified* durchführen kann. Ein direktes Absetzen des Requests über das Formular

examples/interface/http-post.html

ist daher nicht möglich.

#### 2.7.4.2. Lesen eines assoziativen Arrays

Die folgenden Beispiele demonstrieren das Lesen einer Infobox vom Typ assoziatives Array. Es stehen Befehle zum Lesen von Schlüsseln des Arrays, zum Lesen eines Wertes zu einem bestimmten Schlüssel, sowie zum Lesen von Schlüssel/Wert-Paaren zur Verfügung.

##### *2.7.4.2.1 Lesen von schlüsseln*

Dieses Beispiel demonstriert das Lesen von Schlüsseln aus der standardisierten Infobox `Certificates`.

##### *2.7.4.2.2 Lesen des Werts zu einem Schlüssel*

Dieses Beispiel demonstriert das Lesen eines Werts zu einem bestimmten Schlüssel aus der standardisierten Infobox `Certificates`.

#### *2.7.4.2.3 Lesen von Schlüssel/Wert-Paaren*

Dieses Beispiel demonstriert das Lesen von Schlüssel/Wert-Paaren aus der standardisierten Infobox `Certificates`.

### 2.7.5. Infobox verändern

#### 2.7.5.1. Verändern einer Binärdatei

Dieses Beispiel demonstriert das Verändern einer Infobox vom Typ Binärdatei. Eine Binärdatei wird verändert, in dem sie als Gesamtheit neu beschrieben wird. Wenn Sie dieses Beispiel nachvollziehen möchten, führen Sie bitte zuerst das Abschnitt 2.7.2.1, „Ein erstes Beispiel“ (Anlegen der Infobox) aus.

#### *Anfrage*

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:InfoboxUpdateRequest
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]   <sl:InfoboxIdentifier>TutorialBinary</sl:InfoboxIdentifier>
[05]   <sl:BinaryFileParameters>
[06]     <sl:XMLContent>
[07]       <my:Customer xmlns:my="urn:my.namespace">
[08]         <my:Name>Tassilo Tester</my:Name>
[09]         <my:LastVisit>2004-12-31</my:LastVisit>
[10]       </my:Customer>
[11]     </sl:XMLContent>
[12]   </sl:BinaryFileParameters>
[13] </sl:InfoboxUpdateRequest>
```

Das Element `sl:InfoboxIdentifier` in Zeile 4 enthält den Bezeichner der zu verändernden Infobox (hier `TutorialBinary`).

`sl:BinaryFileParameters` enthält den neuen Inhalt für die zu verändernde Infobox. In diesem Fall handelt es sich bei den neuen Daten um eine XML-Struktur, die daher als Inhalt von `sl:XMLContent` kodiert wird. Alternativ könnten die Daten auch base64-kodiert als Inhalt von `sl:Base64Content` angegeben werden.

#### *Antwort*

Die entsprechende Antwort der *Bürgerkarten-Umgebung* sollte in etwa wie folgt aussehen.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:InfoboxUpdateResponse
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"/>
```

Die Antwort besteht aus dem inhaltslosen Element `sl:InfoboxUpdateResponse`.

#### *Downloads zu diesem Beispiel*

- examples/interface/infoboxUpdate/First.Binary.xml
- examples/interface/infoboxUpdate/First.Binary.Response.xml

#### *Weiterführende Informationen*

- Abschnitt 7.6, „Verändern von Daten einer Infobox“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 7.1.1, „Binärdatei“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer

#### 2.7.5.2. Verändern eines assoziativen Arrays

Die folgenden Beispiele demonstrieren das Verändern einer Infobox vom Typ assoziatives Array. Es stehen Befehle zum Hinzufügen/Ändern eines Wertes, zum Ändern eines Schlüssels, sowie zum Löschen eines Schlüssels zur Verfügung.

#### *2.7.5.2.1 Hinzufügen/Ändern eines Wertes*

Dieses Beispiel demonstriert das Hinzufügen eines Wertes zu einem assoziativen Array. Wenn Sie dieses Beispiel nachvollziehen möchten, führen Sie bitte zuerst das Abschnitt 2.7.2.2, „Erweitertes Beispiel“ (Anlegen der Infobox) aus.

#### *2.7.5.2.2 Ändern eines Schlüssels*

Dieses Beispiel demonstriert das Ändern des Schlüssels in einem assoziativen Array. Wenn Sie dieses Beispiel nachvollziehen möchten, führen Sie bitte zuerst das Abschnitt 2.7.2.2, „Erweitertes Beispiel“ (Anlegen der Infobox) aus.

#### *2.7.5.2.3 Löschen eines Schlüssel/Wert-Paares*

Dieses Beispiel demonstriert das Löschen eines Schlüssels sowie des dazugehörigen Wertes aus einem assoziativen Array. Wenn Sie dieses Beispiel nachvollziehen möchten, führen Sie bitte zuerst das Abschnitt 2.7.2.2, „Erweitertes Beispiel“ (Anlegen der Infobox) aus.

## 2.8. Abfrage von Eigenschaften

Die folgenden Beispiele demonstrieren die Abfrage von Eigenschaften der *Bürgerkarten-Umgebung* sowie die Abfrage des Status des Bürgerkarten-Tokens.

### 2.8.1. Umgebungs-Eigenschaften

Dieses Beispiel erläutert die Abfrage von Eigenschaften der *Bürgerkarten-Umgebung*. Damit kann die *Applikation* eine Reihe von Parametern auslesen, die für die weitere Kommunikation mit der *Bürgerkarten-Umgebung* bedeutend sein können, wie etwa unterstützte Anzeigeformate oder unterstützte Transformationen im Zusammenhang mit XML-Signaturen.

#### 2.8.1.1. Anfrage

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:GetPropertiesRequest
```

```
[03]    xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"/>
```

Die Anfrage besteht aus dem inhaltslosen Element `sl:GetProperitesRequest`.

#### 2.8.1.2. Antwort

Die entsprechende Antwort der *Bürgerkarten-Umgebung* sollte in etwa wie folgt aussehen. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert wurde.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:GetPropertiesResponse
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]   <sl:ViewerMediaType>text/plain</sl:ViewerMediaType>
[05]   <sl:ViewerMediaType>text/xml</sl:ViewerMediaType>
[06]   <sl:ViewerMediaType>application/xml</sl:ViewerMediaType>
[07]   <sl:ViewerMediaType>text/sgml</sl:ViewerMediaType>
[08]   <sl:ViewerMediaType>application/sgml</sl:ViewerMediaType>
[09]   <sl:ViewerMediaType>text/tab-separated-values</sl:ViewerMediaType>
[10]   <sl:ViewerMediaType>message/rfc822</sl:ViewerMediaType>
[11]   <sl:ViewerMediaType>text/html</sl:ViewerMediaType>
[12]   <sl:ViewerMediaType>application/xhtml+xml</sl:ViewerMediaType>
```

Zunächst enthält die Antwort je unterstütztem Anzeigeformat ein Element `sl:ViewerMediaType`. Dieses Element enthält den *Mime Type* für das unterstützte Anzeigeformat. Die Elemente in den Zeilen 4 - 12 entsprechen der Abschnitt 9, „Anzeigeformate und Zeichensätze“ in Die österreichische Bürgerkarte - Minimale Umsetzung des Security-Layers, die von einer *Bürgerkarten-Umgebung* unterstützt werden müssen.

```
[13]   <sl:XMLSignatureTransform>
[14]     http://www.w3.org/TR/2001/REC-xml-c14n-20010315</sl:XMLSignatureTransform>
[15]   <sl:XMLSignatureTransform>
[16]     http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments</sl:XMLSignatureTransform>
[17]   <sl:XMLSignatureTransform>
[18]     http://www.w3.org/2001/10/xml-exc-c14n#</sl:XMLSignatureTransform>
[19]   <sl:XMLSignatureTransform>
[20]     http://www.w3.org/2001/10/xml-exc-c14n#WithComments</sl:XMLSignatureTransform>
[21]   <sl:XMLSignatureTransform>
[22]     http://www.w3.org/2000/09/xmldsig#base64</sl:XMLSignatureTransform>
[23]   <sl:XMLSignatureTransform>
[24]     http://www.w3.org/TR/1999/REC-xpath-19991116</sl:XMLSignatureTransform>
[25]   <sl:XMLSignatureTransform>
[26]     http://www.w3.org/2002/06/xmldsig-filter2</sl:XMLSignatureTransform>
[27]   <sl:XMLSignatureTransform>
[28]     http://www.w3.org/2000/09/xmldsig#enveloped-signature</sl:XMLSignatureTransform>
[29]   <sl:XMLSignatureTransform>
[30]     http://www.w3.org/TR/1999/REC-xslt-19991116</sl:XMLSignatureTransform>
```

Anschließend enthält die Antwort je unterstütztem Transformationsalgorithmus für XML-Signaturen ein Element `sl:XMLSignatureTransform`. Dieses Element enthält die URI für den unterstützten Algorithmus. Die Elemente in den Zeilen 13 - 30 entsprechen der Abschnitt 5.1.4, „Transformationsalgorithmen“ in Die österreichische Bürgerkarte - Minimale Umsetzung des Security-Layers, die von einer *Bürgerkarten-Umgebung* unterstützt werden müssen.

```
[31]   <sl:KeyboxIdentifier Signature="1" Encryption="0">
[32]     SecureSignatureKeypair</sl:KeyboxIdentifier>
[33]   <sl:KeyboxIdentifier Signature="1" Encryption="1">
[34]     CertifiedKeypair</sl:KeyboxIdentifier>
```

Es folgen mit den Elementen `sl:KeyBoxIdentifier` die Bezeichner aller in der *Bürgerkarten-Umgebung* vorhandenen Schlüsselpaare. Für jedes Schlüsselpaar wird angegeben, ob es zur Verwendung im Kontext Signatur (Attribut `Signature`) bzw. Verschlüsselung (Attribut) geeignet ist. Die Elemente in den Zeilen 31 - 34 entsprechen den Abschnitt 2, „Keyboxen“ in Die österreichische Bürgerkarte - Standardisierte Key- und Infoboxen der Spezifikation *Standardisierte Key- und Infoboxen*.

```
[35]   <sl:Binding Identifier="TCP/IP"/>
[36]   <sl:Binding Identifier="TLS"/>
[37]   <sl:Binding Identifier="HTTP"/>
[38]   <sl:Binding Identifier="HTTPS"/>
```

Danach enthält die Antwort mit den Elementen `sl:Binding` die unterstützten Transportprotokolle für das XML-Protokoll des Security-Layers. Der Wert von `sl:Binding/@Identifier` enthält jeweils den Namen des unterstützten Transportprotokolls. Die Elemente in den Zeilen 35 - 38 entsprechen allen in der Spezifikation Transportprotokolle Security-Layer beschriebenen Mechanismen.

```
[39]   <sl:ProtocolVersion>1.0</sl:ProtocolVersion>
[40]   <sl:ProtocolVersion>1.1</sl:ProtocolVersion>
[41]   <sl:ProtocolVersion>1.2</sl:ProtocolVersion>
[42] </sl:GetPropertiesResponse>
```

Abschließend geben die Elemente `sl:ProtocolVersion` die Versionen des XML-Protokolls des Security-Layers an, die von der *Bürgerkarten-Umgebung* unterstützt werden. Die Elemente in den Zeilen 39 - 41 bedeuten, dass die *Bürgerkarten-Umgebung* alle bisher erschienenen Protokollversionen des Security-Layers unterstützt.

##### *Downloads zu diesem Beispiel*

- examples/interface/getProperties/First.xml
- examples/interface/getProperties/First.Response.xml

##### *Weiterführende Informationen*

- Abschnitt 8.1, „Abfrage der Umgebungseigenschaften“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 9, „Anzeigeformate und Zeichensätze“ in Die österreichische Bürgerkarte - Minimale Umsetzung des Security-Layers
- Abschnitt 5.1.4, „Transformationsalgorithmen“ in Die österreichische Bürgerkarte - Minimale Umsetzung des Security-Layers
- Abschnitt 2, „Keyboxen“ in Die österreichische Bürgerkarte - Standardisierte Key- und Infoboxen
- Transportprotokolle Security-Layer
- Abschnitt 3, „Transportprotokolle für den Security-Layer“ in Die österreichische Bürgerkarte - Minimale Umsetzung des Security-Layers

### 2.8.2. Token-Status

Dieses Beispiel erläutert die Abfrage des Status des Bürgerkarten-Tokens. Es wird hier der Vollständigkeit halber angeführt. In der Praxis wird dieser (historische) Befehl wohl kaum mehr zum Einsatz kommen.

#### 2.8.2.1. Anfrage

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:GetStatusRequest
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"/>
```

Die Anfrage besteht hier lediglich aus dem leeren Element `sl:GetStatusRequest`. Für weitere Möglichkeiten siehe die Abschnitt 8.2, „Abfrage des Tokenstatus“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer.

#### 2.8.2.2. Antwort

Die entsprechende Antwort der *Bürgerkarten-Umgebung* sollte in etwa wie folgt aussehen. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert wurde.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:GetStatusResponse
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]   <sl:TokenStatus>ready</sl:TokenStatus>
[05] </sl:GetStatusResponse>
```

Die Antwort enthält ein einziges Element `sl:TokenStatus`, dessen Textinhalt anzeigt, ob der Bürgerkarten-Token bereit ist (Wert `ready`) oder nicht (Wert `removed`). Im Fall einer lokalen *Bürgerkarten-Umgebung* kann damit erkannt werden, ob die Bürgerkarte im Kartenlesegerät steckt oder nicht. Im Fall einer serverbasierten *Bürgerkarten-Umgebung* wird wohl immer der Wert `ready` zurückgeliefert werden.

*Downloads zu diesem Beispiel*

- examples/interface/getStatus/First.xml
- examples/interface/getStatus/First.Response.xml

*Weiterführende Informationen*

- Abschnitt 8.2, „Abfrage des Tokenstatus“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer

## 2.9. Null-Operation

Dieses Beispiel demonstriert die Verwendung der Null-Operation. Wie der Name schon sagt, führt die *Bürgerkarten-Umgebung* bei Aufruf dieses Befehls keine Funktionen aus, sondern liefert lediglich eine statische Antwort zurück. Für eine Motivation dieses Befehls siehe seine Abschnitt 9, „Null-Operation“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer.

### 2.9.1. Anfrage

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:NullOperationRequest
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"/>
```

Die Anfrage besteht aus dem leeren Element `sl:NullOperationRequest`.

### 2.9.2. Antwort

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:NullOperationResponse
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"/>
```

Die Antwort besteht aus dem leeren Element `sl:NullOperationResponse`.

### 2.9.3. Downloads zu diesem Beispiel

- examples/interface/nullOperation/First.xml
- examples/interface/nullOperation/First.Response.xml

### 2.9.4. Weiterführende Informationen

- Abschnitt 9, „Null-Operation“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer

## 2.10. Fehlerbehandlung

Konnte eine Befehlsanfrage von der *Bürgerkarten-Umgebung* nicht korrekt abgearbeitet werden, antwortet sie mit einer Fehler-Antwort anstatt der korrespondierenden Befehlsantwort. Im Folgenden wird eine solche Fehler-Antwort beispielhaft aufgeführt.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:ErrorResponse
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]   <sl:ErrorCode>4901</sl:ErrorCode>
[05]   <sl:Info>Token nicht bereit (bzw. Token nicht vorhanden)</sl:Info>
[06] </sl:ErrorResponse>
```

Die Fehler-Antwort besteht immer aus den zwei Elementen `sl:ErrorCode` und `sl:Info`.

`sl:ErrorCode` enthält als Textinhalt einen vierstelligen Fehlercode entsprechend der Spezifikation Fehlercodes Security-Layer.

`sl:Info` enthält als Textinhalt eine Freitextbeschreibung der Fehlerursache. Diese dient der einfachen Interpretation durch einen Menschen.

### 2.10.1. Downloads zu diesem Beispiel

- examples/interface/error/Error.xml

### 2.10.2. Weiterführende Informationen

- Abschnitt 10, „Fehlerbehandlung“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Fehlercodes Security-Layer

# 3. Transportprotokolle

Dieser Abschnitt behandelt das Senden von Befehlen an die *Bürgerkarten-Umgebung* über die spezifizierten Transportprotokolle TCP/IP bzw. TLS sowie HTTP bzw. HTTPS. Der Schwerpunkt liegt dabei auf den letzten beiden Protokollen.

## 3.1. TCP/IP bzw. TLS

Die in Abschnitt 2 besprochenen XML-Schnittstellenbefehle werden über TCP/IP bzw. TLS direkt und ohne jegliche weitere Kodierung oder Umsetzung/Einbettung übertragen.

Die *Applikation* öffnet also eine TCP/IP- bzw. TLS-Verbindung zur *Bürgerkarten-Umgebung* und sendet die entsprechende XML-Befehlsanfrage. Die *Bürgerkarten-Umgebung* antwortet mit der korrespondierenden XML-Befehlsantwort und schließt danach die Verbindung.

Sie können das Senden von XML-Schnittstellenbefehlen über TCP/IP an eine lokale *Bürgerkarten-Umgebung* einfach ausprobieren, indem Sie mit einem Telnet-Client eine Verbindung zur Adresse `localhost:3495` herstellen und einen der in Abschnitt 2 besprochenen XML-Befehlsanfragen senden.

### 3.1.1. Weiterführende Informationen

- Abschnitt 2, „TCP/IP-Bindung“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer
- Abschnitt 4, „SSL/TLS-Bindung“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer

## 3.2. HTTP bzw. HTTPS

Die Übertragung einer in Abschnitt 2 besprochenen XML-Befehlsanfrage über HTTP bzw. HTTPS erfolgt durch Einbettung in einen HTTP Request. Genauer gesagt wird die XML-Befehlsanfrage zusammen mit eventuell vorhandenen weiteren Parametern (siehe unten) wahlweise in einen HTTP GET oder HTTP POST Request eingebettet. Die Kodierung der Parameter erfolgt dabei wahlweise als `application/x-www-form-urlencoded` oder als `multipart/form-data`. Somit ist es möglich, die XML-Befehlsanfrage mittels HTML-Formular direkt aus dem Browser des *Bürgers* heraus an die *Bürgerkarten-Umgebung* abzusenden.

Die Antwort der *Bürgerkarten-Umgebung* ist abhängig von den eventuell zusammen mit der eigentlichen XML-Befehlsanfrage im HTTP Request übermittelten weiteren *Formular-Parameter*. Details dazu erfahren Sie in den folgenden Unterabschnitten.

### 3.2.1. Resultat zurück an den Browser

Im einfachsten Fall sendet die *Bürgerkarten-Umgebung* die XML-Befehlsantwort auf eine per HTTP Request vom Browser des *Bürgers* erhaltene XML-Befehlsanfrage direkt in der korrespondierenden HTTP Response an den Browser zurück.

#### 3.2.1.1. Resultat als XML

Wird im HTTP Request lediglich die XML-Befehlsanfrage als *Formular-Parameter* `XMLRequest` angegeben, darüber hinaus aber keine weiteren *Formular-Parameter*, sendet die *Bürgerkarten-Umgebung* die XML-Befehlsantwort direkt als Nutzlast der korrespondierenden HTTP Response an den Browser zurück.

Das folgende Beispiel demonstriert diesen Fall. Gesendet wird der Befehl *NullOperation*.

##### *HTML-Seite mit Formular*

Nachfolgend sehen Sie die HTML-Seite mit dem Formular, das an die *Bürgerkarten-Umgebung* gesendet werden soll. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert und umgebrochen (gekennzeichnet durch das Zeichen \ am Ende einer Zeile) wurde.

