



Zentrum für sichere Informationstechnologie – Austria
Secure Information Technology Center – Austria

A-1040 Wien, Weyringergasse 35
Tel.: (+43 1) 503 19 63-0
Fax: (+43 1) 503 19 63-66

A-8010 Graz, Inffeldgasse 16a
Tel.: (+43 316) 873-5514
Fax: (+43 316) 873-5520

<http://www.a-sit.at>
E-Mail: office@a-sit.at

DVR: 1035461

ZVR: 948166612

**GUTACHTEN ÜBER DIE EIGNUNG VON PRODUKTEN
FÜR DIE SICHERE SIGNATUR**

trustview

Version 2.1.1 R9

Mit der Änderung der Signaturverordnung durch BGBl. II Nr. 527/2004 vom 30. Dezember 2004 ist eine Bescheinigung nach §18(5) für Signaturprodukte, die der Umgebung der Signaturerstellungseinheit zuzuzählen sind, nicht mehr erforderlich. Dieses Gutachten geht in analoger Weise wie Bescheinigungen für derartige Produkte auf die Eignung für die sichere Signatur ein.

Projektnummer	A-SIT 1.035
Auftraggeber	IT Solution GmbH, 1070 Wien, Neubaugasse 12-14
Ansprechperson	Georg B. Schmidt
Auftrag erteilt am	27.4.2005
Typenbezeichnung	Trustview 2.1.1 R9 vom 6.12.2007
Gutachten ausgestellt am	27.12.2007
Vertraulichkeit	Kurzfassung zur Veröffentlichung

Inhalt

1. Zusammenfassung	3
2. Beschreibung des Produktes	3
2.1. Lieferumfang	3
2.2. Technische Einsatzumgebung	4
2.3. Funktionsumfang	4
2.4. Funktionsbeschreibung	5
2.4.1. Signaturformat 1: Security-Layer XML Format	5
2.4.2. Signaturformat 2: Security-Layer CMS Format	6
2.4.3. Prüfung des Dateiformates und Anzeige der Datenobjekte	7
3. Befundaufnahme	9
3.1. Referenzmuster	9
3.2. Unterlagen	9
3.3. Durchführung der Befundaufnahme	10
4. Gutachten	10
4.1. Eignung für die sichere Signatur	10
4.2. Detailgutachten	10
5. Einsatzbedingungen	10
6. Unterschriften	11
Anhang A – Security Layer 1.2 Format	12
1. Signatur nach CMS	12
1.1. Anfrage	12
1.1.1. Struktur der Signatur	12
1.1.2. Bezeichnung des Signaturschlüssels	12
1.1.3. Informationen zum Datenobjekt	12
1.1.4. Antwort	12
1.1.5. Signierte Metainformationen	12
1.1.6. Signierte Zertifikatsreferenz	13
1.1.7. Zeitpunkt der Signaturerstellung	13
2. Signatur nach XMLDSIG	13
2.1. Anfrage	13
2.1.1. Bezeichnung des Signaturschlüssels	13
2.1.2. Informationen zum Datenobjekt	13
2.1.3. Datenobjekt	13
2.1.4. Transformationswege	14
2.1.5. Informationen zum Signaturdokument	15
2.1.6. Antwort	16
2.1.7. Signierte Daten	16
2.1.8. Implizite Transformationsparameter	16
2.1.9. Signaturattribute	17
Anhang B – XHTML Schemata	19
1. SCHEMA 1 (LEGACY)	19
2. SCHEMA 2 (SLXHTML12)	38
2.1. Profil von XHTML 1.1	39
2.1.1. Einschränkungen gegenüber XHTML 1.1	39
2.1.2. XML-Schema für das Standard-Anzeigeformat	42
2.1.3. Sonstiges	42
2.2. Profil von CSS 2	42
2.2.1. Einbindung von CSS-Formaten in das Standard-Anzeigeformat	42
2.2.2. Anwendung von CSS-Formaten durch die Bürgerkarten-Umgebung	42
2.2.3. @-Regeln	43
2.2.4. CSS-Selektoren	44
2.2.5. CSS-Eigenschaften	44
2.3. Bilder im Standard-Anzeigeformat	50
2.3.1. Integration in die XML-Signatur	50
2.3.2. Prozessmodell für Signaturerstellung und Signaturprüfung	51

1. Zusammenfassung

A-SIT wurde von der IT Solution GmbH mit der Erstellung eines Gutachtens über die Eignung des Produktes „trustview 2.1.1 R9“ (nachstehend Secure Viewer genannt) für die sichere Signatur beauftragt.

Eine ausführliche Beschreibung des Produktes und seiner Funktion wird in Kapitel 2 gegeben und Kapitel 2.4.2 beschreibt die durchgeführten Befundaufnahmen.

Zusammenfassung der gutachterlichen Aussagen:

Wie in Kapitel 4 im Detail ausgeführt, ist der Secure Viewer unter den in Kapitel 5 genannten Einsatzbedingungen für die Darstellung der zu signierenden Daten in der Systemumgebung der Signaturerstellungseinheit bei sicheren Signaturen geeignet. Die gutachterlichen Aussagen sind zum Zeitpunkt des Ausstellens des gegenständlichen Gutachtens gültig.

2. Beschreibung des Produktes

Der Gegenstand des Gutachtens ist „trustview“, Version 2.1.1 R9.

Trustview ist die Viewer Komponente des Signaturclients „trustDesk“, welcher von der IT Solution GmbH in unterschiedlichen Versionen angeboten wird. Trustview erlaubt die Anzeige der zu signierenden Daten vor der Auslösung des Signaturvorganges.

Der Hersteller von trustview ist die IT Solution GmbH, A-1070 Wien, Neubaugasse 12-14.

2.1. Lieferumfang

Trustview 2.1.1 R9 wird als Komponente der Signaturclients der „trustDesk“ Familie ausgeliefert. trustDesk ist jedoch nicht Gegenstand dieses Gutachtens.

Die Benutzerdokumentation gehört nicht zum Standard Lieferumfang sondern kann gesondert von der Homepage der Firma IT Solution GmbH bezogen werden.

In folgenden Produkten ist Trustview 2.1.1 R9 inkludiert:

- (1) trustDesk basic
- (2) trustDesk standard
- (3) trustDesk professional
- (4) trustDesk business
- (5) trustDesk terminalserver
- (6) trustDesk infoterminal
- (7) trustDesk BAIK-Archiv
- (8) trustDesk Telekom
- (9) trustDesk bit4Health
- (10) trustDesk Angebotsassistent

Die Auslieferung an den Endkunden und das Packaging zum Vertrieb wird zur Gänze von der Firma IT Solution GmbH in den Firmenräumlichkeiten durchgeführt.

Folgende Auslieferungsformen stehen zur Verfügung:

- Auslieferung über das Internet
IT Solution stellt die Software auf der Homepage des Unternehmens zum Download zur Verfügung.
- Auslieferung auf Datenträger
Trustview kann auch auf einem read-only Datenträger (CD-ROM) per Postweg bzw. persönlicher Abholung bezogen werden.
- Auslieferung über Distributoren und Partner
Die Auslieferung an Distributoren bzw. Partner kann wie bereits oben beschrieben per Master CD Kopie oder per Download erfolgen.