```
[01] <html>
[02]   <head>
[03]     <title>Direkte Ansteuerung</title>
[04]   </head>
[05]   <body>
[06]     <p>Direktes Senden des Befehls <code>NullOperation</code>.</p>
[07]     <form method="post" action="http://127.0.0.1:3495/http-security-layer-request">
[08]       <input name="XMLRequest" type="hidden"
[09]       value="&lt;?xml version='1.0' encoding='UTF-8'?>&lt;NullOperationRequest \
[10] xmlns='http://www.buergerkarte.at/namespaces/securitylayer/1.2#'/>"/>
[11]       <input type="submit" value="Request absenden"/>
[12]     </form>
[13]   </body>
[14] </html>
```

Man erkennt in den Zeilen 7 - 12 das HTML-Formular, dessen Formulardaten mittels HTTP POST (vergleiche Attribut `method` in Zeile 7) an die *Bürgerkarten-Umgebung* gesendet werden soll (vergleiche Attribut `action` in Zeile 7; als Pfadkomponente in der URL muss immer `http-security-layer-request` in genau dieser Schreibweise verwendet werden).

Die Zeilen 8 - 10 enthalten die Angaben zum XML-Schnittstellenbefehl. Der Name des Formularelements muss stets `XMLRequest` lauten. Das Attribut `value` enthält den XML-Schnittstellenbefehl; beachten Sie bitte das innerhalb des Attribut-Werts notwendige Escaping z.B. für das Zeichen < (`<`). Nachdem der XML-Schnittstellenbefehl für den Bürger nicht sichtbar sein soll, wurde das Formularelement als versteckt definiert (vergleiche Attribut `type` in Zeile 8).

##### *HTTP Request*

Nachfolgend sehen Sie den HTTP Request, der aus dem Absenden des Formulars in obiger HTML-Seite resultiert. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen \ am Ende einer Zeile) wurde.

```
[01] POST /http-security-layer-request HTTP/1.1
[02] Host: localhost
[03] User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.5) Gecko/20041107 Firefox/1.0
[04] Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,\
[05] image/png,*/*;q=0.5
[06] Accept-Language: de-at,de;q=0.7,en;q=0.3
[07] Accept-Encoding: gzip,deflate
[08] Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
[09] Keep-Alive: 300
[10] Connection: keep-alive
```

```
[11] Content-Type: application/x-www-form-urlencoded
[12] Content-Length: 183
[13]
[14] XMLRequest=%3C%3Fxml+version%3D%271.0%27+encoding%3D%27UTF-8%27%3F%3E%3CNullOperationRequest\
[15] +xmlns%3D%27http%3A%2F%2Fwww.buergerkarte.at%2Fnamespaces%2Fsecuritylayer%2F1.2%23%27%2F%3E
```

Man erkennt in Zeile 11, dass die Formulardaten als `application/x-www-form-urlencoded` gesendet wurden. Der Formularparameter `XMLRequest` ist entsprechend kodiert in den Zeilen 14 - 15 erkennbar.

#### *HTTP Response*

Im Folgenden sehen Sie die HTTP Response, welche die *Bürgerkarten-Umgebung* an den Browser des *Bürgers* zurücksendet. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen `\`) wurde.

```
[00] HTTP/1.1 200 OK
[01] Server: citizen-card-environment/1.2 trustDeskbasic/2.2.3-developer
[02] Content-Type: text/xml; charset=UTF-8
[03] Content-Length: 133
[04]
[05] <?xml version="1.0" encoding="UTF-8"?><sl:NullOperationResponse \
[06] xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"/>
```

In Zeile 1 Abschnitt 3.3.2.1, „HTTP-Response an die Browser-Verbindung“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer sich die *Bürgerkarten-Umgebung* mittels des *Headers* `Server`.

In Zeile 5-6, also als Nutzlast der HTTP Response erkennt man die XML-Befehlsantwort der *Bürgerkarten-Umgebung*. Deshalb enthält auch der *Header* `Content-Type` in Zeile 2 den passenden *Mime Type* `text/xml`.

#### *Downloads zu diesem Beispiel*

- examples/bindings/direct/Request.html
- examples/bindings/direct/Browser.HttpRequest.txt
- examples/bindings/direct/Browser.HttpResponse.txt

#### *Weiterführende Informationen*

- Abschnitt 3, „HTTP-Bindung“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer
- Abschnitt 3.3.1, „HTTP-Request“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer
- Abschnitt 3.3.2.1, „HTTP-Response an die Browser-Verbindung“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer

### 3.2.1.2. Transformiertes Resultat als HTML

Wird im HTTP Request neben der XML-Befehlsanfrage als *Formular-Parameter* `XMLRequest` auch ein weiterer *Formular-Parameter* `StylesheetURL` angegeben, transformiert die *Bürgerkarten-Umgebung* die XML-Befehlsantwort zunächst unter Zuhilfename des in `StylesheetURL` referenzierten Stylesheets (z.B.) in ein HTML-Dokument, um dieses dann als Nutzlast der korrespondierenden HTTP Response an den Browser zurückzusenden.

Das folgende Beispiel demonstriert diesen Fall. Gesendet wird der Befehl *GetPropertiesRequest*.

#### *HTML-Seite mit Formular*

Nachfolgend sehen Sie die HTML-Seite mit dem Formular, das an die *Bürgerkarten-Umgebung* gesendet werden soll. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert und umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) wurde.

```
[01] <html>
[02]   <head>
[03]     <title>Direkte Ansteuerung mit Stylesheet-Transformation</title>
[04]   </head>
[05]   <body>
[06]     <p>Direktes Senden des Befehls <code>GetProperties</code> mit Stylesheet-\
[07] Transformation der Befehlsantwort.</p>
[08]     <form method="post" action="http://127.0.0.1:3495/http-security-layer-request">
[09]       <input name="XMLRequest" type="hidden"
[10]         value="&lt;?xml version='1.0' encoding='UTF-8'?>&lt;GetPropertiesRequest \
[11] xmlns='http://www.buergerkarte.at/namespaces/securitylayer/1.2#'/>"/>
[12]       <input name="StylesheetURL" type="hidden"
[13]         value="http://www.buergerkarte.at/konzept/securitylayer/spezifikation/20040514\
[14] /tutorial/examples/bindings/stylesheet/Stylesheet.xslt"/>
[15]       <input type="submit" value="Request absenden"/>
[16]     </form>
[17]   </body>
[18] </html>
```

Im Unterschied zu Abschnitt 3.2.1.1, „Resultat als XML“ ist hier in den Zeilen 12 - 14 zusätzlich der *Formular-Parameter* `StylesheetURL` angegeben. Der Wert des Attributs `value` enthält eine URL auf den Stylesheet, der von der *Bürgerkarten-Umgebung* für die Transformation der Befehlsantwort herangezogen werden soll.

#### *HTTP Request*

Nachfolgend sehen Sie den HTTP Request, der aus dem Absenden des Formulars in obiger HTML-Seite resultiert. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) wurde.

```
[01] POST /http-security-layer-request HTTP/1.1
[02] Host: localhost
[03] User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.5) Gecko/20041107 \
[04] Firefox/1.0
[05] Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;\
[06] q=0.8,image/png,*/*;q=0.5
[07] Accept-Language: de-at,de;q=0.7,en;q=0.3
[08] Accept-Encoding: gzip,deflate
[09] Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
[10] Keep-Alive: 300
[11] Connection: keep-alive
[12] Content-Type: application/x-www-form-urlencoded
[13] Content-Length: 355
```

```
[14]
[15] XMLRequest=%3C%3Fxml+version%3D%271.0%27+encoding%3D%27UTF-8%27%3F%3E%3CGetProperties\
[16] Request+xmlns%3D%27http%3A%2F%2Fwww.buergerkarte.at%2Fnamespaces%2Fsecuritylayer\
[17] %2F1.2%23%27%2F%3E&StylesheetURL=http%3A%2F%2Fwww.buergerkarte.at%2Fkonzept%2F\
[18] securitylayer%2Fspezifikation%2F20040514%2Ftutorial%2Fexamples%2Fbindings%2F
[19] stylesheet%2FStylesheet.xslt
```

Man erkennt in Zeile 21, dass die Formulardaten als `application/x-www-form-urlencoded` gesendet wurden. Die Formularparameter `XMLRequest` und `StylesheetURL` sind entsprechend kodiert in den Zeilen 15 - 19 erkennbar.

#### *HTTP Response*

Im Folgenden sehen Sie die HTTP Response, welche die *Bürgerkarten-Umgebung* an den Browser des *Bürgers* zurücksendet. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit gekürzt und umgebrochen (gekennzeichnet durch das Zeichen \) wurde.

```
[01] HTTP/1.1 200 OK
[02] Server: citizen-card-environment/1.2 trustDeskbasic/2.2.3-developer
[03] Content-Type: text/html; charset=UTF-8
[04] Content-Length: 2441
[05]
[06] <!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" \
[07] "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
[08] <html xmlns="http://www.w3.org/1999/xhtml" \
[09] xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[10] <head>
[11] <title>Stylesheet: Resultat</title>
[12] </head>
[13] <body><table xmlns="" border="1" bgcolor="silver">
[14] <tbody>
[15] <tr>
[16] <td>Anzeigeformat</td><td><code>text/plain</code></td></tr>
[17] <tr>
[18] ...
[19] <tr>
[20] <td>XMLDSIG-Transformation</td><td><code>http://www.w3.org/TR/2001/\
[21] REC-xml-c14n-20010315</code></td>
[22] </tr>
[23] ...
[24] <tr>
[25] <td>Schlüssel (Signatur)</td><td><code>SecureSignatureKeypair</code></td>
[26] </tr>
[27] ...
[28] <tr>
[29] <td>Transport-Protokoll</td><td><code>TCP/IP</code></td>
[30] </tr>
[31] ...
[32] <tr>
[33] <td>Protokoll-Version</td><td><code>1.0</code></td>
[34] </tr>
[35] ...
[36] </tbody>
[37] </table>
[38] </body>
[39] </html>
```

Im Gegensatz zum Abschnitt 3.2.1.1, „Resultat als XML“ enthält die Nutzlast (vgl. Zeile 6 - 39) hier nicht die XML-Befehlsantwort, sondern das Ergebnis der ihrer Transformation mit dem in `StylesheetURL` referenzierten Stylesheet. Die Transformation hat ein HTML-Dokument erzeugt, deshalb enthält auch der *Header* `Content-Type` in Zeile 2 den passenden *Mime Type* `text/html`.

#### *Downloads zu diesem Beispiel*

- examples/bindings/stylesheet/Request.html
- examples/bindings/stylesheet/Browser.HttpRequest.txt
- examples/bindings/stylesheet/Browser.HttpResponse.txt
- examples/bindings/stylesheet/Stylesheet.xslt

#### *Weiterführende Informationen*

- Abschnitt 3, „HTTP-Bindung“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer
- Abschnitt 3.2.3, „Schrittweiser Ablauf“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer

### 3.2.2. Resultat an den Applikations-Server

Die *Bürgerkarten-Umgebung* kann durch Angabe des *Formular-Parameters* `DataURL` im HTTP Request dazu veranlasst werden, die XML-Befehlsantwort nicht in der HTTP Response an den Browser zurückzusenden, sondern an den in der `DataURL` referenzierten *Applikations*-Server.

Welche Daten letztendlich von der *Bürgerkarten-Umgebung* in der HTTP Response an den Browser zurückübermittelt werden, hängt sowohl von der Reaktion des *Applikations*-Servers sowie von eventuell weiteren, im HTTP Request des Browsers angegebenen *Formular-Parametern* ab.

Die nachfolgenden Beispiele demonstrieren einige Kombinationsmöglichkeiten.

> **Anmerkung**
>
> Sie können die nachfolgenden Beispiele auch selbst ausprobieren. Die dazu notwendige *Applikation* finden Sie in Form einer J2EE Web Application als Web-Archiv. Bringen Sie die Web Application so zum Einsatz, dass ihr Root Context unter der URL `http://localhost:8080/SL12Tutorial/` bzw. per HTTPS unter der URL http://localhost:8443/SL12Tutorial/ erreichbar ist. Beachten Sie dabei bitte, dass die Web Application lediglich als Prototyp zu verstehen und nicht für einen Echteinsatz gedacht ist (mangelnde Robustheit und Modularität).

#### 3.2.2.1. Asynchrone Benutzerführung mittels RedirectURL und DataURL

Im folgenden Beispiel werden im HTTP Request neben der XML-Befehlsanfrage als *Formular-Parameter* `XMLRequest` zwei weitere *Formular-Parameter* angegeben:

- Der *Formular-Parameter* `DataURL` enthält als Wert eine URL des *Applikations*-Servers, an den die *Bürgerkarten-Umgebung* die XML-Befehlsantwort übermitteln soll, anstatt sie wie bisher in der HTTP Response an den Browser zurückzusenden.
- Der *Formular-Parameter* `RedirectURL` enthält als Wert eine weitere URL des *Applikations*-Servers, an welche der Browser von der *Bürgerkarten-Umgebung* als Reaktion auf den empfangenen HTTP Request umgeleitet werden soll.

Durch die Kombination dieser Formularparameter wird im Beispiel der folgende Ablauf realisiert:

1. Der *Bürger* sendet einen HTTP Request mit dem `XMLRequest` (im konkreten Fall mit dem Befehl *GetProperties*) und den beiden weiteren *Formular-Parametern* `DataURL` sowie `RedirectURL` an die *Bürgerkarten-Umgebung*.
2. Die *Bürgerkarten-Umgebung* antwortet auf den HTTP Request des *Bürgers* sofort mit einem HTTP *Redirect* (Code `302` oder `303`) auf die in `RedirectURL` angegebene Lokation. Diese Lokation ist Teil der *Applikation* und fungiert als Warteschleife, in welcher der *Bürger* so lange gehalten wird, bis die Antwort der *Bürgerkarten-Umgebung* bei der Applikation eintrifft (siehe Schritt 3).
3. Die *Bürgerkarten-Umgebung* bearbeitet die XML-Befehlsanfrage und übermittelt die XML-Befehlsantwort in einem HTTP *POST* Request an die in der `DataURL` angegebene Lokation, einem Teil der *Applikation*.
4. Die *Applikation* bestätigt der *Bürgerkarten-Umgebung* in der HTTP Response den korrekten Empfang der XML-Befehlsantwort.
5. Die *Applikation* verarbeitet die XML-Befehlsantwort und antwortet dem *Bürger* anstatt mit der Warteseite mit dem nächsten Schritt im Verfahren.

### HTML-Formular

Das folgende HTML-Formular wurde von der Applikation dynamisch generiert und kann unter der Voraussetzung der installierten ??? von der URL `http://localhost:8080/SL12Tutorial/Redirect` geladen werden. Bitte beachten Sie, dass es aus Gründen der besseren Lesbarkeit formatiert und umgebrochen (gekennzeichnet durch das Zeichen \ am Ende einer Zeile) wurde.

```
[01] <html>
[02]   <head>
[03]     <title>Redirect: Start</title>
[04]   </head>
[05]   <body>
[06]     <form method="post" action="http://127.0.0.1:3495/http-security-layer-request">
[07]       <input name="XMLRequest" type="hidden"
[08]         value="<?xml version='1.0' encoding='UTF-8'?><GetPropertiesRequest \
[09] xmlns='http://www.buergerkarte.at/namespaces/securitylayer/1.2#'/>" />
[10]       <input name="DataURL" type="hidden"
[11]         value="http://localhost:8080/SL12Tutorial/Redirect;jsessionid=\
[12] 546C378088AAE921AE9BEE488582580E?use=dataurl" />
[13]       <input name="RedirectURL" type="hidden"
[14]         value="http://localhost:8080/SL12Tutorial/Redirect;jsessionid=\
[15] 546C378088AAE921AE9BEE488582580E?use=redirecturl" />
[16]       <input name="Request absenden" type="submit" />
[17]     </form>
[18]   </body>
[19] </html>
```

In Zeile 10 - 12 ist der *Formular-Parameter* `DataURL`, in Zeile 13 - 15 der *Formular-Parameter* `RedirectURL` kodiert.

Man erkennt, dass beide URLs die *Session ID* enthalten, die für das spätere Zusammenführen des Browsers des *Bürgers*, der in der Warteschleife hängen wird, mit der XML-Befehlsantwort, die von der *Bürgerkarten-Umgebung* an die *Applikation* übermittelt werden wird, notwendig ist. Die *Session ID* muss in den URLs mitgeführt werden, weil ein Mitführen mittels *Cookies* nicht möglich ist, da es die *Applikation* ja mit zwei unterschiedlichen Clients (Browser des *Bürgers*, *Bürgerkarten-Umgebung*) zu tun hat.

### HTTP Request des Browsers an die Bürgerkarten-Umgebung

Nachfolgend sehen Sie den HTTP Request, der aus dem Absenden des Formulars in obiger HTML-Seite resultiert. Bitte beachten Sie, dass er aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen \ am Ende einer Zeile) wurde.

```
[01] POST /http-security-layer-request HTTP/1.1
[02] Host: 127.0.0.1
[03] User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.5) Gecko/20041107 \
[04] Firefox/1.0
[05] Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;\
[06] q=0.8,image/png,*/*;q=0.5
[07] Accept-Language: de-at,de;q=0.7,en;q=0.3
[08] Accept-Encoding: gzip,deflate
[09] Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
[10] Keep-Alive: 300
[11] Connection: keep-alive
[12] Content-Type: application/x-www-form-urlencoded
[13] Content-Length: 467
[14]
[15] XMLRequest=%3C%3Fxml+version%3D%271.0%27+encoding%3D%27UTF-8%27%3F%3E%3CGetPropertiesRequest\
[16] +xmlns%3D%27http%3A%2F%2Fwww.buergerkarte.at%2Fnamespaces%2Fsecuritylayer%2F1.2%23%27%2F%3E&\
[17] DataURL=http%3A%2F%2Flocalhost%3A8080%2FSL12Tutorial%2FRedirect%3Bjsessionid%3D546C378088AAE\
[18] 921AE9BEE488582580E%3Fuse%3Ddataurl&RedirectURL=http%3A%2F%2Flocalhost%3A8080%2FSL12Tutorial\
[19] %2FRedirect%3Bjsessionid%3D546C378088AAE921AE9BEE488582580E%3Fuse%3Dredirecturl&Request+absenden
```

### HTTP Response der Bürgerkarten-Umgebung an den Browser

Im Folgenden sehen Sie die HTTP Response der *Bürgerkarten-Umgebung* an den Browser. Bitte beachten Sie, dass sie aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen \ am Ende einer Zeile) wurde.

```
[01] HTTP/1.1 302 Object Moved
[02] Server: citizen-card-environment/1.2 trustDeskbasic/2.2.3-developer
[03] Location: http://localhost:8080/SL12Tutorial/Redirect;jsessionid=\
[04] 546C378088AAE921AE9BEE488582580E?use=redirecturl
```

Es wurde also ein HTTP *Redirect* (vergleiche Code `302` in Zeile 1) gesendet. Der *Header* `Location` enthält den Wert aus dem *Formular-Parameter* `RedirectURL`.

### HTTP Request/Response an/von RedirectURL

Es folgen der HTTP Request des Browsers an die `RedirectURL` sowie die HTTP Response der dahinter stehenden *Applikation*. Bitte beachten Sie, dass sie aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen \ am Ende einer Zeile) wurden.

```
[01] GET /SL12Tutorial/Redirect;jsessionid=546C378088AAE921AE9BEE488582580E?\
[02] use=redirecturl HTTP/1.1
[03] Host: 127.0.0.1
[04] User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.5) Gecko/20041107 \
[05] Firefox/1.0
[06] Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;\
[07] q=0.8,image/png,*/*;q=0.5
[08] Accept-Language: de-at,de;q=0.7,en;q=0.3
[09] Accept-Encoding: gzip,deflate
[10] Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
[11] Keep-Alive: 300
[12] Connection: keep-alive
```

```
[01] HTTP/1.1 200 OK
[02] Content-Type: text/html
[03] Content-Length: 340
[04] Date: Mon, 27 Dec 2004 20:17:21 GMT
[05] Server: Apache Coyote/1.0
[06]
[07] <html>
[08] <head>
[09] <title>Redirect: Warteschleife</title>
[10] <meta http-equiv="refresh" content="10; URL=http://localhost:8080/SL12Tutorial/\
[11] Redirect;jsessionid=546C378088AAE921AE9BEE488582580E?use=redirecturl"/>
[12] </head>
[13] <body>
[14] <p>Bitte warten. Sobald die Antwort von der BKU eingetroffen ist, wird sie \
[15] hier angezeigt.</p>
[16] </body>
[17] </html>
```

Die Antwort-Seite der Applikation ist als Warteschleife konzipiert. In Zeile 10 enthält die HTML-Seite ein Meta-Tag, mit dem der Browser veranlasst wird, die Seite alle 10 Sekunden neu zu laden.

### *HTTP Request der Bürgerkarten-Umgebung an DataURL*

Als Nächstes sehen Sie den HTTP Request der *Bürgerkarten-Umgebung* an die hinter der `DataURL` stehende *Applikation*. Dieser enthält die u. a. die XML-Befehlsantwort. Bitte beachten Sie, dass er aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) und gekürzt (gekennzeichnet durch `...`) wurde.

```
[01] POST /SL12Tutorial/Redirect;jsessionid=546C378088AAE921AE9BEE488582580E?\
[02] use=dataurl HTTP/1.1
[03] Referer: localhost
[04] User-Agent: citizen-card-environment/1.2 trustDeskbasic/2.2.3-developer
[05] Content-Type: application/x-www-form-urlencoded
[06] Host: 127.0.0.1
[07] Content-Length: 2275
[08] Connection: Keep-Alive
[09] Cache-Control: no-cache
[10]
[11] XMLResponse=%3C%3Fxml+version%3D%221.0%22+encoding%3D%22UTF-8%22%3F%3E%0D%0A%3C\
[12] GetPropertiesResponse+xmlns%3D%22http%3A%2F%2Fwww.buergerkarte.at%2Fnamespaces%\
[13] ...%3C%2FGetPropertiesResponse%3E&ResponseType=HTTP-Security-Layer-RESPONSE
```

Es handelt sich bei diesem Request stets um HTTP POST, wie auch in Zeile 1 zu sehen ist.

Zeile 4 enthält die obligatorische Abschnitt 3.3.2.2, „HTTP-Request an DataURL“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer der *Bürgerkarten-Umgebung* als *User Agent*.

Die Nutzlast enthält wiederum eine Reihe von *Formular-Parametern*, die (so wie hier) als `application/x-www-form-urlencoded` oder als `multipart/form-data` Abschnitt 3.3.2.2, „HTTP-Request an DataURL“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer sein können. Jedenfalls enthalten sein müssen (so wie hier) die Parameter `XMLResponse` mit der XML-Befehlsantwort sowie der Parameter `ResponseType` mit dem fixen Wert `HTTP-Security-Layer-RESPONSE`.

### *HTTP Response von DataURL an die Bürgerkarten-Umgebung*

Im Folgenden finden Sie die HTTP Response der hinter der `DataURL` stehenden *Applikation* an die *Bürgerkarten-Umgebung*. Bitte beachten Sie, dass sie aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) wurde.

```
[01] HTTP/1.1 200 OK
[02] Content-Type: text/plain
[03] Content-Length: 5
[04] Date: Mon, 27 Dec 2004 20:17:42 GMT
[05] Server: Apache Coyote/1.0
[06]
[07] <ok/>
```

Es handelt sich dabei um die Bestätigung, dass die *Applikation* die XML-Befehlsantwort korrekt erhalten hat. Die Nutzlast der HTTP Response muss dazu aus dem String `<ok/>` in genau dieser Schreibweise bestehen. Der *Header* `Content-Type` darf wahlweise den Wert `text/xml` oder `text/plain` (so wie hier in Zeile 2) oder `text/html` aufweisen.

### *Finaler HTTP Response von RedirectURL*

Schließlich finden Sie nachfolgend noch die finale Antwort der *Applikation* hinter der `RedirectURL` an den Browser des *Bürgers*. Diese wird übermittelt, sobald die *Applikation* die XML-Befehlsantwort der *Bürgerkarten-Umgebung* ausgewertet hat. Bitte beachten Sie, dass sie aus Gründen der besseren Lesbarkeit gekürzt (gekennzeichnet durch `...`) wurde.

```
[01] HTTP/1.1 200 OK
[02] Content-Type: text/html
[03] Content-Length: 1987
[04] Date: Mon, 27 Dec 2004 20:17:51 GMT
[05] Server: Apache Coyote/1.0
[06]
[07] <html>
[08] <head>
[09] <title>Redirect: Synchronisation erfolgreich</title>
[10] </head>
```

```
[11] <body>
[12] <p>Die Antwort von der BKU ist eingetroffen:</p>
[13] <p><pre><?xml version="1.0" encoding="UTF-8"?>
[14] <GetPropertiesResponse xmlns="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[15] ...
[16] </GetPropertiesResponse></pre></p>
[17] </body>
[18] </html>
```

#### *Downloads zu diesem Beispiel*

- examples/bindings/redirect/Request.html
- examples/bindings/redirect/Browser.HttpRequest.txt
- examples/bindings/redirect/Browser.HttpResponse.txt
- examples/bindings/redirect/RedirectLoc.HttpRequest.txt
- examples/bindings/redirect/RedirectLoc.HttpResponse.txt
- examples/bindings/redirect/DataURL.HttpRequest.txt
- examples/bindings/redirect/DataURL.HttpResponse.txt
- examples/bindings/redirect/RedirectLoc.Final.HttpResponse.txt
- examples/bindings/redirect/Redirect.java

#### *Weiterführende Informationen*

- Abschnitt 3, „HTTP-Bindung“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer
- Abschnitt 3.2.3, „Schrittweiser Ablauf“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer
- Abschnitt 3.3.2.2, „HTTP-Request an DataURL“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer
- Abschnitt 3.3.2.2, „HTTP-Request an DataURL“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer

### 3.2.2.2. Synchrone Benutzerführung mittels DataURL

Im folgenden Beispiel wird im HTTP Request neben der XML-Befehlsanfrage als *Formular-Parameter* `XMLRequest` ein weiterer *Formular-Parameter* angegeben:

- Der *Formular-Parameter* `DataURL` enthält als Wert eine URL des *Applikations*-Servers, an den die *Bürgerkarten-Umgebung* die XML-Befehlsantwort übermitteln soll.

Durch die Kombination dieses Formularparameters mit einer gegenüber dem Abschnitt 3.2.2.1, „Asynchrone Benutzerführung mittels RedirectURL und DataURL“ veränderten Antwort des *Applikations*-Servers wird im Beispiel der folgende Ablauf realisiert:

1. Der *Bürger* sendet einen HTTP Request mit dem `XMLRequest` (im konkreten Fall mit dem Befehl *InfoboxAvailable*) und dem weiteren *Formular-Parameter* `DataURL` an die *Bürgerkarten-Umgebung*.
2. Die *Bürgerkarten-Umgebung* bearbeitet die XML-Befehlsanfrage und übermittelt die XML-Befehlsantwort in einem HTTP *POST* Request an die in der `DataURL` angegebene Lokation, einem Teil der *Applikation*.
3. Die *Applikation* antwortet der *Bürgerkarten-Umgebung* in der HTTP Response mit beliebigen Daten in der Nutzlast, z.B. mit einem HTML-Dokument, das den nächsten Schritt im Verfahren darstellt.
4. Die *Bürgerkarten-Umgebung* sendet dem *Bürger* in der HTTP Response als Antwort auf den HTTP Request von Schritt 1 exakt jene Nutzdaten, die sie zuvor in Schritt 3 von der *Applikation* übermittelt bekommen hat.

#### *HTML-Formular*

Das folgende HTML-Formular wurde von der Applikation dynamisch generiert und kann unter der Voraussetzung der installierten ??? von der URL `http://localhost:8080/SL12Tutorial/DataURL` geladen werden. Bitte beachten Sie, dass es aus Gründen der besseren Lesbarkeit formatiert und umgebrochen (gekennzeichnet durch das Zeichen \ am Ende einer Zeile) wurde.

```
[01] <html>
[02]   <head>
[03]     <title>DataURL: Start</title>
[04]   </head>
[05]   <body>
[06]     <form method="post" action="http://127.0.0.1:3495/http-security-layer-request">
[07]       <input name="XMLRequest" type="hidden"
[08]         value="<?xml version='1.0' encoding='UTF-8'?><InfoboxAvailableRequest \
[09] xmlns='http://www.buergerkarte.at/namespaces/securitylayer/1.2#'/>" />
[10]       <input name="DataURL" type="hidden"
[11]         value="http://localhost:8080/SL12Tutorial/DataURL?use=dataurl" />
[12]       <input name="Request absenden" type="submit" />
[13]     </form>
[14]   </body>
[15] </html>
```

In Zeile 10 - 12 ist der *Formular-Parameter* `DataURL` kodiert.

#### *HTTP Request des Browsers an die Bürgerkarten-Umgebung*

Nachfolgend sehen Sie den HTTP Request, der aus dem Absenden des Formulars in obiger HTML-Seite resultiert. Bitte beachten Sie, dass er aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen \ am Ende einer Zeile) wurde.

```
[01] POST /http-security-layer-request HTTP/1.1
[02] Host: 127.0.0.1
[03] User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.5) Gecko/20041107 \
[04] Firefox/1.0
[05] Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;\
[06] q=0.8,image/png,*/*;q=0.5
[07] Accept-Language: de-at,de;q=0.7,en;q=0.3
[08] Accept-Encoding: gzip,deflate
[09] Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
[10] Keep-Alive: 300
[11] Connection: keep-alive
```

```
[12] Content-Type: application/x-www-form-urlencoded
[13] Content-Length: 284
[14]
[15] XMLRequest=%3C%3Fxml+version%3D%271.0%27+encoding%3D%27UTF-8%27%3F%3E%3CInfobox\
[16] AvailableRequest+xmlns%3D%27http%3A%2F%2Fwww.buergerkarte.at%2Fnamespaces%2Fsec\
[17] uritylayer%2F1.2%23%27%2F%3E&DataURL=http%3A%2F%2Flocalhost%3A8080%2FSL12Tutori\
[18] al%2FDataURL%3Fuse%3Ddataurl&Request+absenden
```

### *HTTP Request der Bürgerkarten-Umgebung an DataURL*

Als Nächstes sehen Sie den HTTP Request der *Bürgerkarten-Umgebung* an die hinter der `DataURL` stehende *Applikation*. Dieser enthält die u. a. die XML-Befehlsantwort. Bitte beachten Sie, dass er aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) wurde.

```
[01] POST /SL12Tutorial/DataURL?use=dataurl HTTP/1.1
[02] Referer: localhost
[03] User-Agent: citizen-card-environment/1.2 trustDeskbasic/2.2.3-developer
[04] Content-Type: application/x-www-form-urlencoded
[05] Host: 127.0.0.1
[06] Content-Length: 763
[07] Connection: Keep-Alive
[08] Cache-Control: no-cache
[09]
[10] XMLResponse=%3C%3Fxml+version%3D%221.0%22+encoding%3D%22UTF-8%22%3F%3E%0D%0A%3C\
[11] sl%3AInfoboxAvailableResponse+xmlns%3Asl%3D%22http%3A%2F%2Fwww.buergerkarte.at%\
[12] 2Fnamespaces%2Fsecuritylayer%2F1.2%23%22%3E%0D%0A%3Csl%3AInfoboxIdentifier%3ECe\
[13] rtificates%3C%2Fsl%3AInfoboxIdentifier%3E%0D%0A%3Csl%3AInfoboxIdentifier%3EIden\
[14] tityLink%3C%2Fsl%3AInfoboxIdentifier%3E%0D%0A%3Csl%3AInfoboxIdentifier%3ECompre\
[15] ssedIdentityLink%3C%2Fsl%3AInfoboxIdentifier%3E%0D%0A%3Csl%3AInfoboxIdentifier%\
[16] 3EMandates%3C%2Fsl%3AInfoboxIdentifier%3E%0D%0A%3Csl%3AInfoboxIdentifier%3ETuto\
[17] rialBinary%3C%2Fsl%3AInfoboxIdentifier%3E%0D%0A%3Csl%3AInfoboxIdentifier%3ETuto\
[18] rialAssocArray%3C%2Fsl%3AInfoboxIdentifier%3E%0D%0A%3C%2Fsl%3AInfoboxAvailableR\
[19] esponse%3E&ResponseType=HTTP-Security-Layer-RESPONSE
```

Es handelt sich bei diesem Request wiederum um HTTP POST, wie in Zeile 1 zu sehen ist.

Zeile 4 enthält die obligatorische Abschnitt 3.3.2.2, „HTTP-Request an DataURL“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer der *Bürgerkarten-Umgebung* als *User Agent*.

Die Nutzlast enthält wiederum eine Reihe von *Formular-Parametern*, die (so wie hier) als `application/x-www-form-urlencoded` oder als `multipart/form-data` Abschnitt 3.3.2.2, „HTTP-Request an DataURL“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer sein können. Jedenfalls enthalten sein müssen (so wie hier) die Parameter `XMLResponse` mit der XML-Befehlsantwort sowie der Parameter `ResponseType` mit dem fixen Wert `HTTP-Security-Layer-RESPONSE`.

### *HTTP Response von DataURL an die Bürgerkarten-Umgebung*

Im Folgenden finden Sie die HTTP Response der hinter der `DataURL` stehenden *Applikation* an die *Bürgerkarten-Umgebung*. Bitte beachten Sie, dass sie aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) wurde.

```
[01] HTTP/1.1 200 OK
[02] Content-Type: text/html
[03] Content-Length: 773
[04] Date: Mon, 27 Dec 2004 21:21:10 GMT
[05] Server: Apache Coyote/1.0
[06]
[07] <html>
[08] <head>
[09] <title>DataURL: Auswertung</title>
[10] </head>
[11] <body>
[12] <p>Die Antwort von der BKU ist eingetroffen:</p>
[13] <p><pre style="background-color: silver;">&lt;?xml version="1.0" encoding="UTF-8"?>
[14] &lt;sl:InfoboxAvailableResponse xmlns:sl="http://www.buergerkarte.at/namespaces/\
[15] securitylayer/1.2#">
[16] &lt;sl:InfoboxIdentifier>Certificates&lt;/sl:InfoboxIdentifier>
[17] &lt;sl:InfoboxIdentifier>IdentityLink&lt;/sl:InfoboxIdentifier>
[18] &lt;sl:InfoboxIdentifier>CompressedIdentityLink&lt;/sl:InfoboxIdentifier>
[19] &lt;sl:InfoboxIdentifier>Mandates&lt;/sl:InfoboxIdentifier>
[20] &lt;sl:InfoboxIdentifier>TutorialBinary&lt;/sl:InfoboxIdentifier>
[21] &lt;sl:InfoboxIdentifier>TutorialAssocArray&lt;/sl:InfoboxIdentifier>
[22] &lt;/sl:InfoboxAvailableResponse></pre></p>
[23] </body>
[24] </html>
```

Die Nutzlast der HTTP Response besteht im Unterschied zum Abschnitt 3.2.2.1, „Asynchrone Benutzerführung mittels RedirectURL und DataURL“ nicht aus dem simplen String `<ok/>`, sondern aus einem HTML-Dokument (vgl. Zeile 7 - 24). Dieses HTML-Dokument wird die *Bürgerkarten-Umgebung* im nächsten und zugleich letzten Schritt in der HTTP Response an den Browser des *Bürgers* senden.

### *HTTP Response der Bürgerkarten-Umgebung an den Browser*

Schließlich sehen Sie unterbei noch die HTT -Response der *Bürgerkarten-Umgebung* an den Browser des *Bürgers.* Bitte beachten Sie, dass sie aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) wurde.

```
[01] HTTP/1.1 200 OK
[02] Content-Type: text/html
[03] Content-Length: 773
[04] Date: Mon, 27 Dec 2004 21:21:10 GMT
[05] Server: Apache Coyote/1.0
[06]
[07] <html>
[08] <head>
[09] <title>DataURL: Auswertung</title>
[10] </head>
[11] <body>
[12] <p>Die Antwort von der BKU ist eingetroffen:</p>
[13] <p><pre style="background-color: silver;">&lt;?xml version="1.0" encoding="UTF-8"?>
[14] &lt;sl:InfoboxAvailableResponse xmlns:sl="http://www.buergerkarte.at/namespaces/\
[15] securitylayer/1.2#">
```

```
[16] &lt;sl:InfoboxIdentifier>Certificates&lt;/sl:InfoboxIdentifier>
[17] &lt;sl:InfoboxIdentifier>IdentityLink&lt;/sl:InfoboxIdentifier>
[18] &lt;sl:InfoboxIdentifier>CompressedIdentityLink&lt;/sl:InfoboxIdentifier>
[19] &lt;sl:InfoboxIdentifier>Mandates&lt;/sl:InfoboxIdentifier>
[20] &lt;sl:InfoboxIdentifier>TutorialBinary&lt;/sl:InfoboxIdentifier>
[21] &lt;sl:InfoboxIdentifier>TutorialAssocArray&lt;/sl:InfoboxIdentifier>
[22] &lt;/sl:InfoboxAvailableResponse></pre></p>
[23] </body>
[24] </html>
```

Ein Vergleich mit der HTTP Response der *Applikation* an die *Bürgerkarten-Umgebung* zeigt, dass die beiden Antworten ident sind.

#### *Downloads zu diesem Beispiel*

- examples/bindings/dataurl/Request.html
- examples/bindings/dataurl/Browser.HttpRequest.txt
- examples/bindings/dataurl/DataURL.HttpRequest.txt
- examples/bindings/dataurl/DataURL.HttpResponse.txt
- examples/bindings/dataurl/Browser.HttpResponse.txt
- examples/bindings/dataurl/DataURL.java

#### *Weiterführende Informationen*

- Abschnitt 3, „HTTP-Bindung“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer
- Abschnitt 3.2.3, „Schrittweiser Ablauf“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer
- Abschnitt 3.3.2.2, „HTTP-Request an DataURL“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer
- Abschnitt 3.3.2.2, „HTTP-Request an DataURL“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer

### 3.2.2.3. Verwendung von Weitergabe-Parametern und Weitergabe-Headern

Der Ablauf des folgenden Beispiels ist grundsätzlich gleich wie jener im Abschnitt 3.2.2.2, „Synchrone Benutzerführung mittels DataURL“ Zusätzlich zeigt es jedoch die Verwendung von *Abschnitt 3.2.1.1, „Terminologie“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer* Abschnitt 3.2.1.1, „Terminologie“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer:

Werden solche Formular-Felder, die einer bestimmten Nomenklatur entsprechen müssen, im HTTP Request des Browsers an die *Bürgerkarten-Umgebung* angegeben, werden diese in identischer Form zusammen mit der XML-Befehlsantwort im HTTP Request der *Bürgerkarten-Umgebung* an die *Applikation* weitergeleitet. *Weitergabe-Parameter* erscheinen auch dort wiederum als Formular-Paramter, *Weitergabe-Header* werden dort als HTTP *Header* kodiert.

Einen praktischen Anwendungsfall für einen Weitergabe-Parameter finden Sie in Abschnitt 3.2.3.2, „Signieren eines Antrags mit Beilagen“.

#### *HTML-Formular*

Das folgende HTML-Formular wurde von der Applikation dynamisch generiert und kann unter der Voraussetzung der installierten ??? von der URL `http://localhost:8080/SL12Tutorial/PassOn` geladen werden. Bitte beachten Sie, dass es aus Gründen der besseren Lesbarkeit formatiert und umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) wurde.