Trustview ist nicht als Einzelkomponente erhältlich. Sie muss in einem von IT Solution signierten Distributionspaket ausgeliefert werden um sicherzustellen, dass die Software nicht manipuliert wurde.

2.2. Technische Einsatzumgebung

Im Rahmen des Signaturclients trustDesk und dessen Versionen kommt trustview auf unterschiedlichen Betriebssystemen zum Einsatz. Gegenstand dieses Gutachtens ist ausschließlich die Windows Version von trustview.

Folgende Versionen von Windows werden unterstützt:

- Windows 2000
- Windows 2000 Server
- Windows XP
- Windows 2003 Server
- Windows Vista

Die Technische Einsatzumgebung und die Hardwaremindestanforderungen können der Dokumentation¹ von trustDesk entnommen werden.

2.3. Funktionsumfang

Folgende Funktionen von trustview sind für dieses Gutachten relevant:

- Sichere Anzeige
- Datentransformation
 - C14N²
 - C14N with comments³
 - EC14N⁴
 - EC14N with comments⁵
 - Base64 Decoder⁶
 - XPath Filter 1⁷
 - XPath Filter 2⁸
 - Enveloped Signature⁹
 - XSLT¹⁰
- Hashberechnung
 - SHA-1¹¹ mit 160 Bit
 - RIPEMD160
- Auslösen der Signaturerstellung
 - RSA¹² mit Schlüssellänge 1024 Bit oder höher
 - ECDSA mit Schlüssellänge 192 Bit oder höher

Trustview kann ausgehend und abhängig von den durch den Signaturclient zu signierenden Daten zwei unterschiedliche Signaturformate erzeugen:

- Security-Layer¹³ XML Format

¹ http://itsolution.at/downloads/Benutzerdokumentation_%20trustDeskprofessional_1_0.pdf

² <http://www.w3.org/TR/2001/REC-xml-c14n-20010315>

³ <http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments>

⁴ <http://www.w3.org/2001/10/xml-exc-c14n>

⁵ <http://www.w3.org/2001/10/xml-exc-c14n#WithComments>

⁶ <http://www.w3.org/2000/09/xmldsig#base64>

⁷ <http://www.w3.org/TR/1999/REC-xpath-19991116>

⁸ <http://www.w3.org/2002/06/xmldsig-filter2>

⁹ <http://www.w3.org/2000/09/xmldsig#enveloped-signature>

¹⁰ <http://www.w3.org/TR/1999/REC-xslt-19991116>

¹¹ <http://www.w3.org/2000/09/xmldsig#sha1>

¹² <http://www.rsa.com/rsalabs/node.asp?id=2146>

- Security-Layer CMS Format

2.4. Funktionsbeschreibung

Der Secure Viewer trustview überprüft, ob das Format der Eingangsdatei einem der beiden oben genannten Signaturformaten entspricht. Je nach erkanntem Signaturformat führt der Secure Viewer die Anzeige und Signatur der in der Eingangsdatei enthaltenen zu signierenden Daten, entsprechend dem Signaturformat, durch und speichert das Signaturergebnis, wieder entsprechend dem Signaturformat, in einer Ausgangsdatei.

2.4.1. Signaturformat 1: Security-Layer XML Format

Eingangsdatei:

Trustview 2.1.1 R9 muss hierzu mit einer XML Datei, gemäß Security-Layer Spezifikation 1.2, welche einen gültigen *CreateXMLSignatureRequest* enthält, als Eingangsdatei gestartet werden.

Diese XML Datei enthält:

- Die Datenobjekte (*dataObject*), welche entweder im Format Text oder XML oder XHTML oder TIFF (Erweiterung) vorliegen
- (optional) die Transformationen für die Datenobjekte (*Transforms*) und
- einen Hinweis auf das für die Signatur zu verwendende Signaturschlüsselpaar (*Keyboxidentifizier*)

Signaturvorgang:

Die zu signierenden Daten sind gemäß Security-Layer 1.2 Spezifikation als *DataObject*, mit den vor der Signatur durchzuführenden *Transformationen*, in der *CreateXMLSignatureRequest-XML-Struktur* enthalten. Diese Datenobjekte werden extrahiert und mittels der, ebenfalls im *CreateXMLSignatureRequest* angegebenen, Transformationen transformiert. Anschließend wird gemäß W3C XMLDSig eine *SignedInfo-Datenstruktur* erzeugt, welche eine Liste der Referenzen auf die Signaturdaten und deren Hashwerte enthält. Weiters wird gemäß Security-Layer Spezifikation 1.2 ein ETSI Datenobjekt hinzugefügt, welches Daten über das Signaturzertifikat und den Signaturzeitpunkt enthält und gegebenenfalls ein Signaturmanifest, welches Referenzen und Hashwerte (Dokumenten-) externer Objekte enthält. Die Datenobjekte werden anschließend signiert.

Ausgangsdatei:

Als Ausgangsdatei wird aus den Eingangsdaten und der Signatur der Datenobjekte eine XML Datei erstellt, welche eine *CreateXMLSignatureResponse-XML-Struktur* gemäß Security-Layer Spezifikation 1.2 enthält. In dieser XML Datei enthalten sind:

- (optional) die signierten Datenobjekte
- die Signatur der Datenobjekte
- Das Signaturzertifikat
- Das ETSI Datenobjekt und
- (optional) das Signaturmanifest

Erweiterung XML Signatur um TIFF Datenobjekte

¹³ Die Applikationsschnittstelle Security-Layer zur österreichischen Bürgerkarte; siehe <http://www.buergerkarte.at/konzept/securitylayer/spezifikation/aktuell/>

- Das Datenobjekt muss eine TIFF Grafik enthalten
- Der URI des referenzierten Datenobjekts darf bei TIFF Datenobjekten neben *http* bzw. *https* URLs auch eine *file* Angabe beinhalten, welche eine Datei auf jenem Client referenziert, auf dem die Software installiert ist. Diese Angabe hat relativ zu erfolgen.
- Im Zuge der Prüfung des TIFF Datenobjektes auf dessen Gültigkeit werden bei Erkennen von unzulässigem Inhalt folgende Möglichkeiten zur Behebung angeboten:
 - Das Dokument automatisch korrigieren lassen (problematische TIFF Tags werden korrigiert)
 - Signaturvorgang abbrechen

2.4.2. Signaturformat 2: Security-Layer CMS Format

Der Secure Viewer Trustview muss mit einer XML Datei, gemäß Security-Layer Spezifikation 1.2, welche einen gültigen *CreateCMSSignatureRequest* enthält, gestartet werden.

Eingangsdatei:

Diese XML Datei enthält:

- Die Datenobjekte (*dataObject*), welche entweder im Format Text oder XML oder XHTML oder TIFF (Erweiterung) vorliegen
- einen Hinweis auf das für die Signatur zu verwendende Signaturschlüsselpaar (Keyboxidentifizier)

Signaturvorgang:

Ist die Eingangsdatei eine XML Datei mit gültiger *CreateCMSSignature-Request XML* Struktur, sind die zu signierenden Daten gemäß Security-Layer 1.2 Spezifikation als *DataObject* in der *CreateCMSSignatureRequest-XML-Struktur* enthalten. Diese Datenobjekte werden extrahiert und signiert.

Ausgangsdatei:

Als Ausgangsdatei wird eine XML Datei erstellt, welche eine *CreateCMSSignatureResponse XML-Struktur* gemäß Security-Layer 1.2 enthält. In dieser XML Datei ist das base64-kodierte CMS Signaturobjekt enthalten.