```
[01] <html>
[02]   <head>
[03]     <title>Weitergabe: Start</title>
[04]   </head>
[05]   <body>
[06]     <form method="post" action="http://127.0.0.1:13495/http-security-layer-request">
[07]       <input name="XMLRequest" type="hidden"
[08]         value="<?xml version='1.0' encoding='UTF-8'?><NullOperationRequest \
[09] xmlns='http://www.buergerkarte.at/namespaces/securitylayer/1.2#'/>" />
[10]       <input name="DataURL" type="hidden"
[11]         value="http://localhost:18080/SL12Tutorial/PassOn?use=dataurl" />
[12]       <input name="WeitergabeParameter_" type="hidden"
[13]         value="Inhalt des Weitergabe-Parameters" />
[14]       <input name="WeitergabeHeader__" type="hidden"
[15]         value="X-Test-Weitergabe: Inhalt des Weitergabe-Headers" />
[16]       <input name="Request absenden" type="submit" />
[17]     </form>
[18]   </body>
[19] </html>
```

Neben den schon bekannten *Formular-Parametern* `XMLRequest` (enthält den Befehl *NullOperation*) und `DataURL` finden Sie in Zeile 12 - 13 den *Weitergabe-Parameter* `WeitergabeParameter_` (als solcher qualifiziert durch den nachlaufenden Unterstrich _), sowie in Zeile 14 - 15 den *Weitergabe-Header* `WeitergabeHeader__` (als solcher qualifiziert durch den doppelten nachlaufenden Unterstrich __). Beachten Sie bitte, dass bei einem *Weitergabe-Header* der Name des HTTP *Headers* nicht durch das Attribut `name` (vgl. Zeile 14) bestimmt ist, sondern dass das Attribut `value` (vgl. Zeile 15) sowohl den Namen als auch den Wert des HTTP *Headers* enthält (konkret lautet hier der Name des *Headers* `X-Test-Weitergabe` und der Wert des *Headers* `Inhalt des Weitergabe-Headers`).

#### *HTTP Request des Browsers an die Bürgerkarten-Umgebung*

Nachfolgend sehen Sie den HTTP Request, der aus dem Absenden des Formulars in obiger HTML-Seite resultiert. Bitte beachten Sie, dass er aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) wurde.

```
[01] POST /http-security-layer-request HTTP/1.1
[02] Host: 127.0.0.1
[03] User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.5) Gecko/20041107 \
[04] Firefox/1.0
[05] Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;\
[06] q=0.8,image/png,*/*;q=0.5
[07] Accept-Language: de-at,de;q=0.7,en;q=0.3
[08] Accept-Encoding: gzip,deflate
[09] Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
[10] Keep-Alive: 300
[11] Connection: keep-alive
[12] Content-Type: application/x-www-form-urlencoded
[13] Content-Length: 404
[14]
```

```
[15] XMLRequest=%3C%3Fxml+version%3D%271.0%27+encoding%3D%27UTF-8%27%3F%3E%3CNullOpe\
[16] rationRequest+xmlns%3D%27http%3A%2F%2Fwww.buergerkarte.at%2Fnamespaces%2Fsecuri\
[17] tylayer%2F1.2%23%27%2F%3E&DataURL=http%3A%2F%2Flocalhost%3A8080%2FSL12Tutorial%\
[18] 2FPassOn%3Fuse%3Ddataurl&WeitergabeParameter_=Inhalt+des+Weitergabe-Parameters&\
[19] WeitergabeHeader__=X-Test-Weitergabe%3A+Inhalt+des+Weitergabe-Headers&Request+a\
[20] bsenden
```

### HTTP Request der Bürgerkarten-Umgebung an DataURL

Als Nächstes sehen Sie den HTTP Request der *Bürgerkarten-Umgebung* an die hinter der `DataURL` stehende *Applikation*. Dieser enthält die u. a. die XML-Befehlsantwort, den *Weitergabe-Parameter* `WeitergabeParameter_`, sowie den *Weitergabe-Header* `X-Test-Weitergabe`. Bitte beachten Sie, dass der HTTP Request aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) wurde.

```
[01] POST /SL12Tutorial/PassOn?use=dataurl HTTP/1.1
[02] Referer: localhost
[03] User-Agent: citizen-card-environment/1.2 trustDeskbasic/2.2.3-developer
[04] Content-Type: application/x-www-form-urlencoded
[05] X-Test-Weitergabe: Inhalt des Weitergabe-Headers
[06] Host: 127.0.0.1
[07] Content-Length: 291
[08] Connection: Keep-Alive
[09] Cache-Control: no-cache
[10]
[11] XMLResponse=%3C%3Fxml+version%3D%221.0%22+encoding%3D%22UTF-8%22%3F%3E%3Csl%3AN\
[12] ullOperationResponse+xmlns%3Asl%3D%22http%3A%2F%2Fwww.buergerkarte.at%2Fnamespa\
[13] ces%2Fsecuritylayer%2F1.2%23%22%2F%3E&WeitergabeParameter_=Inhalt+des+Weitergab\
[14] e-Parameters&ResponseType=HTTP-Security-Layer-RESPONSE
```

In Zeile 5 erkennt man den *Weitergabe-Header* `X-Test-Weitergabe`.

In Zeile 13 - 14 erkennt man in der Nutzlast den *Weitergabe-Parameter* `WeitergabeParameter_`.

### HTTP Response von DataURL an die Bürgerkarten-Umgebung

Im Folgenden finden Sie die HTTP Response der hinter der `DataURL` stehenden *Applikation* an die *Bürgerkarten-Umgebung*. Bitte beachten Sie, dass sie aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) wurde.

```
[01] HTTP/1.1 200 OK
[02] Content-Type: text/html
[03] Content-Length: 479
[04] Date: Mon, 27 Dec 2004 22:03:58 GMT
[05] Server: Apache Coyote/1.0
[06]
[07] <html>
[08] <head>
[09] <title>Weitergabe: Auswertung</title>
[10] </head>
[11] <body>
[12] <p>Der Wert des Weitergabe-Parameters <code style="background-color: silver;">\
[13] WeitergabeParameter_</code> lautet: <code style="background-color: silver;">\
[14] Inhalt des Weitergabe-Parameters</code></p>
[15] <p>Der Wert des Weitergabe-Headers <code style="background-color: silver;">\
[16] X-Test-Weitergabe</code> lautet: <code style="background-color: silver;">\
[17] Inhalt des Weitergabe-Headers</code></p>
[18] </body>
[19] </html>
```

Die *Applikation* hat den HTTP-Request der *Bürgerkarten-Umgebung* ausgewertet und daraus ein entsprechendes HTML-Dokument generiert, das von der *Bürgerkarten-Umgebung* in weiterer Folge an den Browser des *Bürgers* weiter übermittelt wird.

### HTTP Response der Bürgerkarten-Umgebung an den Browser

Schließlich sehen Sie unterbei noch die HTTP Response der *Bürgerkarten-Umgebung* an den Browser des *Bürgers.* Bitte beachten Sie, dass sie aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) wurde.

```
[01] HTTP/1.1 200 OK
[02] Content-Type: text/html
[03] Content-Length: 479
[04] Date: Mon, 27 Dec 2004 22:03:58 GMT
[05] Server: Apache Coyote/1.0
[06]
[07] <html>
[08] <head>
[09] <title>Weitergabe: Auswertung</title>
[10] </head>
[11] <body>
[12] <p>Der Wert des Weitergabe-Parameters <code style="background-color: silver;">\
[13] WeitergabeParameter_</code> lautet: <code style="background-color: silver;">\
[14] Inhalt des Weitergabe-Parameters</code></p>
[15] <p>Der Wert des Weitergabe-Headers <code style="background-color: silver;">\
[16] X-Test-Weitergabe</code> lautet: <code style="background-color: silver;">\
[17] Inhalt des Weitergabe-Headers</code></p>
[18] </body>
[19] </html>
```

### Downloads zu diesem Beispiel

- examples/bindings/passon/Request.html
- examples/bindings/passon/Browser.HttpRequest.txt
- examples/bindings/passon/DataURL.HttpRequest.txt
- examples/bindings/passon/DataURL.HttpResponse.txt
- examples/bindings/passon/Browser.HttpResponse.txt
- examples/bindings/passon/PassOn.java

#### *Weiterführende Informationen*

- Abschnitt 3, „HTTP-Bindung“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer
- Abschnitt 3.2.1.1, „Terminologie“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer
- Abschnitt 3.3.2.2, „HTTP-Request an DataURL“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer

### 3.2.2.4. 3.2.2.4 Befehlskaskadierung über DataURL

Mit diesem Beispiel soll nun noch eine weitere Variationsmöglichkeit des Grundablaufs aus Abschnitt 3.2.2.2, „Synchrone Benutzerführung mittels DataURL“ gezeigt werden.

Bisher wurden zwei mögliche Reaktionen des hinter der `DataURL` stehenden *Applikations*-Servers auf den HTTP Request der *Bürgerkarten-Umgebung* demonstriert: Einerseits die einfache Bestätigung des Erhalts der XML-Befehlsantwort, andererseits die Übermittlung (z.B.) eines HTML-Dokuments, welches von der *Bürgerkarten-Umgebung* in identer Form an den Browser des *Bürgers* weiter zu übermitteln ist.

Als dritte Möglichkeit kann der *Applikations*-Server in der HTTP Response an die *Bürgerkarten-Umgebung* einen XML-Befehl übermitteln, der dann von dieser zu verarbeiten ist. Dabei ist zwischen drei Möglichkeiten zu differenzieren:

1. Der *Applikations*-Server sendet lediglich einen neuen XML-Befehl. Alle übrigen Formular-Felder (*Formular-Parameter*, *Weitergabe-Parameter*, *Weitergabe-Header* und sonstige Formular-Felder) bleiben unverändert vom zuletzt von der *Bürgerkarten-Umgebung* abgearbeiteten Befehl erhalten.
2. Der *Applikations*-Server sendet einen neuen XML-Befehl als auch einen neuen Wert für den *Formular-Parameter* `DataURL`. Alle übrigen Formular-Felder bleiben unverändert vom zuletzt von der *Bürgerkarten-Umgebung* abgearbeiteten Befehl erhalten.
3. Der *Applikations*-Server sendet einen komplett neuen Satz von Formular-Feldern (*Formular-Parameter*, *Weitergabe-Parameter*, *Weitergabe-Header*, sonstige Formular-Felder).

In diesem Beispiel wird von Möglichkeit 2 Gebrauch gemacht: Zunächst wird vom HTML-Formular aus der Befehl *InfoboxAvailable* an die *Bürgerkarten-Umgebung* abgesetzt. Diese sendet die entsprechende XML-Befehlsantwort an die `DataURL`. Der *Applikations*-Server analysiert nun diese Antwort und reagiert abhängig davon, ob eine Infobox mit dem Namen `TutorialBinary` vorhanden ist, unterschiedlich:

- Ist die Infobox vorhanden, sendet er in der HTTP Response an die *Bürgerkarten-Umgebung* einen neuen XML-Befehl *InfoboxUpdate* sowie einen neuen Wert für die `DataURL`.
- Ist die Infobox nicht vorhanden, sendet er stattdessen einen neuen XML-Befehl *InfoboxCreate* sowie einen anderen neuen Wert für die `DataURL`.

Im zweiten Fall sendet der *Applikations*-Server dann den XML-Befehl *InfoboxUpdate* als dritten Befehl im Reaktion auf den Erhalt der XML-Befehlsantwort für *InfoboxCreate*. Dieser zweite Fall wird im Folgenden dargestellt.

#### *HTML-Formular*

Das folgende HTML-Formular wurde von der Applikation dynamisch generiert und kann unter der Voraussetzung der installierten ??? von der URL `http://localhost:8080/SL12Tutorial/Cascade` geladen werden. Bitte beachten Sie, dass es aus Gründen der besseren Lesbarkeit formatiert und umgebrochen (gekennzeichnet durch das Zeichen \ am Ende einer Zeile) wurde.

```
[01] <html>
[02]   <head>
[03]     <title>Kaskadierung: Start</title>
[04]   </head>
[05]   <body>
[06]     <form method="post" action="http://127.0.0.1:3495/http-security-layer-request">
[07]       <input name="XMLRequest" type="hidden"
[08]         value="<?xml version='1.0' encoding='UTF-8'?><InfoboxAvailableRequest \
[09] xmlns='http://www.buergerkarte.at/namespaces/securitylayer/1.2#'/>" />
[10]       <input name="DataURL" type="hidden"
[11]         value="http://localhost:8080/SL12Tutorial/Cascade?use=available" />
[12]       <input name="Request absenden" type="submit" />
[13]     </form>
[14]   </body>
[15] </html>
```

#### *HTTP Request des Browsers an die Bürgerkarten-Umgebung*

Nachfolgend sehen Sie den HTTP Request, der aus dem Absenden des Formulars in obiger HTML-Seite resultiert. Bitte beachten Sie, dass er aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen \ am Ende einer Zeile) wurde.

```
[01] POST /http-security-layer-request HTTP/1.1
[02] Host: 127.0.0.1
[03] User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.5) Gecko/20041107 \
[04] Firefox/1.0
[05] Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;\
[06] q=0.8,image/png,*/*;q=0.5
[07] Accept-Language: de-at,de;q=0.7,en;q=0.3
[08] Accept-Encoding: gzip,deflate
[09] Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
[10] Keep-Alive: 300
[11] Connection: keep-alive
[12] Referer: http://localhost:8080/SL12Tutorial/Cascade
[13] Content-Type: application/x-www-form-urlencoded
[14] Content-Length: 286
[15]
[16] XMLRequest=%3C%3Fxml+version%3D%271.0%27+encoding%3D%27UTF-8%27%3F%3E%3CInfobox\
[17] AvailableRequest+xmlns%3D%27http%3A%2F%2Fwww.buergerkarte.at%2Fnamespaces%2Fsec\
[18] uritylayer%2F1.2%23%27%2F%3E&DataURL=http%3A%2F%2Flocalhost%3A8080%2FSL12Tutori\
[19] al%2FCascade%3Fuse%3Davailable&Request+absenden
```

#### *Erster HTTP Request der Bürgerkarten-Umgebung an DataURL*

Als Nächstes sehen Sie den ersten HTTP Request der *Bürgerkarten-Umgebung* an die hinter der `DataURL` stehende *Applikation*. Dieser enthält die u. a. die XML-Befehlsantwort für *InfoboxAvailable*. Bitte beachten Sie, dass der HTTP Request aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen \ am Ende einer Zeile) wurde.

```
[01] POST /SL12Tutorial/Cascade?use=available HTTP/1.1
[02] Referer: localhost
[03] User-Agent: citizen-card-environment/1.2 trustDeskbasic/2.2.3-developer
[04] Content-Type: application/x-www-form-urlencoded
[05] Host: 127.0.0.1
```

```
[06] Content-Length: 684
[07] Connection: Keep-Alive
[08] Cache-Control: no-cache
[09]
[10] XMLResponse=%3C%3Fxml+version%3D%221.0%22+encoding%3D%22UTF-8%22%3F%3E%0D%0A%3C\
[11] sl%3AInfoboxAvailableResponse+xmlns%3Asl%3D%22http%3A%2F%2Fwww.buergerkarte.at%\
[12] 2Fnamespaces%2Fsecuritylayer%2F1.2%23%22%3E%0D%0A%3Csl%3AInfoboxIdentifier%3ECe\
[13] rtificates%3C%2Fsl%3AInfoboxIdentifier%3E%0D%0A%3Csl%3AInfoboxIdentifier%3EIden\
[14] tityLink%3C%2Fsl%3AInfoboxIdentifier%3E%0D%0A%3Csl%3AInfoboxIdentifier%3ECompre\
[15] ssedIdentityLink%3C%2Fsl%3AInfoboxIdentifier%3E%0D%0A%3Csl%3AInfoboxIdentifier%\
[16] 3EMandates%3C%2Fsl%3AInfoboxIdentifier%3E%0D%0A%3Csl%3AInfoboxIdentifier%3ETuto\
[17] rialAssocArray%3C%2Fsl%3AInfoboxIdentifier%3E%0D%0A%3C%2Fsl%3AInfoboxAvailableR\
[18] esponse%3E&ResponseType=HTTP-Security-Layer-RESPONSE
```

### *Erste HTTP Response von DataURL an die Bürgerkarten-Umgebung*

Im Folgenden finden Sie die erste HTTP Response der hinter der `DataURL` stehenden *Applikation* an die *Bürgerkarten-Umgebung*. Bitte beachten Sie, dass sie aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) wurde.

```
[01] HTTP/1.1 307 Temporary Redirect
[02] Location: http://localhost:8080/SL12Tutorial/Cascade?use=update
[03] Content-Type: text/xml
[04] Content-Length: 442
[05] Date: Tue, 28 Dec 2004 09:55:39 GMT
[06] Server: Apache Coyote/1.0
[07]
[08] <?xml version='1.0' encoding='UTF-8'?>
[09] <sl:InfoboxCreateRequest xmlns:sl='http://www.buergerkarte.at/namespaces/securi\
[10] tylayer/1.2#'>
[11]   <sl:InfoboxIdentifier>TutorialBinary</sl:InfoboxIdentifier>
[12]   <sl:InfoboxType>BinaryFile</sl:InfoboxType>
[13]   <sl:Creator>Tutorium zur Ã¶sterreichischen BÃ¼rgerkarte</sl:Creator>
[14]   <sl:Purpose>Demonstriert das Anlegen, Lesen, VerÃ¤ndern und LÃ¶schen einer Bi\
[15] nÃ¤rdatei.</sl:Purpose>
[16] </sl:InfoboxCreateRequest>
```

Es wird hier von Möglichkeit 2 Gebrauch gemacht, einen neuen XML-Befehl (*InfoboxCreate*) an die *Bürgerkarten-Umgebung* zu senden.

Gesendet wird daher ein HTTP Redirect (vgl. Code `307` in Zeile 1) mit einer Nutzlast vom Typ `text/xml` (vgl. Zeile 3).

Der neue Wert für den Formular-Parameter `DataURL` wird als Wert des HTTP *Headers* `Location` in Zeile 2 übermittelt.

Der neue XML-Befehl wird als Nutzlast der HTTP Response (vgl. Zeile 8 - 16) gesendet.

### *Zweiter HTTP Request der Bürgerkarten-Umgebung an DataURL*

Im Weiteren sehen Sie den zweiten HTTP Request der *Bürgerkarten-Umgebung* an die hinter der `DataURL` stehende *Applikation*. Dieser enthält die u. a. die XML-Befehlsantwort für *InfoboxCreate*. Bitte beachten Sie, dass der HTTP Request aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) wurde.

```
[01] POST /SL12Tutorial/Cascade?use=update HTTP/1.1
[02] Referer: localhost
[03] User-Agent: citizen-card-environment/1.2 trustDeskbasic/2.2.3-developer
[04] Content-Type: application/x-www-form-urlencoded
[05] Host: localhost:8080
[06] Content-Length: 407
[07] Connection: Keep-Alive
[08] Cache-Control: no-cache
[09]
[10] XMLResponse=%3C%3Fxml+version%3D%221.0%22+encoding%3D%22UTF-8%22%3F%3E%0D%0A%3C\
[11] sl%3AInfoboxCreateResponse+xmlns%3Asl%3D%22http%3A%2F%2Fwww.buergerkarte.at%2Fn\
[12] amespaces%2Fsecuritylayer%2F1.2%23%22%2F%3E&ResponseType=HTTP-Security-Layer-RE\
[13] SPONSE
```

### *Zweite HTTP Response von DataURL an die Bürgerkarten-Umgebung*

Im Folgenden finden Sie die zweite HTTP Response der hinter der `DataURL` stehenden *Applikation* an die *Bürgerkarten-Umgebung*. Bitte beachten Sie, dass sie aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) wurde.

```
[01] HTTP/1.1 307 Temporary Redirect
[02] Location: http://localhost:8080/SL12Tutorial/Cascade?use=done
[03] Content-Type: text/xml
[04] Content-Length: 491
[05] Date: Tue, 28 Dec 2004 09:55:42 GMT
[06] Server: Apache Coyote/1.0
[07]
[08] <?xml version='1.0' encoding='UTF-8'?>
[09] <sl:InfoboxUpdateRequest xmlns:sl='http://www.buergerkarte.at/namespaces/securi\
[10] tylayer/1.2#'>
[11]   <sl:InfoboxIdentifier>TutorialBinary</sl:InfoboxIdentifier>
[12]   <sl:BinaryFileParameters>
[13]     <sl:XMLContent>
[14]       <my:Customer xmlns:my='urn:my.namespace'>
[15]         <my:Name>Tassilo Tester</my:Name>
[16]         <my:LastVisit>2005-01-01</my:LastVisit>
[17]       </my:Customer>
[18]     </sl:XMLContent>
[19]   </sl:BinaryFileParameters>
[20] </sl:InfoboxUpdateRequest>
```

Auch hier von Möglichkeit 2 Gebrauch gemacht, um den letzten neuen XML-Befehl (*InfoboxUpdate*) an die *Bürgerkarten-Umgebung* zu senden (siehe erste Response).

### *Dritter HTTP Request der Bürgerkarten-Umgebung an DataURL*

Als Nächstes sehen Sie den dritten HTTP Request der *Bürgerkarten-Umgebung* an die hinter der `DataURL` stehende*Applikation*. Dieser enthält die u. a. die XML-Befehlsantwort für *InfoboxUpdate*. Bitte beachten Sie, dass der HTTP Request aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) wurde.

```
[01] POST /SL12Tutorial/Cascade?use=done HTTP/1.1
[02] Referer: localhost
[03] User-Agent: citizen-card-environment/1.2 trustDeskbasic/2.2.3-developer
[04] Content-Type: application/x-www-form-urlencoded
[05] Host: localhost:8080
[06] Content-Length: 240
[07] Connection: Keep-Alive
[08] Cache-Control: no-cache
[09]
[10] XMLResponse=%3C%3Fxml+version%3D%221.0%22+encoding%3D%22UTF-8%22%3F%3E%0D%0A%3C\
[11] sl%3AInfoboxUpdateResponse+xmlns%3Asl%3D%22http%3A%2F%2Fwww.buergerkarte.at%2Fn\
[12] amespaces%2Fsecuritylayer%2F1.2%23%22%3E&ResponseType=HTTP-Security-Layer-RESPO\
[13] NSE
```

#### *Dritte HTTP Response von DataURL an die Bürgerkarten-Umgebung*

Es folgt die dritte HTTP Response der hinter der `DataURL` stehenden *Applikation* an die *Bürgerkarten-Umgebung*. Bitte beachten Sie, dass sie aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) wurde.

```
[01] HTTP/1.1 200 OK
[02] Content-Type: text/html
[03] Content-Length: 205
[04] Date: Tue, 28 Dec 2004 09:56:08 GMT
[05] Server: Apache Coyote/1.0
[06]
[07] <html>
[08] <head><title>Kaskadierung: Abschluss</title></head>
[09] <body>
[10] <p>Infobox <code style='background-color: silver;'>TutorialBinary</code> \
[11] ist angelegt und auf dem letzten Stand.</p>
[12] </body>
[13] </html>
```

Ähnlich wie in den Beispielen zuvor sendet die *Applikation* zum Abschluß ein HTML-Dokument, das von der *Bürgerkarten-Umgebung* in identer Form an den Browser des *Bürgers* weiter zu übermitteln ist (vgl. Zeile 7 - 13).

#### *HTTP Response der Bürgerkarten-Umgebung an den Browser*

Schließlich sehen Sie unterbei noch die HTTP Response der *Bürgerkarten-Umgebung* an den Browser des *Bürgers.* Bitte beachten Sie, dass sie aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) wurde.

```
[01] HTTP/1.1 200 OK
[02] Content-Type: text/html
[03] Content-Length: 205
[04] Date: Tue, 28 Dec 2004 09:56:08 GMT
[05] Server: Apache Coyote/1.0
[06]
[07] <html>
[08] <head><title>Kaskadierung: Abschluss</title></head>
[09] <body>
[10] <p>Infobox <code style='background-color: silver;'>TutorialBinary</code> \
[11] ist angelegt und auf dem letzten Stand.</p>
[12] </body>
[13] </html>
```

#### *Downloads zu diesem Beispiel*

- examples/bindings/cascade/Request.html
- examples/bindings/cascade/Browser.HttpRequest.txt
- examples/bindings/cascade/DataURL.1.HttpRequest.txt
- examples/bindings/cascade/DataURL.1.HttpResponse.txt
- examples/bindings/cascade/DataURL.2.HttpRequest.txt
- examples/bindings/cascade/DataURL.2.HttpResponse.txt
- examples/bindings/cascade/DataURL.3.HttpRequest.txt
- examples/bindings/cascade/DataURL.3.HttpResponse.txt
- examples/bindings/cascade/Browser.HttpResponse.txt
- examples/bindings/cascade/Cascade.java

#### *Weiterführende Informationen*

- Abschnitt 3, „HTTP-Bindung“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer
- Abschnitt 3.2.3, „Schrittweiser Ablauf“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer
- Abschnitt 3.3.2.2, „HTTP-Request an DataURL“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer

### 3.2.3. Typische Anwendungsfälle

Dieser Abschnitt stellt in Ergänzung zu den Kapiteln Abschnitt 3.2.1, „Resultat zurück an den Browser“ und Abschnitt 3.2.2, „Resultat an den Applikations-Server“ weitere Anwendungsbeispiele für das Senden von XML-Befehlen über HTTP bzw. HTTPS als Transportprotokoll dar. Konkret werden Lösungen für in der Praxis oft auftretende Fragestellungen aufgezeigt.

#### 3.2.3.1. Personenbindung übermitteln und Dokument signieren (Sign On)

Dieses Beispiel präsentiert eine Lösung für die in vielen Fällen nötige Kombination aus dem Auslesen der Infobox `IdentityLink` (Personenbindung) sowie das anschließende Signieren eines Dokuments durch den Bürger.

Der grundsätzliche Ablauf entspricht wiederum jenem aus Abschnitt 3.2.2.4, „3.2.2.4 Befehlskaskadierung über DataURL“:

1. Der Browser des *Bürgers* sendet mittels eines HTML-Formulars den XML-Befehl *InfoboxRead* für das Auslesen der Personenbindung an die *Bürgerkarten-Umgebung*.

2. Die *Bürgerkarten-Umgebung* sendet die XML-Befehlsantwort für *InfoboxRead* an die hinter der `DataURL` stehende *Applikation*.
3. Die *Applikation* antwortet mit einem neuen XML-Befehl *CreateXMLSignature* an die *Bürgerkarten-Umgebung*.
4. Die *Bürgerkarten-Umgebung* sendet die XML-Befehlsantwort für *CreateXMLSignature* an die hinter der `DataURL` stehende *Applikation*.
5. Die *Applikation* antwortet mit einem HTML-Dokument.
6. Die *Bürgerkarten-Umgebung* übermittelt dieses HTML-Dokument an den Browser des *Bürgers*.

#### *HTML-Formular*

Das folgende HTML-Formular wurde von der Applikation dynamisch generiert und kann unter der Voraussetzung der installierten ??? von der URL `http://localhost:8080/SL12Tutorial/SignOn` geladen werden. Bitte beachten Sie, dass es aus Gründen der besseren Lesbarkeit formatiert und umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) wurde.

```
[01] <html>
[02]   <head>
[03]     <title>Sign On: Start</title>
[04]   </head>
[05]   <body>
[06]     <form method="post" action="http://127.0.0.1:3495/http-security-layer-request">
[07]       <input name="XMLRequest" type="hidden"
[08]         value="<?xml version='1.0' encoding='UTF-8'?><InfoboxReadRequest \
[09] xmlns='http://www.buergerkarte.at/namespaces/securitylayer/1.2#'><InfoboxIdentifier>\
[10] IdentityLink</InfoboxIdentifier><BinaryFileParameters ContentIsXMLEntity='true'/>\
[11] </InfoboxReadRequest>" />
[12]       <input name="DataURL" type="hidden"
[13]         value="https://localhost:8443/SL12Tutorial/SignOn;\
[14] jsessionid=B498616CC781C4BA5B0B4BE03CAA523A?use=sign" />
[15]       <input name="Request absenden" type="submit" />
[16]     </form>
[17]   </body>
[18] </html>
```

In Zeile 7 - 11 wird der Formular-Parameter `XMLRequest` mit der XML-Befehlsanfrage kodiert.

In Zeile 12 - 14 wird der Formular-Parameter `DataURL` kodiert. Bitte beachten Sie, dass es sich in diesem Fall um eine URL im Schema `https` handelt. Dies ist notwendig, weil die *Bürgerkarten-Umgebung* die Personenbindung nur an ein Ziel senden darf, das über ein SSL-Serverzertifikat identifiziert und einer Behörde zugeordnet werden kann (entweder dadurch, dass der Domänenname des SSL-Serverzertifikats auf `.gv.at` endet, oder dadurch, dass das SSL-Serverzertifikat die Zertifikatserweiterung *Verwaltungseigenschaft* beinhaltet).

Anmerkung: Wenn Sie dieses Beispiel im privatwirtschaftlichen Bereich verwenden möchten, müssen Sie den Befehl zum Auslesen der Personenbindung so modifizieren, dass die *Bürgerkarten-Umgebung* die Stammzahl in der Personenbindung ausmaskiert (siehe „2.7.4.1.2 Lesen der Personenbindung durch die Privatwirtschaft“). Dann darf die *Bürgerkarten-Umgebung* die Personenbindung auch an eine per SSL-Serverzertifikat identifziertes Ziel senden, dass nicht einer Behörde zugeordnet werden kann.

> **Anmerkung**
>
> Damit Sie dieses Beispiel mit Hilfe der mit diesem Tutorium mitgelieferten ??? *ausprobieren können, müssen Sie für die* Web Application *ein SSL-Serverzertifikat verwenden, das den oben erwähnten Anforderungen genügt. Das kann für Testzwecke durchaus ein selbst ausgestelltes Zertifikat sein, jedoch müssen Sie dieses Zertifikat dann auch in der*
>
> *../introduction/Introduction.html#glossar.Buergerkarten-Umgebung*
>
> *als vertrauenswürdig konfigurieren.*

#### *HTTP Request des Browsers an die Bürgerkarten-Umgebung*

Nachfolgend sehen Sie den HTTP Request, der aus dem Absenden des Formulars in obiger HTML-Seite resultiert. Bitte beachten Sie, dass er aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) wurde.

```
[01] POST /http-security-layer-request HTTP/1.1
[02] Host: 127.0.0.1
[03] User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.5) Gecko/20041107 \
[04] Firefox/1.0
[05] Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;\
[06] q=0.8,image/png,*/*;q=0.5
[07] Accept-Language: de-at,de;q=0.7,en;q=0.3
[08] Accept-Encoding: gzip,deflate
[09] Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
[10] Keep-Alive: 300
[11] Connection: keep-alive
[12] Content-Type: application/x-www-form-urlencoded
[13] Content-Length: 469
[14]
[05] XMLRequest=%3C%3Fxml+version%3D%271.0%27+encoding%3D%27UTF-8%27%3F%3E%3CInfobox\
[16] ReadRequest+xmlns%3D%27http%3A%2F%2Fwww.buergerkarte.at%2Fnamespaces%2Fsecurity\
[17] layer%2F1.2%23%27%3E%3CInfoboxIdentifier%3EIdentityLink%3C%2FInfoboxIdentifier%\
[18] 3E%3CBinaryFileParameters+ContentIsXMLEntity%3D%27true%27%2F%3E%3C%2FInfoboxRea\
[19] dRequest%3E&DataURL=https%3A%2F%2Flocalhost%3A8443%2FSL12Tutorial%2FSignOn%3Bjs\
[20] essionid%3DB498616CC781C4BA5B0B4BE03CAA523A%3Fuse%3Dsign&Request+absenden
```

#### *Erster HTTP Request der Bürgerkarten-Umgebung an DataURL*

Die HTTP Verbindung zur Übermittlung der Personenbindung an die hinter der `DataURL` stehende *Applikation* ist mittels TLS verschlüsselt. Der HTTP Request kann daher hier nicht dargestellt werden.

Er enthält die XML-Antwort zum Befehl *InfoboxRead* mit der ausgelesenen Personenbindung und ist ansonsten weitgehend baugleich mit dem ersten HTTP Request der *Bürgerkarten-Umgebung* aus Abschnitt 3.2.2.4, „3.2.2.4 Befehlskaskadierung über DataURL“.

#### *Erste HTTP Response der Bürgerkarten-Umgebung an DataURL*

Die HTTP Verbindung zur Übermittlung der Personenbindung an die hinter der `DataURL` stehende *Applikation* ist mittels TLS verschlüsselt. Die HTTP Response kann daher hier nicht dargestellt werden.

Sie enthält den XML-Befehl *CreateXMLSignature* zum Signieren eines kurzen HTML-Dokuments (Anmelde-Token für ein *Sign On*) und ist ansonsten weitgehend baugleich mit der ersten HTTP Response der *Bürgerkarten-Umgebung* aus Abschnitt 3.2.2.4, „3.2.2.4 Befehlskaskadierung über DataURL“.

### *Zweiter HTTP Request der Bürgerkarten-Umgebung an DataURL*

Im Folgenden sehen Sie den zweiten HTTP Request der *Bürgerkarten-Umgebung* an die hinter der `DataURL` stehende *Applikation*. Dieser enthält die u. a. die XML-Befehlsantwort für *CreateXMLSignature*. Bitte beachten Sie, dass der HTTP Request aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen \ am Ende einer Zeile) und gekürzt (gekennzeichnet durch . . . ) wurde.

```
[01] POST /SL12Tutorial/SignOn;jsessionid=B498616CC781C4BA5B0B4BE03CAA523A?use=done HTTP/1.1
[02] Referer: localhost
[03] User-Agent: citizen-card-environment/1.2 trustDeskbasic/2.2.3-developer
[04] Content-Type: application/x-www-form-urlencoded
[05] Host: 127.0.0.1
[06] Content-Length: 6209
[07] Connection: Keep-Alive
[08] Cache-Control: no-cache
[09]
[10] XMLResponse=%3C%3Fxml+version%3D%221.0%22+encoding%3D%22UTF-8%22%3F%3E%3Csl11%3\
[11] ACreateXMLSignatureResponse+xmlns%3Asl11%3D%22http%3A%2F%2Fwww.buergerkarte.at%\
[12] ...
[13] dsig%3ASignature%3E%3C%2Fsl11%3ACreateXMLSignatureResponse%3E&ResponseType=HTTP\
[14] -Security-Layer-RESPONSE
```

### *Dritte HTTP Response von DataURL an die Bürgerkarten-Umgebung*

Es folgt die zweite HTTP Response der hinter der `DataURL` stehenden *Applikation* an die *Bürgerkarten-Umgebung*. Bitte beachten Sie, dass sie aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen \ am Ende einer Zeile) wurde.

```
[01] HTTP/1.1 200 OK
[02] Content-Type: text/html
[03] Transfer-Encoding: chunked
[04] Date: Tue, 28 Dec 2004 12:30:18 GMT
[05] Server: Apache Coyote/1.0
[06]
[07] 2000
[08] <html>
[09] <head>
[10] <title>Sign On: Abschluss</title>
[11] <meta http-equiv="content-type" content="text/html; charset=UTF-8">
[02] <head>
[13] <body>
[14] <p>Folgende Personenbindung wurde gelesen:</p>
[15] <pre style="background-color: silver;">&lt;?xml version="1.0" encoding="UTF-8"?>
[16] &lt;saml:Assertion AssertionID="szr.bmi.gv.at-AssertionID1086963510976445" \
[17] ...
[18] /dsig:Signature>&lt;/saml:Assertion></pre><p>Folgende Signatur ist eingelangt:</p>
[19] <pre style="background-color: silver;">&lt;?xml version="1.0" encoding="UTF-8"?>
[20] &lt;dsig:Signature Id="signature-28122004132954494" \
[21] ...
[22] &lt;/dsig:Signature></pre></body>
[23] </html>
```

Ähnlich wie in den Beispielen zuvor sendet die *Applikation* zum Abschluß ein HTML-Dokument, das von der *Bürgerkarten-Umgebung* in identer Form an den Browser des *Bürgers* weiter zu übermitteln ist (vgl. Zeile 8 - 23).

### *HTTP Response der Bürgerkarten-Umgebung an den Browser*

Schließlich sehen Sie unterbei noch die HTTP Response der *Bürgerkarten-Umgebung* an den Browser des *Bürgers.* Bitte beachten Sie, dass sie aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen \ am Ende einer Zeile) wurde.