Erweiterung CMS Signatur um TIFF Datenobjekte

- Das Datenobjekt muss eine TIFF Grafik enthalten
- Der URI des referenzierten Datenobjekts darf bei TIFF Datenobjekten neben *http* bzw. *https* URLs auch eine *file* Angabe beinhalten, welche eine Datei auf jenem Client referenziert, auf dem die Software installiert ist. Diese Angabe hat relativ zu erfolgen.
- Im Zuge der Prüfung des TIFF Datenobjektes auf dessen Gültigkeit werden bei Erkennen von unzulässigem Inhalt folgende Möglichkeiten zur Behebung angeboten:
 - Das Dokument automatisch korrigieren lassen (problematische TIFF Tags werden korrigiert)
 - Signaturvorgang abbrechen

- Das TIFF Datenobjekt wird entsprechend dem Security-Layer CMS Request entweder als Base64 Content eingebettet oder mittels Content Attribut („Reference“) referenziert.

2.4.3. Prüfung des Dateiformates und Anzeige der Datenobjekte

Nach dem Laden der Eingangsdatei überprüft der Secure Viewer die Datei auf ihr Dateiformat.

Es wird zwischen folgenden Formaten unterschieden:

- Security-Layer XML mit folgenden Datenobjekten:
XHTML Subset / Encoding ISO-8859-1 gemäß XHTML-SCHEMA
XHTML Subset / Encoding UTF-8 gemäß XHTML-SCHEMA
TEXT / Encoding ISO-8859-1
TEXT / Encoding UTF-8
XML / Encoding ISO-8859-1
XML / Encoding UTF-8
TIFF Grafiken
- Security-Layer CMS mit folgenden Datenobjekten:
XHTML Subset / Encoding ISO-8859-1 gemäß XHTML-SCHEMA
XHTML Subset / Encoding UTF-8 gemäß XHTML-SCHEMA
TEXT / Encoding ISO-8859-1
TEXT / Encoding UTF-8
XML / Encoding ISO-8859-1
XML / Encoding UTF-8
TIFF Grafiken
- Unbekannte Formate

Dateiprüfung für Security-Layer XML

Wenn die geladene Datei diesem Format entspricht, werden die nötigen XMLDSIG Transformationen durchgeführt, um die Datenobjekte zu erhalten. Dabei kann es sich um folgende Datenobjekte handeln:

- XHTML
- TXT
- XML
- TIFF

Diese werden daraufhin geprüft und im Secure Viewer trustview 2.1.1 angezeigt, wobei immer nur ein Objekt pro Seite auf dem Bildschirm angezeigt wird.

Folgende Transformationen werden unterstützt:

Transformation	URI
C14N	http://www.w3.org/TR/2001/REC-xml-c14n-20010315
C14N with comments	http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments
EC14N	http://www.w3.org/2001/10/xml-exc-c14n
EC14N with comments	http://www.w3.org/2001/10/xml-exc-c14n#WithComments
Base64 Decoder	http://www.w3.org/2000/09/xmlsig#base64
XPath Filter 1	http://www.w3.org/TR/1999/REC-xpath-19991116
XPath Filter 2	http://www.w3.org/2002/06/xmlsig-filter2

Enveloped Signature	http://www.w3.org/2000/09/xmlsig#enveloped-signature
XSLT	http://www.w3.org/TR/1999/REC-xslt-19991116

Die nach den Transformationen erhaltenen Datenobjekte werden im Falle von Text und XML auf binären Inhalt, und im Falle von XHTML gegen ein Schema geprüft. Im Falle des Anzeigeformat nach „Standard-Anzeigeformat 1.2¹⁴“ werden weitere Prüfungen des enthaltenen CSS2 Stylesheets gemäß Spezifikation durchgeführt. Wenn ein XHTML Datenobjekt Grafiken referenziert, so werden diese auf ein gültiges GIF/JPEG Dateiformat gemäß Spezifikation „Standard-Anzeigeformat 1.2“ geprüft und in die Signatur aufgenommen.

Folgende Hashalgorithmen werden unterstützt:

Hashalgorithmus	URI
SHA-1 mit 160 bit	http://www.w3.org/2000/09/xmlsih#sha1
RipeMD mit 160 bit	http://www.w3.org/2001/04/xmlenc#ripemd160

Vor dem eigentlichen XMLDSIG Signaturvorgang werden weiters ETSI Properties als zusätzliches Datenobjekt eingefügt und in die Signatur aufgenommen. Werden im XHTML Datenobjekt Grafiken referenziert, so werden auch diese, sofern vorhanden, gemäß Spezifikation „Standard-Anzeigeformat 1.2“ in die Signatur aufgenommen.

Dateiprüfung für Security-Layer CMS

Wenn die geladene Datei diesem Format entspricht, wird das enthaltene Security-Layer CMS Objekt entpackt.

Bei diesem Datenobjekt kann es sich nur um ein einziges der folgenden Objekte handeln:

- XHTML
- TXT
- XML
- TIFF

Das Datenobjekt wird daraufhin geprüft und im Secure Viewer trustview 2.1.1 angezeigt.

Als Erweiterung des Formats wird zusätzlich zu den in den Spezifikationen beschriebenen MIME-Typen der zusätzliche MIME-Type „image/tiff“ und die Verarbeitung von TIFF Objekten analog zu den anderen Datenobjekten unterstützt.

Das erhaltene CMS Datenobjekt wird im Falle von Text und XML auf binären Inhalt geprüft, im Falle von XHTML gegen die Schemata in Appendix B geprüft. Im Falle von TIFF wird auf ein gültiges TIFF Image Format geprüft.

Als zusätzliche Erweiterung kann anstatt eines vollständigen Security-Layer CMS Signature Requests auch direkt das Datenobjekt übergeben werden (die Datei ohne Einbettung in XML Request).

Folgende Hashalgorithmen werden unterstützt:

Hashalgorithmus
SHA-1 mit 160 bit
RipeMD mit 160 bit

¹⁴ Standard-Anzeigeformat zur Bürgerkarten-Umgebung der österreichischen Bürgerkarte; siehe: <http://www.buergerkarte.at/konzept/securitylayer/spezifikation/aktuell/viewerformat/ViewerFormat.html>

Folgende Signaturalgorithmen werden unterstützt:

Signaturalgorithmus
RSA mit Schlüssellänge 1024 Bit oder höher
ECDSA mit Schlüssellänge 192 Bit oder höher

Dateiprüfung für unbekannte Formate

Wenn das Dateiformat keinem der bekannten erlaubten Formate entspricht, kann es nicht im Secure Viewer trustview 2.1.1 angezeigt werden. Über die Funktion „external Viewer“ kann das Datenobjekt jedoch angezeigt werden sofern für das Datenformat eine geeignete Applikation zur Anzeige definiert wurde. Die Anzeige in einer externen Applikation ist nicht Gegenstand dieses Gutachtens.

Es wird geprüft ob der MIME-Type, der im Signaturrequest angegeben ist, im Secure Viewer angezeigt werden kann.

Folgende MIME-Types sind vordefiniert:

- text/plain
- text/tab-separated-values
- text/sgml
- text/xml
- application/sgml
- application/xml
- message/rfc822
- text/html
- application/xhtml+xml
- image/tiff

Ist der angegebene MIME-Type in der Liste enthalten, wird das Dokument im Secure Viewer angezeigt.