```
[01] HTTP/1.1 200 OK
[02] Content-Type: text/html
[03] Transfer-Encoding: chunked
[04] Date: Tue, 28 Dec 2004 12:30:18 GMT
[05] Server: Apache Coyote/1.0
[06]
[07] 2000
[08] <html>
[09] <head>
[10] <title>Sign On: Abschluss</title>
[11] <meta http-equiv="content-type" content="text/html; charset=UTF-8">
[02] <head>
[13] <body>
[14] <p>Folgende Personenbindung wurde gelesen:</p>
[15] <pre style="background-color: silver;">&lt;?xml version="1.0" encoding="UTF-8"?>
[16] &lt;saml:Assertion AssertionID="szr.bmi.gv.at-AssertionID1086963510976445" \
[17] ...
[18] /dsig:Signature>&lt;/saml:Assertion></pre><p>Folgende Signatur ist eingelangt:</p>
[19] <pre style="background-color: silver;">&lt;?xml version="1.0" encoding="UTF-8"?>
[20] &lt;dsig:Signature Id="signature-28122004132954494" \
[21] ...
[22] &lt;/dsig:Signature></pre></body>
[23] </html>
```

### *Downloads zu diesem Beispiel*

- examples/bindings/signon/Request.html
- examples/bindings/signon/Browser.HttpRequest.txt
- examples/bindings/signon/DataURL.2.HttpRequest.txt
- examples/bindings/signon/DataURL.2.HttpResponse.txt
- examples/bindings/signon/Browser.HttpResponse.txt
- examples/bindings/signon/SignOn.java

### *Weiterführende Informationen*

- Abschnitt 3, „HTTP-Bindung“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer

#### 3.2.3.2. Signieren eines Antrags mit Beilagen

Dieses Beispiel präsentiert eine Lösung für das oft benötigte Szenario, dass vom *Bürger* Beilagen zur *Applikation* hochgeladen und anschließend zusammen mit einem ausgefüllten Antrag signiert werden sollen.

Nachdem Beilagen (z.B. CAD-Zeichnungen, PDF-Dokumente, etc.) meist in Formaten vorliegen, die nicht mit Hilfe der sicheren Signatur signiert werden können, empfielt sich der alternative Weg, die Beilagen nicht direkt zu signieren, sondern lediglich einen darüber berechneten Hashwert.

Es stellt sich dann allerdings die Frage, wer diesen Hashwert berechnen soll. Eine erste Idee wäre es, diese Berechnung von der *Applikation* durchführen zu lassen. Das führt aber möglicherweise zu einem Akzeptanzproblem beim *Bürger*, da dieser dann nicht weiß, ob der von der *Applikation* berechnete und zur Signatur vorgeschlagene Hashwert tatsächlich zu der vom *Bürger* hochgeladenen Beilage passt.

Deshalb ist es sinnvoller, die Berechnung des Hashwertes von der *Bürgerkarten-Umgebung* durchführen zu lassen, und zwar mit Hilfe des Befehls *CreateHash*.

Der grundsätzliche Ablauf entspricht wiederum jenem aus Abschnitt 3.2.3.1, „Personenbindung übermitteln und Dokument signieren (Sign On)“:

1. Der Browser des *Bürgers* sendet mittels eines HTML-Formulars den XML-Befehl *CreateHash* für die Berechnung des Hashwertes über eine Beilagean die *Bürgerkarten-Umgebung*. Die Beilage selbst wird als *Weitergabe-Parameter* zusammen mit dem XML-Befehl übermittelt.
2. Die *Bürgerkarten-Umgebung* verständigt den *Bürger* davon, dass ein Hashwert berechnet werden soll. Der *Bürger* hat die Möglichkeit, sich das Dokument, über welches der Hashwert berechnet werden soll, anzeigen oder abspeichern zu lassen. Weiters hält die *Bürgerkarten-Umgebung* den berechneten Hashwert für eine spätere Kontrolle durch den *Bürger* vor. Gibt der *Bürger* sein OK, sendet sie die XML-Befehlsantwort für *CreateHash* an die hinter der `DataURL` stehende *Applikation*.
3. Die *Applikation* liest die XML-Befehlsantwort, in der der berechnete Hashwert enthalten ist, sowie die Beilage selbst, die als *Weitergabe-Parameter* zusammen mit der XML-Befehlsantwort von der *Bürgerkarten-Umgebung* übermittelt wurde. Die *Applikation* antwortet danach mit einem neuen XML-Befehl *CreateXMLSignature* an die *Bürgerkarten-Umgebung*. Teil der mit diesem Befehl zu signierenden Daten ist der Hashwert über die hochgeladene Beilage. Wenn nun der *Bürger* das zu signierende Dokument mit dem darin enthaltenen Hashwert in der Anzeigekomponente der *Bürgerkarten-Umgebung* betrachtet, kann er diesen Hashwert durch eine Abschnitt 3.5, „Hashwert-Berechnung“ in Die österreichische Bürgerkarte - Anforderungen an die Benutzer-Schnittstelle in der *Benutzer-Schnittstelle* der *Bürgerkarten-Umgebung* mit dem zuvor durch die *Bürgerkarten-Umgebung* berechneten Hashwert vergleichen. Stimmen die Hashwerte überein, hat er die Gewissheit, dass er den korrekten Hashwert signiert.
4. Die *Bürgerkarten-Umgebung* sendet die XML-Befehlsantwort für *CreateXMLSignature* an die hinter der `DataURL` stehende *Applikation*.
5. Die *Applikation* antwortet mit einem HTML-Dokument.
6. Die *Bürgerkarten-Umgebung* übermittelt dieses HTML-Dokument an den Browser des *Bürgers*.

##### *HTML-Formular*

Das folgende HTML-Formular wurde von der Applikation dynamisch generiert und kann unter der Voraussetzung der installierten ??? von der URL `http://localhost:8080/SL12Tutorial/SignAttachments` geladen werden. Bitte beachten Sie, dass es aus Gründen der besseren Lesbarkeit formatiert und umgebrochen (gekennzeichnet durch das Zeichen \ am Ende einer Zeile) wurde.

```
[01] <?xml version='1.0' encoding='UTF-8'?>
[02] <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/\
[03] xhtml1/DTD/xhtml1-strict.dtd">
[04] <html xmlns='http://www.w3.org/1999/xhtml'>
[05]    <head>
[06]            <title>Signieren von Beilagen: Start</title>
[07]    </head>
[08]    <body>
[09]            <p>Bitte füllen Sie das folgende Formular 0815 aus:</p>
[10]            <form style='background-color: silver;'
[11]              action='http://127.0.0.1:13495/http-security-layer-request' method='post'
[12]              enctype='multipart/form-data' name='form1' id='form1'>
[13]              <p><code>Vorname: </code><input name='Vorname_' type='text' id='vorname_'/><br/>
[14]              <code>Nachname:</code><input name='Nachname_' type='text' id='nachname_'/></p>
[15]              <p><code>Beilage:</code><input type='file' name='Beilage_'/></p>
[16]              <p><input type='submit' value='Weiter ...'/></p>
[17]              <input type='hidden' name='XMLRequest'
[18]                value='&lt;?xml version="1.0" encoding="UTF-8"?>&lt;sl:CreateHashReques\
[19] t xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">  &lt;sl:\
[20] HashInfo RespondHashData="false">    &lt;sl:HashData>      &lt;sl:MetaInfo>    \
[21]     &lt;sl:MimeType>application/octet-stream&lt;/sl:MimeType>      &lt;/sl:Meta\
[22] Info>      &lt;sl:Content Reference="formdata:Beilage_"/>    &lt;/sl:HashData> \
[23]    &lt;sl:HashAlgorithm>http://www.w3.org/2000/09/xmldsig#sha1&lt;/sl:HashAlgor\
[24] ithm>    &lt;sl:FriendlyName>Beilage zum Formular 0815&lt;/sl:FriendlyName>  &l\
[25] t;/sl:HashInfo>&lt;/sl:CreateHashRequest>'/>
[26]      <input name='DataURL' type='hidden'
[27]        value='http://localhost:18080/SL12Tutorial/SignAttachments;jsessionid=2B\
[28] AF01A9069F6AF073A25B4D9B8E803E?use=sign'/>     </form>
[29]            <p>  </p>
[30]    </body>
[31] </html>
```

In den Zeilen 13 und 14 sind die beiden *Weitergabe-Parameter* `Vorname_` und `Nachname_` kodiert. Sie repräsentieren den eigentlichen Antrag.

In der Zeile 15 ist der *Weitergabe-Parameter* `Beilage_` kodiert. An dessen Attribut `type="file"` erkennt man, dass damit eine Datei hochgeladen werden soll.

In den Zeilen 17 - 25 ist der XML-Befehl als *Formular-Parameter* `XMLRequest` kodiert. Bitte beachten Sie, wie in diesem Befehl in Zeile 22 das Attribut `sl:Content/@Reference` mit Hilfe der URL `Abschnitt 3.2.1.2, „Referenzieren von Formularfeldern“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer:Beilage_` auf den Inhalt des *Weitergabe-Paramters* referenziert, der die hochzuladende Beilage enthält.

##### *HTTP Request des Browsers an die Bürgerkarten-Umgebung*

Nachfolgend sehen Sie den HTTP Request, der aus dem Absenden des Formulars in obiger HTML-Seite resultiert. Bitte beachten Sie, dass er aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen \ am Ende einer Zeile) wurde.

```
[01] POST /http-security-layer-request HTTP/1.1
[02] Host: 127.0.0.1
[03] User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.5) Gecko/20041107 \
[04] Firefox/1.0
[05] Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;\
[06] q=0.8,image/png,*/*;q=0.5
[07] Accept-Language: de-at,de;q=0.7,en;q=0.3
[08] Accept-Encoding: gzip,deflate
[09] Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
```

```
[10] Keep-Alive: 300
[11] Connection: keep-alive
[12] Content-Type: multipart/form-data; boundary=----------------------------2330864292941
[13] Content-Length: 1800
[14]
[15] ----------------------------2330864292941
[16] Content-Disposition: form-data; name="Vorname_"
[17]
[18] Thassilo
[19] ----------------------------2330864292941
[20] Content-Disposition: form-data; name="Nachname_"
[21]
[22] Tester
[23] ----------------------------2330864292941
[24] Content-Disposition: form-data; name="Beilage_"; filename="Beilage.png"
[25] Content-Type: image/png
[26]
[27] ...
[28] ----------------------------2330864292941
[29] Content-Disposition: form-data; name="XMLRequest"
[30]
[31] <?xml version="1.0" encoding="UTF-8"?><sl:CreateHashRequest \
[32] xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">  \
[33] <sl:HashInfo RespondHashData="false">    <sl:HashData>      <sl:MetaInfo>        \
[34] <sl:MimeType>application/octet-stream</sl:MimeType>      </sl:MetaInfo>      \
[35] <sl:Content Reference="http://www.buergerkarte.at/konzept/securitylayer/\
[36] spezifikation/20040514/tutorial/examples/bindings/signattachments/Beilage.png"/>    \
[37] </sl:HashData>    <sl:HashAlgorithm>http://www.w3.org/2000/09/xmldsig#sha1\
[38] </sl:HashAlgorithm>    <sl:FriendlyName>Beilage zum Formular 0815</sl:FriendlyName>  \
[39] </sl:HashInfo></sl:CreateHashRequest>
[40] ----------------------------2330864292941
[41] Content-Disposition: form-data; name="DataURL"
[42]
[43] http://localhost:18080/SL12Tutorial/SignAttachments;jsessionid=2BAF01A9069F6AF0\
[44] 73A25B4D9B8E803E?use=sign
[45] ----------------------------2330864292941--
```

Nachdem im Formular ein Input-Element vom Typ *file* enthalten war, wählt der Browser `multipart/form-data` für die Kodierung der Formular-Elemente (vergleiche Zeile 12).

In der Nutzlast finden sich nacheinander die *Weitergabe-Parameter* `Vorname_` (Zeile 16 - 18), `Nachname_` (Zeile 20 - 22), `Beilage_` (ein Bild vom Typ *png*, Zeile 24 - 27), der *Formular-Parameter* `XMLRequest` (*CreateHash*, Zeile 29 - 39), sowie der *Formular-Parameter* `DataURL` (Zeile 41 - 44).

### *Erster HTTP Request der Bürgerkarten-Umgebung an DataURL*

Als Nächstes sehen Sie den ersten HTTP Request der *Bürgerkarten-Umgebung* an die hinter der `DataURL` stehende *Applikation*. Dieser enthält die u. a. die XML-Befehlsantwort für *CreateHash* sowie die oben besprochenen *Weitergabe-Parameter* `Vorname_`, `Nachname_` und `Beilage_`. Bitte beachten Sie, dass der HTTP Request aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) wurde.

```
[01] POST /SL12Tutorial/SignAttachments;jsessionid=2BAF01A9069F6AF073A25B4D9B8E803E?use=sign HTTP/1.1
[02] Referer: localhost
[03] User-Agent: citizen-card-environment/1.2 trustDeskbasic/2.2.3-developer
[04] Content-Type: multipart/form-data; boundary=----------------------------0505ccc0949c35
[05] Host: 127.0.0.1
[06] Content-Length: 1469
[07] Connection: Keep-Alive
[08] Cache-Control: no-cache
[09]
[00] ----------------------------0505ccc0949c35
[01] Content-Disposition: form-data; name="ResponseType"
[02]
[03] HTTP-Security-Layer-RESPONSE
[04] ----------------------------0505ccc0949c35
[05] Content-Disposition: form-data; name="XMLResponse"
[06]
[07] <?xml version="1.0" encoding="UTF-8"?><sl:CreateHashResponse \
[08] xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#"><sl:HashInfo>\
[09] <sl:HashAlgorithm>http://www.w3.org/2000/09/xmldsig#sha1</sl:HashAlgorithm>\
[00] <sl:FriendlyName>Beilage zum Formular 0815</sl:FriendlyName>\
[01] <sl:HashValue>ijB9/RGrjhilvBXCB5YWGSVusoI=</sl:HashValue></sl:HashInfo>\
[02] </sl:CreateHashResponse>
[03] ----------------------------0505ccc0949c35
[04] Content-Disposition: form-data; name="Vorname_"
[05]
[06] Thassilo
[07] ----------------------------0505ccc0949c35
[08] Content-Disposition: form-data; name="Nachname_"
[09]
[00] Tester
[01] ----------------------------0505ccc0949c35
[02] Content-Disposition: form-data; name="Beilage_"; filename="Beilage.png"
[03] Content-Type: image/png
[04]
[05] ...
[06] ----------------------------0505ccc0949c35--
```

### *Erste HTTP Response von DataURL an die Bürgerkarten-Umgebung*

Im Folgenden finden Sie die erste HTTP Response der hinter der `DataURL` stehenden *Applikation* an die *Bürgerkarten-Umgebung*. Bitte beachten Sie, dass sie aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen `\` am Ende einer Zeile) wurde.

```
[01] HTTP/1.1 307 Temporary Redirect
[02] Location: http://localhost:8080/SL12Tutorial/SignAttachments;\
[03] jsessionid=2BAF01A9069F6AF073A25B4D9B8E803E?use=done
[04] Content-Type: text/xml
[05] Content-Length: 996
[06] Date: Tue, 28 Dec 2004 16:49:00 GMT
[07] Server: Apache Coyote/1.0
```

```
[08]
[09] <?xml version='1.0' encoding='UTF-8'?>
[10] <sl:CreateXMLSignatureRequest \
[11] xmlns:sl='http://www.buergerkarte.at/namespaces/securitylayer/1.2#'>
[12]   <sl:KeyboxIdentifier>SecureSignatureKeypair</sl:KeyboxIdentifier>
[13]   <sl:DataObjectInfo Structure='enveloping'>
[14]     <sl:DataObject>
[15]       <sl:XMLContent><html xmlns='http://www.w3.org/1999/xhtml'>
[16]    <head>
[17]            <title>Formular 0815</title>
[18]            <style type='text/css'>.silver{ background-color: silver;}</style>
[19]    </head>
[20]    <body>
[21]            <h3>Formular 0815</h3>
[22]            <p>Vorname: <code class='silver'>Thassilo</code></p>
[23]            <p>Nachname: <code class='silver'>Tester</code></p>
[24]            <h3>Beilage zu Formular 0815</h3>
[25]            <p>PrÃ¼fsumme (SHA-1, base64-kodiert): \
[26] <code class='silver'>ijB9/RGrjhilvBXCB5YWGSVusoI=</code></p>
[27]    </body>
[28] </html>
[29] </sl:XMLContent>
[30]     </sl:DataObject>
[31]     <sl:TransformsInfo>
[32]       <sl:FinalDataMetaInfo>
[33]         <sl:MimeType>text/html</sl:MimeType>
[34]       </sl:FinalDataMetaInfo>
[35]     </sl:TransformsInfo>
[36]   </sl:DataObjectInfo>
[37] </sl:CreateXMLSignatureRequest>
```

Die Nutzlast (vgl. Zeile 9 - 37) enthält den nächsten Befehl an die Bürgerkarten-Umgebung (*CreateXMLSignature*).

Die Zeilen 15 - 28 zeigen das vom *Bürger* zu signierende Dokument. Man erkennt, das darin auch der zuvor von der *Bürgerkarten-Umgebung* berechnete Hashwert (Zeile 25 - 26) enthalten ist.

### *Zweiter HTTP Request der Bürgerkarten-Umgebung an DataURL*

Im Weiteren sehen Sie den zweiten HTTP Request der *Bürgerkarten-Umgebung* an die hinter der `DataURL` stehende *Applikation*. Dieser enthält die u. a. die XML-Befehlsantwort für *CreateXMLSignature*. Bitte beachten Sie, dass der HTTP Request aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen \ am Ende einer Zeile) und gekürzt (gekennzeichnet durch . . . ) wurde.

```
[01] POST /SL12Tutorial/SignAttachments;jsessionid=2BAF01A9069F6AF073A25B4D9B8E803E?use=done HTTP/1.1
[02] Referer: localhost
[03] User-Agent: citizen-card-environment/1.2 trustDeskbasic/2.2.3-developer
[04] Content-Type: multipart/form-data; boundary=----------------------------04a36030005bad
[05] Host: localhost:18080
[06] Content-Length: 6602
[07] Connection: Keep-Alive
[08] Cache-Control: no-cache
[09]
[10] ----------------------------04a36030005bad
[11] Content-Disposition: form-data; name="ResponseType"
[12]
[13] HTTP-Security-Layer-RESPONSE
[14] ----------------------------04a36030005bad
[15] Content-Disposition: form-data; name="XMLResponse"
[16]
[17] <?xml version="1.0" encoding="UTF-8"?><sl11:CreateXMLSignatureResponse \
[18] xmlns:sl11="http://www.buergerkarte.at/namespaces/securitylayer/20020831#">\
[19] ...
[20] </dsig:Signature></sl11:CreateXMLSignatureResponse>
[21] ----------------------------04a36030005bad
[22] Content-Disposition: form-data; name="Vorname_"
[23]
[24] Thassilo
[25] ----------------------------04a36030005bad
[26] Content-Disposition: form-data; name="Nachname_"
[27]
[28] Tester
[29] ----------------------------04a36030005bad
[30] Content-Disposition: form-data; name="Beilage_"; filename="Beilage.png"
[31] Content-Type: image/png
[32]
[33] ...
[34] ----------------------------04a36030005bad--
```

### *Zweite HTTP Response von DataURL an die Bürgerkarten-Umgebung*

Im Folgenden finden Sie die zweite HTTP Response der hinter der `DataURL` stehenden *Applikation* an die *Bürgerkarten-Umgebung*. Bitte beachten Sie, dass sie aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen \ am Ende einer Zeile) und gekürzt (gekennzeichnet durch . . . ) wurde.

```
[01] HTTP/1.1 200 OK
[02] Content-Type: text/html
[03] Content-Length: 5919
[04] Date: Tue, 28 Dec 2004 16:49:51 GMT
[05] Server: Apache Coyote/1.0
[06]
[07] <html>
[08] <head>
[09] <title>Signieren von Beilagen: Abschluss</title>
[10] <meta http-equiv="content-type" content="text/html; charset=UTF-8">
[11] <head>
[12] <body>
[13] <p>Folgende Signatur ist eingelangt:</p>
[14] <pre style="background-color: silver;">&lt;?xml version="1.0" encoding="UTF-8"?>
[15] &lt;dsig:Signature Id="signature-28122004174904293" \
```

```
[16] xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">\
[17] ...
[18] &lt;/dsig:Signature></pre></body>
[19] </html>
```

Ähnlich wie in den Beispielen zuvor sendet die *Applikation* zum Abschluß ein HTML-Dokument, das von der *Bürgerkarten-Umgebung* in identer Form an den Browser des *Bürgers* weiter zu übermitteln ist (vgl. Zeile 8 - 23).

#### *HTTP Response der Bürgerkarten-Umgebung an den Browser*

Schließlich sehen Sie unterbei noch die HTTP Response der *Bürgerkarten-Umgebung* an den Browser des *Bürgers.* Bitte beachten Sie, dass sie aus Gründen der besseren Lesbarkeit umgebrochen (gekennzeichnet durch das Zeichen \ am Ende einer Zeile) wurde.