Ist der angegebene MIME-Type nicht in dieser Liste enthalten, wird geprüft ob für den angegebenen MIME-Type vom Benutzer eine Applikation zur Anzeige konfiguriert wurde.

- a. Wenn ja, wird versucht das Dokument mit Hilfe der Applikation zu öffnen
- b. Wenn nein, wird dem Benutzer ein „Save as“ Dialog angeboten

3. Befundaufnahme

3.1. Referenzmuster

Der Hersteller hat

- den Secure Viewer trustview in der Version 2.1.1 R9 am 6.12.2007 als Download zur Verfügung gestellt.

3.2. Unterlagen

Der Hersteller hat folgende Dokumente zur Begutachtung bereitgestellt:

- Deckblatt zur Hash-Implementierung
- Auslieferung und Betrieb
- Security Target
- Handbuch
- Feinspezifikation

3.3. Durchführung der Befundaufnahme

Die Befundaufnahme wurde im Rahmen des Gutachtens von A-SIT durchgeführt. Als Leitlinie für die Begutachtung der Vertrauenswürdigkeit wurden die Gemeinsamen Kriterien für die Prüfung und Bewertung der Sicherheit von Informationstechnik (Common Criteria – ISO/IEC 15408) - Teil 3: Anforderungen an die Vertrauenswürdigkeit (Vertrauenswürdigkeitsstufe EAL3) herangezogen.

Folgende Bereiche wurden begutachtet:

- Sicherheitsvorgaben
- Konfigurationsmanagement
- Auslieferung und Betrieb der begutachteten Komponente
- Entwicklung der begutachteten Komponente
- Handbücher
- Lebenszyklus-Unterstützung
- Tests
- Schwachstellenbewertung

Eine Bewertung der Mechanismenstärke wurde nicht durchgeführt.

4. Gutachten

4.1. Eignung für die sichere Signatur

Der Secure Viewer überprüft, ob die zu signierenden Daten einem der oben definierten Content-Formate entsprechen und ermöglicht, dass dem Signator die zu signierenden Daten vor Auslösung des Signaturvorganges dargestellt werden. In den verwendeten Formaten können keine dynamischen Veränderungen codiert werden.

Der Secure Viewer ist damit unter den in Kapitel 5 angeführten Einsatzbedingungen für die Darstellung der zu signierenden Daten in der Systemumgebung der Signaturerstellungseinheit bei sicheren Signaturen geeignet.

Das gegenständliche Gutachten stellt eine Momentaufnahme unter Berücksichtigung des aktuellen Standes der Technik zum Zeitpunkt der Ausstellung dar. Die Aussagen sind daher zum Zeitpunkt der Ausstellung bei Berücksichtigung aller in Kapitel 5 genannten Einsatzbedingungen gültig.

4.2. Detailgutachten

Hinweis: Dieses Kapitel ist in der ggf. auf der A-SIT-Website veröffentlichten Fassung nicht enthalten.


5. Einsatzbedingungen

- (1) Die vorgesehene Einsatzumgebung von trustview sind Rechner mit den unter 2.2 genannten Betriebssystemen. Der Zugang zum verwendeten Rechner kann vom Signator kontrolliert werden. Manipulationen an der Hardware und Software des Rechners, auf dem der Secure Viewer als Teil einer Signatursoftware (siehe 2.1) installiert ist, sind zu verhindern. Es ist sicherzustellen, dass die Sicherheit der technischen Einsatzumgebung nicht kompromittiert ist.
- (2) Für einen sicheren Betrieb ist es erforderlich, dass die Empfehlungen der Benutzerdokumentation eingehalten und die Anforderungen an die Einsatzumgebung beachtet werden. Insbesondere muss sich der Signator vergewissern, dass die Bildschirmauflösung und Farbtiefe korrekt gemäß den Benutzungsvorschriften des Herstellers eingestellt ist. Die Integrität des Secure Viewers, der verwendeten Programmbibliotheken und Treiber sowie der für die Anzeige erstellten temporären Dateien ist durch geeignete technische und/oder organisatorische Maßnahmen in der Einsatzumgebung sicherzustellen.

- (3) Zur Erzeugung der sicheren elektronischen Signatur sind ausschließlich sichere Signaturerstellungseinheiten zu verwenden, welche die Anforderungen von SigG und SigV erfüllen.
- (4) Die Verantwortung für die Integrität der Daten bei der Übertragung zum zur Verbindung des Rechners mit der Signaturerstellungseinheit verwendeten Chipkartenleser liegt nicht im Verantwortungsbereich der begutachteten Komponente. Die Integrität der Daten ist durch geeignete technische und/oder organisatorische Maßnahmen in der Einsatzumgebung sicherzustellen. Der Signator muss sich von der unmittelbaren und sicheren Verbindung des Chipkartenlesers mit dem Arbeitsplatzrechner vergewissern können.
- (5) Die verwendeten Formate zur Darstellung des Inhaltes der zu signierenden Daten (siehe auch 2.4) müssen vom Zertifizierungsdiensteanbieter empfohlen sein.¹⁵

6. Unterschriften

A-SIT Zentrum für sichere Informationstechnologie - Austria

Signaturwert	k/QK5XVtQcVRPzfPm39hYbD7b9mb016wwAwBm7zegLHD61YZySTV5c8ikZVWF+1n	
	Unterzeichner	Geschäftsführender Vorstand, Manfred Holzbach
	Datum/Zeit-UTC	2007-12-27T10:46:55Z
	Aussteller-Zertifikat	CN=a-sign-Premium-Sig-02,OU=a-sign-Premium-Sig-02,O=A-Trust Ges. f. Sicherheitssysteme im elektr. Datenverkehr GmbH,C=AT
	Serien-Nr.	97349
	Methode	urn:pdfsigfilter:bka.gv.at:text:v1.1.0
	Parameter	etsi-bka-1.0@1198752415-2233078@2207-17452-0-17834-27932
Prüfhinweis	Informationen zur Signaturprüfung finden Sie unter: www.a-sit.at/de/dokumente_publicationen/a-sit_signaturen/index.php .	

¹⁵ Diese Einsatzbedingung ist nur relevant, solange SigV § 4 Abs. 1, erster Satz, in der Fassung von BGBl. II Nr. 527/2004 vom 30. Dezember 2004 in Kraft ist. („Für die Darstellung des Inhalts der zu signierenden Daten vor der Auslösung des Signaturvorgangs dürfen nur die vom Zertifizierungsdiensteanbieter empfohlenen Formate verwendet werden.“)

Anhang A – Security Layer 1.2 Format

Auszug aus der Spezifikation: Die Applikationsschnittstelle Security-Layer zur österreichischen Bürgerkarte. Applikationsschnittstelle Security-Layer Version 1.2.2 vom 1.3.2005, publiziert unter: <http://www.buergerkarte.at/konzept/securitylayer/spezifikation/aktuell/core/Core.html>

Die Schnittstelle Security-Layer unterstützt zwei mögliche Formate für die Erzeugung einer elektronischen Signatur: [CMS] und [XMLDSIG].

1. Signatur nach CMS

1.1. Anfrage

Mit einer Signatur nach [CMS] kann genau ein Datenobjekt signiert werden.

1.1.1. Struktur der Signatur

Zunächst muss im Attribut Structure der Signaturanfrage (sl:CreateCMSSignatureRequest) angegeben werden, ob das nachfolgend spezifizierte Datenobjekt in die Signaturstruktur eingebunden werden soll (Wert "enveloping"), oder nicht (Wert "detached").

1.1.2. Bezeichnung des Signaturschlüssels

Weiters muss in der Signaturanfrage der Bezeichner des für die Signaturerstellung zu verwendenden Schlüssels (sl:KeyboxIdentifier) angegeben werden. Bezeichner aller verfügbaren Schlüssel können mit Hilfe des Befehls sl:GetPropertiesRequest von der Bürgerkarten-Umgebung abgefragt werden.

1.1.3. Informationen zum Datenobjekt

Schließlich enthält die Signaturanfrage einen Behälter sl:DataObject, der das zu signierende Datenobjekt (sl:Content), sowie Metainformationen (sl:MetaInfo) für die Anfertigung der Signatur sowie für die gegebenenfalls notwendige Anzeige in der Bürgerkarten-Umgebung enthält.

Jedenfalls an Metainformation spezifiziert werden muss der Mime-Type (siehe [MIME]) des zu signierenden Datenobjekts. Weiters darf eine verbale Beschreibung des Datenobjekts angegeben werden (sl:Description). Schließlich dürfen noch weitere beliebige Elemente zu diesen Metainformationen hinzugefügt werden.

Anmerkung: Soll das zu signierende Datenobjekt von der Bürgerkarten-Umgebung als Text interpretiert werden, muss die Applikation den Mime-Type text/plain verwenden; soll es im Sinne des Standard-Anzeigeformats des Security-Layer interpretiert werden, muss die Applikation den Mime-Type application/xhtml+xml verwenden. Siehe dazu auch Minimale Umsetzung des Security-Layers.

Die Angabe des zu signierenden Datenobjekts kann auf zwei Arten erfolgen: Entweder enthält das Element sl:Content die base64-kodierten Daten, oder das Element sl:Content ist leer, hat aber sein Attribut Reference gesetzt. Die Bürgerkarten-Umgebung muss im letzten Fall versuchen, die in diesem Attribut angegebene URI aufzulösen, um so die zu signierenden Daten zu erhalten.

1.1.4. Antwort

Die Antwort besteht aus dem Element sl:CMSSignature, welches die erzeugte Signatur nach [CMS] in base64-kodierter Form enthält. Für detaillierte Vorgaben für die zu erzeugende Signatur hinsichtlich Digest-Algorithmen, Signatur-Algorithmen und Schlüsselinformationen siehe *Minimale Umsetzung des Security-Layers*.

1.1.5. Signierte Metainformationen

In die [CMS]-Signatur muss ein signiertes Signaturattribut ContentHints nach [ESS-S/MIME], Abschnitt 2.9 aufgenommen werden. Zur Anfertigung dieses Signaturattributs sind jene Metainformationen (sl:MetaInfo) zu verwenden, die in der Anfrage im Behälter für das Datenobjekt (sl:DataObject) angegeben wurden.

Für das Element sl:MimeType aus sl:MetaInfo muss ein erstes Feld contentDescription in ContentHints verwendet werden; ist das Element sl:Description aus sl:MetaInfo vorhanden, muss es in einem weiteren Feld contentDescription in ContentHints codiert werden. Das Feld contentType muss jedenfalls mit dem Wert des Object Identifiers für id-data (siehe [CMS], Abschnitt 4) belegt werden.

1.1.6. Signierte Zertifikatsreferenz

Weiters muss ein signiertes Signaturattribut OtherSigningCertificate nach [ETSI-CMS] in die [CMS]-Signatur aufgenommen werden, mit dem das für die Verifikation der Signatur zu verwendende Signatorzertifikat eindeutig identifiziert wird.

1.1.7. Zeitpunkt der Signaturerstellung

Schließlich muss ein Signaturattribut SigningTime nach [ETSI-CMS] in die [CMS]-Signatur aufgenommen werden, das den vom Signator behaupteten Zeitpunkt der Signaturerstellung enthält.

Anmerkung: Für Beispiele zu diesem Befehl siehe *Tutorium*; für Anforderungen an die Benutzerschnittstelle siehe *Anforderungen an die Benutzerschnittstelle, Abschnitt 3.1*.

2. Signatur nach XMLDSIG

2.1. Anfrage

Die Signatur nach [XMLDSIG] erlaubt im Gegensatz zur Signatur nach [CMS] auch die Signierung mehrerer Datenobjekte mittels einer einzigen Signatur.

2.1.1. Bezeichnung des Signaturschlüssels

In der Signaturanfrage muss der Bezeichner des für die Signaturerstellung zu verwendenden Schlüssels (sl:KeyboxIdentifier) angegeben werden. Bezeichner aller verfügbaren Schlüssel können mit Hilfe des Befehls sl:GetPropertiesRequest von der Bürgerkarten-Umgebung abgefragt werden.

2.1.2. Informationen zum Datenobjekt

In der Folge enthält die Signaturanfrage für jedes Datenobjekt, das von der Signatur unterschrieben werden soll, einen Behälter (sl:DataObjectInfo), der Informationen für die Anfertigung der Signatur sowie für die gegebenenfalls notwendige Anzeige in der Bürgerkarten-Umgebung enthält. sl:DataObjectInfo ist mit einem Attribut Structure ausgestattet, das angibt, ob das im Behälter spezifizierte Datenobjekt in die Signaturstruktur eingebunden werden soll (Wert "enveloping"), oder ob die Signatur dieses Datenobjekt nur referenzieren soll (Wert "detached").

2.1.3. Datenobjekt

Ein solcher Behälter besteht zunächst aus Informationen zu jenem Datenobjekt, das gegebenenfalls transformiert und nachfolgend signiert wird (sl:DataObject). In Abhängigkeit des oben besprochenen Attributs Structure sind für Inhalt von sl:DataObject sowie sein Attribut Reference folgende Kombinationen zulässig (alle anderen Kombinationen sind ungültig):

Struktur	Möglichkeit	Beschreibung
"enveloping"	A ^(1,2,3)	Das Attribut Reference wird nicht verwendet, der Inhalt von sl:DataObject repräsentiert das Datenobjekt. Ist das Datenobjekt XML-kodiert (Verwendung des Elements sl:XMLContent in sl:DataObject), muss es als gepartes XML in die Signaturstruktur eingebunden werden.
	B ^(2,4)	Das Attribut Reference enthält eine URI, die von der <u>Bürgerkarten-Umgebung</u> aufgelöst werden muss, um das Datenobjekt zu erhalten. Der Inhalt von sl:DataObject bleibt leer. Handelt es sich bei dem so referenzierten Datenobjekt um Text oder XML-Daten, ist es sinnvoll, das Datenobjekt als gepartes XML in die Signaturstruktur einzubinden. Die <u>Bürgerkarten-Umgebung</u> soll daher bei der Auflösung Informationen des Transportprotokolls auswerten, um das eventuelle Vorliegen von Text oder XML-Daten herauszufinden. Für eine URI, die als Protokoll http oder https aufweist, muss dazu die Information im HTTP-Header (Content-Type) ausgewertet werden.