```
[01] HTTP/1.1 200 OK
[02] Content-Type: text/html
[03] Content-Length: 5919
[04] Date: Tue, 28 Dec 2004 16:49:51 GMT
[05] Server: Apache Coyote/1.0
[06]
[07] <html>
[08] <head>
[09] <title>Signieren von Beilagen: Abschluss</title>
[10] <meta http-equiv="content-type" content="text/html; charset=UTF-8">
[11] <head>
[12] <body>
[13] <p>Folgende Signatur ist eingelangt:</p>
[14] <pre style="background-color: silver;">&lt;?xml version="1.0" encoding="UTF-8"?>
[15] &lt;dsig:Signature Id="signature-28122004174904293" \
[16] xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">\
[17] ...
[18] &lt;/dsig:Signature></pre></body>
[19] </html>
```

#### *Downloads zu diesem Beispiel*

- examples/bindings/signattachments/Request.html
- examples/bindings/signattachments/Beilage.png
- examples/bindings/signattachments/Browser.HttpRequest.txt
- examples/bindings/signattachments/DataURL.1.HttpRequest.txt
- examples/bindings/signattachments/DataURL.1.HttpResponse.txt
- examples/bindings/signattachments/DataURL.2.HttpRequest.txt
- examples/bindings/signattachments/DataURL.2.HttpResponse.txt
- examples/bindings/signattachments/Browser.HttpResponse.txt
- examples/bindings/signattachments/SignAttachments.java

#### *Weiterführende Informationen*

- Abschnitt 3, „HTTP-Bindung“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer
- Abschnitt 6.1, „Hashwert-Berechnung“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer
- Abschnitt 3.5, „Hashwert-Berechnung“ in Die österreichische Bürgerkarte - Anforderungen an die Benutzer-Schnittstelle
- Abschnitt 3.2.1.2, „Referenzieren von Formularfeldern“ in Die österreichische Bürgerkarte - Transportprotokolle Security-Layer

#### *Downloads zu diesem Beispiel*

- examples/viewerformat/Simple.xhtml

#### *Weiterführende Informationen*

- Abschnitt 9.1, „Formate für die Anzeige der Bürgerkarten-Umgebung“ in Die österreichische Bürgerkarte - Minimale Umsetzung des Security-Layers
- Standard-Anzeigeformat

#### *Downloads zu diesem Beispiel*

- examples/viewerformat/Extended.css.xhtml
- examples/viewerformat/Extended.xhtml

#### *Weiterführende Informationen*

- Standard-Anzeigeformat

## 4. Standard-Anzeigeformat SLXHTML

Ein wichtiger Aspekt beim Erstellen von *Applikationen*, die mit den Signaturbefehlen der Bürgerkarte arbeiten, ist die Auswahl eines Abschnitt 9.1, „Formate für die Anzeige der Bürgerkarten-Umgebung“ in Die österreichische Bürgerkarte - Minimale Umsetzung des Security-Layers für die zu signierenden Daten, damit diese in der Anzeigekomponente der *Bürgerkarten-Umgebung* dargestellt und in weiterer Folge signiert werden können.

Zunächst einmal kann eine *Bürgerkarten-Umgebung* Abschnitt 9.1.1, „Text“ in Die österreichische Bürgerkarte - Minimale Umsetzung des Security-Layers (u. a. *Mime Types* `text/plain`, `text/xml` und `application/xml`) anzeigen und signieren. Nachdem dieses Format weitgehend selbsterklärend sind, werden sie hier nicht weiter erläutert.

Als weiteres Format akzeptiert eine *Bürgerkarten-Umgebung* Dokumente im Format SLXHTML (Security-Layer XHTML, Mime Type `application/xhtml+xml`), dem Standard-Anzeigeformat der Bürgerkarte. Es handelt sich dabei um eine Variante von XHTML 1.0; gegenüber dem Ausgangsformat werden gewisse Einschränkungen gemacht, damit sich das Format für eine Anzeige von zu signierenden Daten in der *Bürgerkarten-Umgebung* eignet:

- Verbot von Inhalten, mit denen ein Dokument zu unterschiedlichen Betrachtungszeitpunkten unterschiedlich dargestellt werden kann; z.B. Script-

Elemente;

- Verbot von Formatierungen, die in Anzeigekomponenten zu verdeckten Dokumentteilen führen können; z.B. fixe Spaltenbereiten in Tabellen oder absolute Positionierung mittels Cascading Style Sheets, Level 2 (CSS 2).

Weiters wird in SLXHTML eine strikte Trennung von Strukur (mittels HTML-Tags) und Formatierung (mittels CSS 2) gefordert. So ist beispielsweise die Formatierung mittels veralteter HTML-Elemente wie etwa `font`, `u` oder `center` ebenso verboten wie die Formatierung von Tabellen mit Hilfe von Attributen wie `align` oder `valign` in den Tabellenelementen `table`, `td` und `tr`. Stattdessen sind passende Konstrukte aus CSS 2 zu verwenden.

Detailliertere Informationen finden Sie in den folgenden Unterabschnitten.

## 4.1. Ein erstes Beispiel

Im Folgenden finden Sie ein erstes minimales Dokument im Format SLXHTML. In den untenstehenden Erläuterungen finden Sie wichtige Hinweise zu den Grundregeln des Dokumentaufbaus.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <html xmlns="http://www.w3.org/1999/xhtml">
[03]   <head>
[04]     <title>Ein einfaches SLXHTML-Dokument</title>
[05]     <style type="text/css">p { color: red; }</style>
[06]   </head>
[07]   <body>
[08]     <p>Ich bin ein einfacher Text in rot.</p>
[09]   </body>
[10] </html>
```

Grundsätzlich handelt es sich bei SLXHTML um einen XML-Dialekt. Syntaktisch muss das Dokument also wohlgeformtes XML sein. Öffnende Tags ohne korrespondierende schließende Tags (etwa `<p>` oder `<br>`) sind also nicht gestattet. Weiters sollte das Dokument immer mit einer XML-Deklaration beginnen, die mit dem Attribut `encoding` den Zeichensatz für das Dokument festlegt (vergleiche Zeile 1).

Alle HTML-Sprachelemente müssen sich im Namenraum `http://www.w3.org/1999/xhtml` befinden. Sinnvollerweise wird man dazu den Default-Namenraum verwenden (vergleiche Zeile 2).

Das Element `head` (Zeile 3 - 6) muss angegeben werden. Sein Inhalt muss aus genau einem Element `title` (Zeile 4), gefolgt von genau einem Element `style` (Zeile 5) bestehen. Das Attribut `style/@type` ist auf den Wert `text/css` fixiert, es dürfen also nur CSS-Formatierungsinformationen enthalten sein. Werden keine Formatierungsinformationen angegeben, ist ein leeres `style`-Element anzugeben.

## 4.2. Ein umfangreiches Beispiel

Nachfolgend finden Sie ein umfangreiches Dokument im Format SLXHTML. Es demonstriert einerseits den zur Verfügung stehenden Satz an HTML-Sprachelementen, sowie andererseits die Möglichkeiten der Formatierung mit Konstrukten aus CSS 2.

```
[001] <?xml version="1.0" encoding="UTF-8"?>
[002] <html xmlns="http://www.w3.org/1999/xhtml">
[003]   <head>
[004]     <title>Ein umfangreiches SLXHTML-Dokument</title>
[005]     <style type="text/css">
```

Das Dokument beginnt wie schon zuvor das einfache Beispiel. Es folgen ab Zeile 5 die CSS 2 Formatierungsanweisungen.

```
[006]     body
[007]     {
[008]       font-family: serif;
[009]       font-size: medium;
[010]     }
```

Für `body` werden die grundsätzlichen Eigenschaften von Text im Dokument festgelegt. Es wird eine Schriftart mit Serifen gewählt, die Schriftgröße wird mit `medium` festgelegt.

```
[011]     h1
[012]     {
[013]       font-family: sans-serif;
[014]       font-size: 147%;
[015]       color: #000099;
[016]       margin-top: 0.5em;
[017]       margin-bottom: 0.5em;
[018]     }
[019]     h2
[020]     {
[021]       font-family: sans-serif;
[022]       font-size: 136%;
[023]       color: #000099;
[024]       margin-top: 0.5em;
[025]       margin-bottom: 0.5em;
[026]     }
[027]     h3
[028]     {
[029]       font-family: sans-serif;
[030]       font-size: 126%;
[031]       color: #000099;
[032]       margin-top: 0.5em;
[033]       margin-bottom: 0.5em;
[034]     }
[035]     h4
[036]     {
[037]       font-family: sans-serif;
[038]       font-size: 117%;
[039]       color: #000099;
[040]       margin-top: 0.5em;
[041]       margin-bottom: 0.5em;
[042]     }
[043]     h5
[044]     {
[045]       font-family: sans-serif;
[046]       font-size: 108%;
[047]       color: #000099;
[048]       margin-top: 0.5em;
```

```
[049]       margin-bottom: 0.5em;
[050]     }
[051]     h6
[052]     {
[053]       font-family: sans-serif;
[054]       font-size: 100%;
[055]       color: #000099;
[056]       margin-top: 0.5em;
[057]       margin-bottom: 0.5em;
[058]     }
```

Hier wird das Aussehen aller sechs Überschriften-Ebenen festgelegt. Für jede Überschrift wird eine seriflose Schriftart ausgewählt (z.B. Zeile 53) , die Schriftgröße entsprechend abgestuft (z.B. Zeile 54 und 56), die Abstände nach oben und unten mit einem Wert relativ zur Schriftgröße festgelegt (jeweils `0.5em`, also die halbe Breite des Zeichens `m`, vgl. z.B. Zeile 56 - 57), sowie als Schriftfarbe einheitlich ein Blauwert gewählt (z.B. Zeile 55) .

```
[059]     p
[060]     {
[061]       text-align: justify;
[062]     }
```

Für einen Absatz wird Blocksatz festgelegt.

```
[063]     code
[064]     {
[065]       color: blue;
[066]       font-family: monospace;
[067]       font-size: 90%;
[068]     }
```

Für Text innerhalb des Elements `code` wird eine Schriftart mit fixer Zeichenbreite und eine blaue Schriftart gewählt. Außerdem wird die Schriftgröße im Verhältnis zum umgebenen Text etwas reduziert.

```
[069]     em
[070]     {
[071]       font-style: italic;
[072]       color: #990000;
[073]     }
```

Als betont ausgezeichneter Text wird schräg und mit einer rotbraunen Schriftfarbe gesetzt.

```
[074]     strong
[075]     {
[076]       font-weight: bold;
[077]       color: #006633;
[078]     }
```

Als stark betont ausgezeichneter Text wird fett und in einer moosgrünen Schriftfarbe gesetzt.

```
[089]     pre
[080]     {
[081]       background-color: #CCCCFF;
[082]     }
```

Vorformatierter Text wird vor einen türkisen Hintergrund gesetzt.

```
[083]     blockquote
[084]     {
[085]       font-style: italic;
[086]     }
```

Lange Zitate werden schräg gesetzt.

```
[087]     table.simple
[088]     {
[089]       background-color: #CCFFCC;
[090]       border-style: dotted;
[091]       border-width: 1px;
[092]       border-color: black;
[093]       margin: 10px;
[094]     }
[095]     caption.simple
[096]     {
[097]       color: #0066FF;
[098]     }
```

Das Dokument weist zwei Tabellen auf. Die obigen Formatanweisungen gelten für die erste Tabelle, da die dortigen Elemente `table` und `caption` ein Attribut `class="simple"` aufweisen (vergleiche Zeile 193 und 194).

Die Tabelle selbst bekommt eine türkise Hintergrundfarbe sowie einen punktierten, schwarzen Rahmen mit einer Dicke von einem Pixel. Der Randabstand wird zu allen Seiten mit 10 Pixel festgelegt.

Der Tabellentitel wird in einer blaufärbigen Schrift gesetzt.

```
[099]     table.extended
[100]     {
[101]       background-color: #FFCC66;
[102]       border-style: solid;
[103]       border-width: 2px;
[104]       border-color: black;
[105]       margin: 20px;
[106]     }
```

Das Dokument weist zwei Tabellen auf. Die obigen Formatanweisungen gelten für die zweite Tabelle, da das dortige Element `table` ein Attribut `class="extended"` aufweist (vergleiche Zeile 214).

Die Tabelle bekommt eine hellorgange Hintergrundfarbe sowie einen durchgehenden, schwarzen Rahmen mit einer Dicke von zwei Pixel. Der Randabstand wird zu allen Seiten mit 20 Pixel festgelegt.

```
[107]     thead, tfoot
[108]     {
[109]       background-color: #FF9900;
[110]       padding: 5px;
```

```
[111]     }
```

Für die Kopf- und Fußzeile einer Tabelle wird ein Innenabstand von 5 Pixel sowie eine dunkelorange Hintergrundfarbe gewählt.

```
[112]     td, th
[113]     {
[114]       padding: 5px;
[115]       border-style: solid;
[116]       border-width: 1px;
[117]       border-color: black;
[118]       }
[119]     td#missing1, td#missing2
[120]     {
[121]       background-color: red;
[122]     }
```

Für Tabellenzellen (`td`, `th`) wird ein Innenabstand von 5 Pixel sowie eine durchgehender, schwarzer Rahmen mit einer Dicke von einem Pixel festgelegt.

Für jene Tabellenzellen, die ein ID-Attribut mit dem Wert `missing1` bzw. `missing2` aufweisen, wird eine rote Hintergrundfarbe gewählt.

```
[123]     img
[124]     {
[125]       margin: 20px;
[126]       border-style: solid;
[127]       border-color: black;
[128]       border-width: 1px;
[129]     }
```

Für Bilder wird ein einheitlicher Abstand von 20 Pixel zu allen Seiten festgelegt. Weiters soll um Bilder ein durchgehender, schwarzer Rahmen mit einer Dicke von einem Pixel gezeichnet werden.

```
[130]     p.img
[131]     {
[132]       text-align: center;
[133]     }
```

Bilder sollen im Dokument zentriert positioniert werden. Das kann erreicht werden, indem man das Bild innerhalb eines eigenen Absatzes positioniert, und für diesen Absatz die CSS-Eigenschaft text-align mit dem Wert center belegt (vergleiche auch Zeile 250).

```
[134]     ol.level2
[135]     {
[136]       list-style-type: lower-alpha;
[137]     }
[138]       ul.level2
[149]     {
[140]       list-style-type: square;
[141]     }
```

Für die zweite Ebene einer numerierten Liste sollen als Listenzeichen Kleinbuchstaben verwendet werden (vergleiche Zeile 259).

Für die zweite Ebene einer nichtnumerierten Liste soll als Listenzeichen ein Quadrat verwendet werden (vergleiche Zeile 271).

```
[142]     .area
[143]     {
[144]       background-color: #CCCCCC;
[145]       border-style: solid;
[146]       border-width: thin;
[147]       border-color: black;
[148]       margin-bottom: 10pt;
[149]       margin-top: 10pt;
[150]       padding: 3px;
[151]     }
```

Jeder Abschnitt des Dokuments soll in einem eigenen, gerahmten Bereich erscheinen. Es wird dazu das HTML-Element `div` verwendet, welches ein Attribut `class` mit dem Wert `area` aufweist (vergleiche z.B. Zeile 163).

Der Bereich erhält eine graue Hintergrundfarbe sowie einen durchgehenden, dünnen Rahmen in schwarzer Farbe. Weiters werden für den Bereich Abstände nach oben und unten von jeweils 10 Punkten festgelegt. Schließlich bekommt der Bereich noch einen Innenabstand von 3 Pixeln.

```
[152]     .blockheading
[153]     {
[154]       color: #990000;
[155]     }
```

Die Überschriften innerhalb eines langen Zitats (Element `blockquote`) sollen in roter Farbe erscheinen (vergleiche Zeiel 186).

```
[156]     .highlight
[157]     {
[158]       background-color: yellow;
[159]     }
```

Eine eigene Klasse für hervorgehobenen Text (gelber Hintergrund) wird definiert. Eine Anwendung mit Hilfe des Elements `span` finden Sie in Zeile 174.

```
[163]     <div class="area">
[164]       <h1>Überschriften</h1>
[165]       <h2>Überschrift Ebene 2</h2>
[166]       <h3>Überschrift Ebene 3</h3>
[167]       <h4>Überschrift Ebene 4</h4>
[168]       <h5>Überschrift Ebene 5</h5>
[169]       <h6>Überschrift Ebene 6</h6>
[170]     </div>
```

Dieser und die folgenden Ausschnitte zeigen die innerhalb eines SLXHTML-Dokuments verfügbaren HTML-Elemente. Hier sehen Sie die 6 Überschriften-Ebenen.

```
[171]     <div class="area">
[172]       <h1>Absätze, Inline-Formate</h1>
[173]       <p>Ich bin ein Absatz. Leider ist mir nicht viel Text eingefallen.</p>
[174]       <p>Ich bin auch ein Absatz.<br/>Mein <span class="highlight">zweiter Satz</span>
[175]         steht in einer eigenen Zeile.</p>
[176]       <p>Ich enthalte folgendes Zitat aus <cite>unbekannter Quelle</cite>:
[177]         <em>Schön war's.</em></p>
```

```
[178]       <p>Und ich enthalte Code: <code>int test = 4;</code></p>
[179]       <p>Ich möchte <em>betonen</em>, dass das nicht so gemeint war.</p>
[180]       <p>Ich enthalte als <strong>strong</strong> ausgezeichneten Text.</p>
[181]       <pre>I c h  bin  vorformatierter  Text.</pre>
[182]     </div>
```

Es folgen der Gebrauch des generischen *inline*-Elements `span`, der *inline*-Elemente `cite`, `em`, `code` und `strong`, des generischen *block-level*-Elements `div`, sowie der *block-level*-Elemente `p` und `pre`.

```
[183]     <div class="area">
[184]       <h1>Blockweises Zitieren</h1>
[185]       <blockquote>
[186]         <h3 class="blockheading">Überschrift Ebene 3 innerhalb von
[187]         <code>blockquote</code></h3>
[188]         <p>Absatz innerhalb von <code>blockquote</code></p>
[189]       </blockquote>
[190]     </div>
```

Für lange Zitate steht das *block-level*-Element `blockquote` zur Verfügung.

```
[191]     <div class="area">
[192]       <h1>Einfache Tabelle</h1>
[193]       <table class="simple">
[194]         <caption class="simple">Einfache Tabelle</caption>
[195]         <tr>
[196]           <th>Vorname</th>
[197]           <th>Nachname</th>
[198]           <th>Geburtsdatum</th>
[199]         </tr>
[200]         <tr>
[201]           <td>Homer</td>
[202]           <td>Simpson</td>
[203]           <td>05 .10. 1955</td>
[204]         </tr>
[205]         <tr>
[206]           <td>Bart</td>
[207]           <td>Simpson</td>
[208]           <td id="missing1">01 .04.</td>
[209]         </tr>
[210]       </table>
[211]     </div>
```

Oben sehen Sie eine Tabelle im einfachen Tabellenformat (ohne die Elemente `thead`, `tbody`, `tfoot`).

```
[212]     <div class="area">
[213]       <h1>Erweiterte Tabelle</h1>
[214]       <table class="extended">
[215]         <caption>Erweiterte Tabelle</caption>
[216]         <thead>
[217]         <tr>
[218]           <th colspan="2">Namen</th>
[219]           <th>Sonstiges</th>
[220]         </tr>
[220]         <tr>
[222]           <th>Vorname</th>
[223]           <th>Nachname</th>
[224]           <th>Geburtsdatum</th>
[225]         </tr>
[226]         </thead>
[227]         <tfoot>
[228]           <tr>
[229]             <th>Vorname</th>
[230]             <th>Nachname</th>
[231]             <th>Geburtsdatum</th>
[232]           </tr>
[233]         </tfoot>
[234]         <tbody>
[235]           <tr>
[236]             <td>Homer</td>
[237]             <td>Simpson</td>
[238]             <td>05 .10. 1955</td>
[239]           </tr>
[240]           <tr>
[241]             <td>Bart</td>
[242]             <td>Simpson</td>
[243]             <td id="missing2">01 .04.</td>
[244]           </tr>
[245]         </tbody>
[246]       </table>
[247]     </div>
```

Und hier sehen Sie eine Tabelle im erweiterten Tabellenformat: Die `table` gliedert sich zunächst in `thead`, `tfoot` und `tbody`; erst darin erscheinen die Tabellenzeilen (`tr`) mit den Tabellenzellen (`th` bzw. `td`).

```
[248]     <div class="area">
[249]       <h1>Bilder</h1>
[250]       <p class="img">
[251]         <img alt="Logo A-Sit" src="LogoAsit.gif"></img>
[252]       </p>
[253]     </div>
```

Im obigen Bereich wird ein Bild integriert. Für das Element `img` muss jedenfalls das Attribut `alt` mit einem alternativen Text angegeben werden.

```
[254]     <div class="area">
[255]             <h1>Numerierte Liste</h1>
[256]         <ol>
[257]           <li>Erster Eintrag</li>
[258]         <li>Zweiter Eintrag
[259]           <ol class="level2">
[260]             <li>Erster Subeintrag</li>
```

```
[261]             <li>Zweiter Subeitrag</li>
[262]           </ol>
[263]         </li>
[264]       </ol>
[265]     </div>
```

Es folgt ein Beispiel einer numerierten Liste.

```
[266]     <div class="area">
[267]       <h1>Nicht numerierte Liste</h1>
[268]       <ul>
[269]         <li>Erster Eintrag</li>
[270]         <li>Zweiter Eintrag
[271]           <ul class="level2">
[272]             <li>Erster Subeintrag</li>
[273]             <li>Zweiter Subeitrag</li>
[274]           </ul>
[275]         </li>
[276]       </ul>
[277]     </div>
[278]   </body>
[279] </html>
```

Abschließend sehen Sie ein Beispiel für eine nichtnumerierte Liste.