"detached"	C	Das Attribut Reference enthält eine URI, die von der <u>Bürgerkarten-Umgebung</u> aufgelöst werden muss, um das Datenobjekt zu erhalten. Darüber hinaus wird diese URI auch für die Kodierung der Referenz auf das Datenobjekt als Bestandteil der XML-Signatur verwendet (Attribut URI im Element dsig:Reference). Der Inhalt von sl:DataObject bleibt leer.
	D ^(1,5)	Das Attribut Reference enthält eine URI, die von der <u>Bürgerkarten-Umgebung</u> für die Kodierung der Referenz auf das Datenobjekt als Bestandteil der XML-Signatur verwendet wird (Attribut URI im Element dsig:Reference). Der Inhalt von sl:DataObject repräsentiert das Datenobjekt.

(1) Soll der Inhalt von sl:Dataobject zur expliziten Angabe des Datenobjekts verwendet werden, stehen drei unterschiedliche Arten der Kodierung zur Verfügung:

sl:Base64Content: Enthält das Datenobjekt in base64-kodierter Form.

sl:XMLContent: Enthält das Datenobjekt als XML kodiert. Wenn es aus Sicht der Applikation wichtig ist, dass die Bürgerkarten-Umgebung Whitespace innerhalb des übergebenen XML nicht verändert, kann für sl:XMLContent das Attribut xml:space mit dem Wert preserve versehen werden. Das Inhaltsmodell von sl:XMLContent ist so definiert, dass es eine beliebige Mischung aus Text und XML-Markup erlaubt. Das schließt ausdrücklich auch reinen Text mit ein. Eine gültige Instanz von sl:XMLContent ist also beispielsweise auch <sl:XMLContent>Text</sl:XMLContent>.

sl:LocRefContent: Enthält eine Referenz auf das Datenobjekt, die von der Bürgerkarten-Umgebung aufzulösen ist, um das Datenobjekt zu erhalten.

(2) Die Einbindung eines im Fall A oder B übergebenen Datenobjekts in die XML-Signatur bzw. die Referenzierung auf dieses in die Signatur eingebundene Datenobjekt aus dem zugehörigen dsig:Reference Element der XML-Signatur muss so erfolgen, dass ausschließlich die im Request in sl:DataObject übergebenen Daten signiert werden. Wird beispielsweise das Datenobjekt als Inhalt eines dsig:Object Elements in die XML-Signatur eingebunden, darf dieses dsig:Object Containerelement nicht mitsigniert werden, sondern lediglich sein Inhalt.

(3) Wenn das Datenobjekt base64-kodiert vorliegt (Verwendung des Elements sl:Base64Content in sl:DataObject), müssen jedenfalls die Base64-dekodierte Daten signiert werden. Kann die Bürgerkarten-Umgebung die Base64-dekodierte Daten nicht direkt in das dsig:Object integrieren (weil darin Zeichen enthalten sind, die nicht als XML-Text darstellbar sind), muss sie stattdessen die base64-kodierten Daten integrieren und eine Base64 Transformation als erste Transformation im zugehörigen dsig:Reference verwenden.

(4) Handelt es sich beim referenzierten Datenobjekt weder um Text noch um XML-Daten, oder wird das Datenobjekt entgegen der Empfehlung nicht auf Vorliegen von Text oder XML-Daten geprüft, muss es in base64-kodierter Form in die Signaturstruktur eingebunden werden. Signiert werden muss jedoch das ursprüngliche Datenobjekt; es ist daher in der auf das eingebundene Datenobjekt verweisenden dsig:Reference eine Base64 Transformation zu verwenden.

(5) Handelt es sich bei der in Reference übergebenen URI um eine interne *URI gemäß [URI]*, Abschnitt 4.2 (*Same-document Reference*), so ist sie auf jenes XML-Dokument zu beziehen, in das die Signatur eingebettet werden soll (und welches in sl:SignatureEnvironment übergeben wird - vergleiche Abschnitt Informationen zum Signaturdokument).

2.1.4. Transformationswege

Weiters enthält der Behälter sl:DataObjectInfo einen oder mehrere Transformationswege für das Datenobjekt (sl:TransformsInfo).

Ein Transformationsweg beschreibt dabei eine Kette von auszuführenden Transformationen (dsig:Transforms), um vom Datenobjekt zu jenen Daten zu gelangen, die in die Hashberechnung und in weiterer Folge in die Signaturberechnung einfließen (nachfolgend als Hash-Eingangsdaten bezeichnet).

Die Hash-Eingangsdaten sind gleichzeitig jene Daten, die gegebenenfalls in der Bürgerkarten-Umgebung dem Benutzer angezeigt werden müssen. Damit die Bürgerkarten-Umgebung weiß, welcher Natur die Hash-Eingangsdaten sind, enthält der Transformationsweg andererseits Metainformationen darüber (sl:FinalDataMetaInfo). Jedenfalls ist der Mime-Type (siehe [MIME]) anzugeben (sl:MimeType), zusätzlich darf eine verbale Beschreibung dieser Daten angegeben werden (sl:Description).

Anmerkung: Soll das zu signierende Datenobjekt von der Bürgerkarten-Umgebung als Text interpretiert werden, *muss* die Applikation den Mime-Type text/plain verwenden; soll es im Sinne des Standard-Anzeigeformats des Security-Layer interpretiert werden, *muss* die Applikation den Mime-Type application/xhtml+xml verwenden. Siehe dazu auch Minimale Umsetzung des Security-Layers.

Soll das Datenobjekt direkt unterzeichnet werden, wird ebenfalls ein Transformationsweg spezifiziert, jedoch unterbleibt die Angabe der Kette von Transformationen. Bei Angabe mehrerer Transformationswege wählt die Bürgerkarten-Umgebung einen Weg frei aus.

Bei der Angabe eines Transformationsweges muss die Applikation sicherstellen, dass alle Namenräume, die innerhalb der Struktur der spezifizierten Transformationen (dsig:Transforms) verwendet werden, innerhalb dieser Struktur explizit deklariert werden. Die Bürgerkarten-Umgebung darf nicht Namenraum-Deklarationen innerhalb von dsig:Transforms hinzufügen, wenn Sie die Struktur in die zu erzeugende Signatur übernimmt.

Ergänzungsobjekte

Optional werden schließlich im Behälter sl:DataObjectInfo Ergänzungsobjekte (sl:Supplements) angegeben. Solche können übergeben werden, damit Daten, die im Rahmen des Transformationsprozesses für das Datenobjekt oder für die Anzeige benötigt werden, nicht von der Bürgerkarten-Umgebung aufgelöst werden müssen. Das Datenobjekt selbst darf jedoch nicht als Ergänzungsobjekt übergeben werden.

Als Beispiel sei hier eine Stylesheet-Transformation angeführt, die sich verschachtelter Stylesheets bedient: Nur der Basis-Stylesheet ist tatsächlich im Transformationsobjekt (dsig:Transform) als Parameter angeführt; im Basis-Stylesheet referenzierte weitere Stylesheets müssten von der Bürgerkarten-Umgebung selbst aufgelöst werden. Dies kann vermieden werden, indem solche referenzierte Stylesheets als Ergänzungsobjekte übergeben werden.