## 4.3. Signieren von SLXHTML-Dokumenten mit Bildern

Dokumente im Standard-Anzeigeformat SLXHTML, die Bilder referenzieren (Element `img`), können von der *Bürgerkarten-Umgebung* signiert werden; dies jedoch nur dann, wenn eine XML-Signatur erstellt wird; für eine CMS-Signatur besteht diese Möglichkeit nicht.

Kann die *Bürgerkarten-Umgebung* ein referenziertes Bild auflösen, muss sie das Bild in der Anzeigekomponente als Teil des SLXHTML-Dokuments darstellen und bei der Berechnung der Signatur für das Bild ein eigenes Element `dsig:Reference` mit einem speziell gewählten Attribut `dsig:Reference/@Type`, sowie einem Attribut `dsig:Reference/@URI`, dessen Wert dem Attribut `img/@src` der Bildreferenz entspricht, Abschnitt 4, „Bilder im Standard-Anzeigeformat“ in Die österreichische Bürgerkarte - Standard-Anzeigeformat.

Kann die *Bürgerkarten-Umgebung* ein referenziertes Bild hingegen nicht auflösen, muss sie in der Anzeigekomponente statt des Bildes den alternativen Text zum Bild (`img/@alt`) als Teil des SLXHTML-Dokuments darstellen, und darf bei der Berechnung kein eigenes Element `dsig:Reference` erstellen.

Enthält das Attribut `img/src` der Bildreferenz eine URL, die von der *Bürgerkarten-Umgebung* nicht aufgelöst werden kann, kann die *Applikation* die Bilddaten im `sl:CreateXMLSignatureRequest` aentweder direkt (als `sl:BinaryContent`) oder indirekt (als `sl:LocRefConent`) als Ergänzungsobjekt (`sl:Supplement`) zu den zu signierenden Daten beisteuern (vergleiche Abschnitt 2.1.2.5, „Ergänzungsobjekte, Signaturmanifest“).

Jedenfalls unterstützt werden von einer *Bürgerkarten-Umgebung* Bilder im Format *gif* und *jpeg*, wobei jeweils einige Abschnitt 2.1.7, „Image Module“ in Die österreichische Bürgerkarte - Standard-Anzeigeformat einzuhalten sind.

### 4.3.1. 4.3.1 Ein erstes Beispiel

Im Folgenden soll für ein SLXHTML-Dokument, das eine von der *Bürgerkarten-Umgebung* auflösbare Bildreferenz enthält, eine XML-Signatur erstellt werden.

#### 4.3.1.1. Anfrage

Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert und umgebrochen (erkennbar durch das Zeichen `\` am Zeilenende) wurde.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:CreateXMLSignatureRequest
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]   <sl:KeyboxIdentifier>SecureSignatureKeypair</sl:KeyboxIdentifier>
[05]   <sl:DataObjectInfo Structure="enveloping">
[06]     <sl:DataObject>
[07]       <sl:XMLContent><html xmlns="http://www.w3.org/1999/xhtml">
[08]   <head>
[09]     <title>Ein einfaches SLXHTML-Dokument mit enthaltenem Bild</title>
[10]     <style type="text/css"/>
[11]   </head>
[12]   <body>
[13]     <p>Das nachfolgende Bild gibt einen Überblick zum Modell Bürgerkarte.</p>
[14]     <p>
[15]       <img src="http://www.buergerkarte.at/konzept/securitylayer/spezifikation/\
[16] 20040514/tutorial/examples/viewerformat/ModellBuergerkarte.gif"
[17]         alt="Modell Bürgerkarte"/>
[18]     </p>
[19]   </body>
[20] </html>
[21] </sl:XMLContent>
[22]     </sl:DataObject>
[23]     <sl:TransformsInfo>
[24]       <sl:FinalDataMetaInfo>
[25]         <sl:MimeType>application/xhtml+xml</sl:MimeType>
[26]       </sl:FinalDataMetaInfo>
[27]     </sl:TransformsInfo>
[28]   </sl:DataObjectInfo>
[29] </sl:CreateXMLSignatureRequest>
```

Das zu signierende SLXHTML-Dokument ist in den Zeilen 7 - 21 angegeben. Man erkennt in den Zeilen 15 - 17 die Bildreferenz mit einer http-URL, die von der *Bürgerkarten-Umgebung* aufgelöst werden kann.

#### 4.3.1.2. Antwort

Die entsprechende Antwort der *Bürgerkarten-Umgebung* sollte in etwa wie folgt aussehen. Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert, gekürzt und umgebrochen wurde, was natürlich die elektronische Signatur bricht.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl11:CreateXMLSignatureResponse
[03]   xmlns:sl11="http://www.buergerkarte.at/namespaces/securitylayer/20020831#">
[04]   <dsig:Signature Id="signature-02012005143921145"
[05]     xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
```

```
[06]     <dsig:SignedInfo>
[07]       ...
[08]       <dsig:Reference Id="reference-0-02012005143921145" URI="#signed-data-0-02012005143921145">
[09]         <dsig:Transforms>
[10]           <dsig:Transform Algorithm="http://www.w3.org/2002/06/xmldsig-filter2">
[11]             <xpf:XPath Filter="intersect" xmlns:xpf="http://www.w3.org/2002/06/xmldsig-filter2">\
[12] //*[@Id='signed-data-0-02012005143921145']/node()</xpf:XPath>
[13]           </dsig:Transform>
[14]         </dsig:Transforms>
[15]         <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[16]         <dsig:DigestValue>1f3BLWEdDbsRxP3MgQuJtARuVAM=</dsig:DigestValue>
[17]       </dsig:Reference>
[18]       ...
[19]       <dsig:Reference Id="reference-02012005143921145-addref-0"
[20]         Type="http://www.buergerkarte.at/specifications/Security-Layer/20031031?\
[21] name=SignedImage&InstanceDocRef=0"
[22]         URI="http://www.buergerkarte.at/konzept/securitylayer/spezifikation/20040514/\
[23] tutorial/examples/viewerformat/ModellBuergerkarte.gif">
[24]         <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[25]         <dsig:DigestValue>BgP4msC8vf30udU9BmuNW+IqdEI=</dsig:DigestValue>
[26]       </dsig:Reference>
[27]     </dsig:SignedInfo>
[28]     ...
[29]     <dsig:Object Id="signed-data-0-02012005143921145">
[30]       <html version="-//www.buergerkarte.at//DOCUMENT SLXHTML 1.2//DE"
[31]         xmlns="http://www.w3.org/1999/xhtml">
[32]         <head>
[33]           <title>Ein einfaches SLXHTML-Dokument mit enthaltenem Bild</title>
[34]           <style media="screen" type="text/css"/>
[35]         </head>
[36]         <body>
[37]           <p>Das nachfolgende Bild gibt einen Überblick zum Modell Bürgerkarte.</p>
[38]           <p>
[39]             <img alt="Modell Bürgerkarte"
[40]               src="http://www.buergerkarte.at/konzept/securitylayer/spezifikation/\
[41] 20040514/tutorial/examples/viewerformat/ModellBuergerkarte.gif"/>
[42]           </p>
[43]         </body>
[44]       </html>
[45]     </dsig:Object>
[46]   </dsig:Signature>
[47] </sl11:CreateXMLSignatureResponse>
```

Zeile 8 - 17 enthält die `dsig:Reference` auf das signierte SLXHTML-Dokument. Das Dokument selbst befindet sich im `dsig:Object` in den Zeilen 29 - 45.

In Zeile 19 - 26 erkennt man die `dsig:Reference`, die von der *Bürgerkarten-Umgebung* für das aus dem SLXHTML-Dokument referenzierte Bild erzeugt wurde. Das Attribut `Type` in Zeile 20 - 21 hat den Wert `http://www.buergerkarte.at/.../20031031?name=SignedImage&InstanceDocRef=0`, wobei der erste Teil immer gleich ist, und lediglich der Wert des URL-Parameters `InstanceDocRef` variiert. Dieser gibt als Ganzzahl an, die wievielte `dsig:Reference` in `dsig:SignedInfo` der XML-Signatur auf das SLXHTML-Dokument verweist, aus dem heraus das Bild referenziert wurde (wobei mit `0` zu zählen bekonnen wird). Das Attribut `URI` (Zeile 22 - 23) enthält den exakt gleichen Wert wie das Attribut `img/@src` des korrespondierenden Bildes (vergleiche Zeile 40 - 41).

#### *Downloads zu diesem Beispiel*

- examples/viewerformat/Image.CreateXMLSignatureRequest.directImgRef.xml
- examples/viewerformat/Image.CreateXMLSignatureRequest.directImgRef.Response.xml
- examples/viewerformat/Image.xhtml

#### *Weiterführende Informationen*

- Abschnitt 4, „Bilder im Standard-Anzeigeformat“ in Die österreichische Bürgerkarte - Standard-Anzeigeformat
- Abschnitt 2.1.7, „Image Module“ in Die österreichische Bürgerkarte - Standard-Anzeigeformat

### 4.3.2. Bilddaten als Ergänzungsobjekt

Das folgende Beispiel ist eine Abwandlung des letzten Beispiels: Signiert werden soll das gleiche SLXHTML-Dokument, wobei nun jedoch die Bildreferenz eine lokale Referenz ist und daher von der *Bürgerkarten-Umgebung* nicht aufgelöst werden kann. Die Applikation muss daher im Request ein Ergänzungsobjekt für die Bilddaten angeben.

#### 4.3.2.1. Anfrage

Bitte beachten Sie, dass das Beispiel aus Gründen der besseren Lesbarkeit formatiert und umgebrochen (erkennbar durch das Zeichen `\` am Zeilenende) wurde.

```
[01] <?xml version="1.0" encoding="UTF-8"?>
[02] <sl:CreateXMLSignatureRequest
[03]   xmlns:sl="http://www.buergerkarte.at/namespaces/securitylayer/1.2#">
[04]   <sl:KeyboxIdentifier>SecureSignatureKeypair</sl:KeyboxIdentifier>
[05]   <sl:DataObjectInfo Structure="enveloping">
[06]     <sl:DataObject>
[07]       <sl:XMLContent><html xmlns="http://www.w3.org/1999/xhtml">
[08]   <head>
[09]     <title>Ein einfaches SLXHTML-Dokument mit enthaltenem Bild</title>
[10]     <style type="text/css"/>
[11]   </head>
[12]   <body>
[13]     <p>Das nachfolgende Bild gibt einen Überblick zum Modell Bürgerkarte.</p>
[14]     <p>
[15]       <img src="ModellBuergerkarte.gif" alt="Modell Bürgerkarte"/>
[16]     </p>
[17]   </body>
[18] </html>
[19] </sl:XMLContent>
[20]     </sl:DataObject>
```

```
[21]      <sl:TransformsInfo>
[22]        <sl:FinalDataMetaInfo>
[23]          <sl:MimeType>application/xhtml+xml</sl:MimeType>
[24]        </sl:FinalDataMetaInfo>
[25]      </sl:TransformsInfo>
[26]      <sl:Supplement>
[27]        <sl:Content Reference="ModellBuergerkarte.gif">
[28]          <sl:Base64Content>...</sl:Base64Content>
[29]        </sl:Content>
[30]      </sl:Supplement>
[31]    </sl:DataObjectInfo>
[32] </sl:CreateXMLSignatureRequest>
```

In Zeile 15 erkennt man, dass `img/@src` diesmal eine relative Bildreferenz enthält, die von der *Bürgerkarten-Umgebung* nicht aufgelöst werden kann.

Es wird daher in Zeile 26 ein Ergänzungsobjekt (vergleiche Abschnitt 2.1.2.5, „Ergänzungsobjekte, Signaturmanifest“) zu den zu signierenden Daten angegeben. Der Wert des Attributs `sl:Content/@Reference` in Zeile 27 entspricht dabei exakt jenem des Attributs `img/@src` in Zeile 15.

#### 4.3.2.2. Antwort

Die Antwort der *Bürgerkarten-Umgebung* liefert die gleiche XML-Signatur wie das vorige Beispiel und wird daher hier nicht näher erläutert.

*Downloads zu diesem Beispiel*

- examples/viewerformat/Image.CreateXMLSignatureRequest.suppl.b64.xml
- examples/viewerformat/Image.CreateXMLSignatureRequest.suppl.b64.Response.xml

*Weiterführende Informationen*

- Abschnitt 4, „Bilder im Standard-Anzeigeformat“ in Die österreichische Bürgerkarte - Standard-Anzeigeformat
- Abschnitt 2.1.7, „Image Module“ in Die österreichische Bürgerkarte - Standard-Anzeigeformat

# Glossar

## Glossar

**Applikation**

Jenes Programm, das Anfragen an die *Bürgerkarten-Umgebung* über den *Security-Layer* richtet und die entsprechenden Antworten entgegennimmt und auswertet.

**Benutzer-Schnittstelle**

Jene Schnittstelle, über die der *Bürger* mit der *Bürgerkarten-Umgebung* kommuniziert. Über diese Schnittstelle wird einerseits die Benutzerinteraktion abgewickelt, die gegebenenfalls zur Abwicklung eines Befehls des *Security-Layers* notwendig ist (z.B. die Anzeige eines zu signierenden Dokuments beim Befehl zur Erzeugung einer XML-Signatur); andererseits kann der *Bürger* über diese Schnittstelle seine *Bürgerkarten-Umgebung* nach seinen persönlichen Bedürfnissen konfigurieren (z.B. kann er Einstellungen zum Zugriffsschutz auf seine Infoboxen verändern). Die Vorgaben an die *Benutzer-Schnittstelle* sind in Minimale Umsetzung des Security-Layers geregelt.

**Bürger**

Jene Person, die die Funktionen der *Bürgerkarten-Umgebung* für die sichere Abwicklung von E-Government oder E-Commerce verwenden möchte. Die Ansteuerung der *Bürgerkarten-Umgebung* erfolgt in der Regel nicht durch den *Bürger* selbst, sondern durch die *Applikation*, welche die E-Government oder E-Commerce Anwendung repräsentiert.

**Bürgerkarte**

Laut [E-GovG], §10 ZI 10 ist die *Bürgerkarte* „die unabhängig von der Umsetzung auf unterschiedlichen technischen Komponenten gebildete logische Einheit, die eine elektronische Signatur mit einer Personenbindung (§ 4 Abs. 2) und den zugehörigen Sicherheitsdaten und -funktionen sowie mit allenfalls vorhandenen Vollmachtsdaten verbindet“. Im Sinne der in den Spezifikationen zur österreichischen Bürgerkarte gebrauchten Terminologie ist die *Bürgerkarten-Umgebung* die Implementierung der logischen Einheit *Bürgerkarte*.

**Bürgerkarten-Umgebung**

Jenes Programm bzw. jener Dienst, der die Funktionalität der *Bürgerkarte* zur Verfügung stellt. Grundsätzlich vorstellbar ist die Ausführung als Programm, das lokal am Rechner des *Bürgers* läuft (*lokale Bürgerkarten-Umgebung*), oder als serverbasierter Dienst, der über das Internet angesprochen wird (*serverbasierte Bürgerkarten-Umgebung*). Die Interaktion mit diesem Programm bzw. Dienst wird über zwei Schnittstellen abgewickelt: Über die *Benutzer-Schnittstelle* sowie über den *Security-Layer*.

**Hash-Eingangsdaten**

Jene Daten, die für die Berechnung des Hash-Wertes für eine `dsig:Reference` verwendet werden. Sind für die `dsig:Reference` Transformationen angegeben, entsprechen diese Daten dem Ergebnis der letzten Transformation. Sind keine Transformationen spezifiziert, gleichen die Hash-Eingangsdaten den *Referenz-Eingangsdaten*.

**Impliziter Transformationsparameter**

Siehe Abschnitt 2.2.2.2, „Implizite Transformationsparameter“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer

**Referenz-Eingangsdaten**

Jene Daten, die sich aus der Auflösung der im Attribut `URI` der `dsig:Reference` angegebenen URI ergeben. Sind für die `dsig:Reference` Transformationen angegeben, werden diese Daten als Eingangsdaten zur Berechnung der ersten Transformation verwendet. Sind keine Transformationen spezifiziert, gleichen die Referenz-Eingangsdaten den *Hash-Eingangsdaten*.

**Security-Layer**

Jene Schnittstelle, über die die *Applikation* mit der *Bürgerkarten-Umgebung* kommuniziert. Das genaue Protokoll, das über diese Schnittstelle gesprochen werden kann, wird in Applikationsschnittstelle Security-Layer spezifiziert. Die möglichen Bindungen dieses Protokolls an Transportschichten wie HTTP oder TCP wird in Transportprotokolle Security-Layer geregelt.

**Signaturmanifest**

Siehe Abschnitt 2.2.2.2, „Implizite Transformationsparameter“ in Die österreichische Bürgerkarte - Applikationsschnittstelle Security-Layer .