Ein Ergänzungsobjekt besteht dabei einerseits aus optionalen Metainformationen (vergleiche sl:FinalDataMetaInfo), andererseits aus dem eigentlichen Daten (sl:Content): Das verpflichtend zu verwendende Attribut Reference enthält dabei als URI die Referenz auf die Ergänzungsdaten, und zwar so, wie sie von der Bürgerkarten-Umgebung zur Auflösung verwendet werden würde. Der Inhalt von sl:Content repräsentiert die Ergänzungsdaten (siehe auch Hinweis in Abschnitt 2.2.1, Datenobjekt).

Stößt die Bürgerkarten-Umgebung während der Berechnung des Transformationsprozesses auf aufzulösende Daten, muss Sie folgenden Ablauf für die Auflösung einhalten:

1. Prüfung, ob die aufzulösende Referenz in einem sl:Supplement innerhalb jenes sl:DataObjectInfo vorkommt, das den gerade berechneten Transformationsprozess spezifiziert. Ist diese Prüfung erfolgreich, sind die in diesem sl:Supplement angegebenen Daten als Ergebnis der Auflösung zu verwenden. Ansonsten ist mit Schritt 2 fortzufahren.
2. Prüfung, ob die aufzulösende Referenz in einem sl:Supplement eines anderen sl:DataObjectInfo vorkommt, das im Befehl zur Erzeugung der XML-Signatur spezifiziert wurde. Die einzelnen sl:DataObjectInfo-Elemente sind dabei in jener Reihenfolge zu untersuchen, in der Sie im Befehl angegeben wurden. Die im ersten so gefundenen sl:Supplement angegebenen Daten sind als Ergebnis der Auflösung zu verwenden. Wird kein sl:Supplement gefunden, ist mit Schritt 3 fortzufahren.
3. Auflösung der Referenz.

2.1.5. Informationen zum Signaturdokument

Soll die zu erstellende Signatur in ein bestehendes XML-Dokument eingebettet werden, findet sich in der Signaturanfrage schließlich der Behälter sl:SignatureInfo. Bei Fehlen dieses Behälters ist die Signatur nirgends einzubetten, sondern direkt als Ergebnis der Anfrage (siehe Abschnitt 2.2.2) zurückzuliefern.

Das Signaturdokument

sl:SignatureInfo enthält zunächst Angaben zum Signaturdokument, in das die Signatur eingebettet werden soll (sl:SignatureEnvironment). Entweder enthält das Attribut Reference einen Verweis auf dieses Dokument, der von der Bürgerkarten-Umgebung aufzulösen ist, oder das Dokument selbst ist als Inhalt von sl:SignatureEnvironment angegeben.

Die direkte Einbettung kann auf zwei Arten erfolgen: Entweder wird das XML-Dokument base64 kodiert und als Text des Kindelements sl:Base64Content integriert, oder aber das Wurzelement des XML-Dokuments wird direkt als einziges Kind des Kindelements sl:XMLContent angegeben. Siehe dazu auch den Hinweis in Abschnitt 2.2.1, Datenobjekt.

Anmerkung: Wird ein XML-Dokument als Signaturdokument verwendet, das eine Document Type Declaration (siehe [XML]) verwendet, *soll* die erste Variante der direkten Einbettung (base64 Kodierung) verwendet werden, damit die dort enthaltenen Informationen vom XML-Parser der Bürgerkarten-Umgebung ausgewertet werden können. Bei der Einbettung des Wurzelements in sl:XMLContent können diese Informationen nicht angegeben werden.

Anmerkung: Für das Parsen des Signaturdokuments wird folgende Vorgangsweise empfohlen: In einem ersten Schritt versucht die Bürgerkarten-Umgebung das Dokument validierend zu parsen.

Informationen über die Grammatik des Dokuments liefern dazu eine ggf. im Dokument vorhandene Document Type Declaration (siehe [XML]), oder XML-Schemata, die mit den in [XML-Schema], Abschnitt 2.6.3 angegebenen Mechanismen referenziert werden. Sollte dieser erste Versuch scheitern, parst die Bürgerkarten-Umgebung das Dokument in einem zweiten Anlauf nichtvalidierend, d. h. ohne Auswertung von Grammatikinformatoren.

Position der Signatur

Das darauffolgende Element sl:SignatureLocation enthält Informationen, an welcher Position im Signatordokument die zu erstellende Signatur von der Bürgerkarten-Umgebung eingefügt werden soll.

Der Textinhalt des Elements enthält einen Ausdruck nach [XPath], mit dem das Eltern-Element der einzufügenden Signatur ausgewählt wird. Als Kontext-Knoten für die Auswertung des XPath-Ausdrucks ist der Dokument-Knoten des in sl:SignatureEnvironment spezifizierten XML-Dokuments zu verwenden. Werden im XPath-Ausdruck Namespace-Prefixes verwendet, müssen die entsprechenden Namespace-Deklarationen im Kontext des Elements sl:SignatureLocation bekannt sein.

Das Attribut Index selektiert die Position der Signatur innerhalb des Elternelements. Hat Index den Wert 0, wird die Signatur als erster Kindknoten des Elternelements eingefügt, hat Index den Wert n, wird die Signatur unmittelbar nach dem n-ten Kindknoten des Elternelements eingefügt.

Ergänzungsobjekte

Zusätzlich können schließlich Ergänzungsobjekte (sl:Supplements) spezifiziert werden, die in Zusammenhang mit dem Signatordokument stehen.

Beispielsweise könnte im Signatordokument, das in sl:SignatureEnvironment spezifiziert wurde, ein Verweis auf die Dokumenttypdefinition enthalten sein. Das auf diese Weise referenzierte Dokument kann von der Applikation als Ergänzungsobjekt übergeben werden, wenn die Bürgerkarten-Umgebung den Verweis nicht selbst auflösen soll, oder wenn die Bürgerkarten-Umgebung den Verweis gar nicht auflösen kann, weil es sich etwa um einen relativen Veweis, bezogen auf das Signatordokument, handelt.

Ein weiteres Beispiel für die Verwendung eines Ergänzungsobjekts könnte ein XML-Schema sein, das im mittels sl:SignatureEnvironment spezifizierten Signatordokument unter Verwendung der in [XML-Schema] vorgeschlagenen Mechanismen (siehe Anmerkung im vorangegangenen Abschnitt *Das Signatordokument*) referenziert wird.

Ein Ergänzungsobjekt besteht dabei einerseits aus optionalen Metainformationen (vergleiche sl:FinalDataMetaInfo in Abschnitt 2.2.1, Transformationswege), andererseits aus dem eigentlichen Daten (sl:Content): Das verpflichtend zu verwendende Attribut Reference enthält dabei als URI die Referenz auf die Ergänzungsdaten, und zwar so, wie sie von der Bürgerkarten-Umgebung zur Auflösung verwendet werden würde. Der Inhalt von sl:Content repräsentiert die Ergänzungsdaten (siehe auch Hinweis in Abschnitt 2.2.1, Datenobjekt).

2.1.6. Antwort

Die Antwort enthält die nach [XMLDSIG] kodierte elektronische Signatur. Wurde in der Anfrage ein sl:SignatureInfo Element angegeben, enthält die Antwort als einziges Kind das in sl:SignatureInfo angegebene Dokument mit der darin integrierten Signatur. Ansonsten enthält die Antwort direkt die erzeugte Signatur.

2.1.7. Signierte Daten

Für jedes in der Anfrage mittels Behälter (sl:DataObjectInfo) übergebene Datenobjekt enthält die XML-Signatur ein dsig:Reference Element. In dessen dsig:Transforms Element ist jene Transformationskette anzugeben, die in der Bürgerkarten-Umgebung durchlaufen wurde, um aus dem übergebenen Datenobjekt jene Daten zu erhalten, die für die Berechnung des Hash-Wertes sowie gegebenenfalls für die Anzeige im Secure Viewer trustview 2.1.1 verwendet worden sind.

2.1.8. Implizite Transformationsparameter

Um den korrekten Zusammenhang zwischen den *Referenz-Eingangsdaten* sowie den Hash-Eingangsdaten später jederzeit überprüfen zu können, müssen die impliziten Transformationsparameter aller in die Signatur aufzunehmenden Datenobjekte in ein einziges Signaturmanifest aufgenommen werden. Liegen keine impliziten Transformationsparameter vor, darf das Signaturmanifest nicht erstellt werden.

Ein impliziter Transformationsparameter ist in diesem Zusammenhang ein Datum, das von der Bürgerkarten-Umgebung zur Berechnung der Transformationen für ein zu signierendes

Datenobjekt verwendet wurde, jedoch nicht explizit als Parameter im entsprechenden Transformationsobjekt (dsig:Transform) aufscheint.

Als (derzeit einzig bekanntes) Beispiel soll hier wiederum die bereits in Abschnitt 2.2.1 erwähnte Stylesheet-Transformation erwähnt werden, die sich verschachtelter Stylesheets bedient. Die im Basis-Stylesheet referenzierten weiteren Stylesheets sind implizite Transformationsparameter im obigen Sinne.

Das Signaturmanifest (dsig:Manifest) enthält für jeden impliziten Transformationsparameter ein eigenes Referenzobjekt (dsig:Reference), das einen Hash-Wert für den impliziten Transformationsparameter inkludiert. Transformationen in den Referenzobjekten des Signaturmanifests dürfen nicht verwendet werden. Das Attribut URI eines Referenzobjekts enthält dabei die Referenz auf den impliziten Transformationsparameter, und zwar in exakt gleicher Weise, wie sie von der Bürgerkarten-Umgebung im Falle der Signaturprüfung zur Auflösung verwendet werden würde.

Die Eingangsdaten für die Berechnung des Hash-Wertes über einen impliziten Transformationsparameter ergeben sich wie folgt:

- Wird der implizite Transformationsparameter als Referenz angegeben, die von der Bürgerkarten-Umgebung selbst aufzulösen ist, ist zwischen einer externen und einer internen Referenz zu unterscheiden:
 - Die Auflösung einer externen Referenz muss einen Byte-Stream liefern. Dieser Byte-Stream bildet die Eingangsdaten für die Hash-Berechnung.
 - Die Auflösung einer internen Referenz muss eine XPath-Knotenmenge liefern (vgl. [XMLDSIG, Abschnitt 4.3.3.3]). Diese Knotenmenge ist nach [C14N] zu kanonisieren, um einen eindeutigen Byte-Stream zu erhalten. Dieser Byte-Stream bildet dann die Eingangsdaten für die Hash-Berechnung.
- Wird der implizite Transformationsparameter als Referenz angegeben, die von der Bürgerkarten-Umgebung nicht selbst aufzulösen ist, weil in der Anfrage ein entsprechendes Ergänzungsobjekt angegeben wurde, ist wiederum zwischen einer externen und einer internen Referenz zu unterscheiden:
 - Enthält das Ergänzungsobjekt Daten für eine externe Referenz, müssen diese Daten als Base64 (in sl:Supplement/sl:Content/sl:Base64Content) vorliegen. Die Base64-dekodierten Daten bilden dann die Eingangsdaten für die Hash-Berechnung.
 - Enthält das Ergänzungsobjekt Daten für eine interne Referenz, müssen diese Daten als XML (in sl:Supplement/sl:Content/sl:XMLContent) vorliegen. Aus diesen Daten ist entsprechend [XMLDSIG, Abschnitt 4.3.3.3] eine XPath-Knotenmenge zu erzeugen. Diese Knotenmenge ist nach [C14N] zu kanonisieren, um einen eindeutigen Byte-Stream zu erhalten. Dieser Byte-Stream bildet dann die Eingangsdaten für die Hash-Berechnung.

Die Einbindung des Signaturmanifests in die Signatur erfolgt durch ein eigenes Referenzobjekt der Signatur (dsig:Reference in dsig:SignedInfo). Dabei ist die Verwendung des Attributs Type im Referenzobjekt zur Kennzeichnung des referenzierten Datums als Signaturmanifest erforderlich. Das Attribut muss folgenden Wert aufweisen:

<http://www.buergerkarte.at/specifications/securitylayer/20020225#SignatureManifest>

2.1.9. Signaturattribute

Die nachfolgend angegebenen Informationen müssen in die Signatur unter Verwendung von Signaturattributen nach [ETSIXML] aufgenommen werden. Die Einbindung der Signaturattribute in die Signatur muss als Direct Incorporation entsprechend den Hinweisen in den Abschnitten 6.3 und insbesondere 6.3.1 von [ETSIXML] erfolgen. Das in Abschnitt 6.3.1 erwähnte Attribut Type muss verwendet werden.

Signierte Metainformationen

Für jedes in der Anfrage spezifizierte zu signierende Datenobjekt (sl:DataObject), ist ein Signaturattribut aufgenommen werden, das die dazu spezifizierten Metainformationen (sl:FinalDataMetaInfo) enthält.

Dazu ist das signierte Signaturattribut etsi:DataObjectFormat nach [ETSIXML] zu verwenden. Das Element sl:MimeType aus sl:FinalDataMetaInfo entspricht dabei dem Element etsi:MimeType aus etsi:DataObjectFormat, das optional vorhandene Element sl:Description aus sl:FinalDataMetaInfo dem Element etsi:Description aus etsi:DataObjectFormat.

Signierte Zertifikatsreferenz

Weiters muss ein Signaturattribut mitaufgenommen werden, mit dem das für die Verifikation der Signatur zu verwendende Signatorzertifikat eindeutig identifiziert wird. Dazu ist das signierte Signaturattribut `etsi:SigningCertificate` nach [ETSIXML] zu verwenden.

Zeitpunkt der Signaturerstellung

Schließlich muss ein Signaturattribut mitaufgenommen werden, das den vom Signator behaupteten Zeitpunkt der Signaturerstellung enthält. Dazu ist das signierte Signaturattribut `etsi:SigningTime` nach [ETSIXML] zu verwenden.

Anmerkung: Für Beispiele zu diesem Befehl siehe Tutorium; für Anforderungen an die Benutzerschnittstelle siehe Anforderungen an die Benutzerschnittstelle, Abschnitt 3.1.

Anhang B – XHTML Schemata

Zur Prüfung von XHTML Inhalten werden das Schema 1 für Legacy XHTML Dokumente (vor Standard-Anzeigeformat 1.2¹⁶) und das Schema für das Standard-Anzeigeformat 1.2 zur Prüfung verwendet. Folgend die beiden Schemata.

1. SCHEMA 1 (LEGACY)

Das Schema zur XHTML Formatprüfung für Legacy Security Layer Dokumente entspricht folgendem angepaßten (dynamische Elemente wurden entfernt) „XHTML Transitional Schema“ des W3C:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema                                targetNamespace="http://www.w3.org/1999/xhtml"
xmlns:xs="http://www.w3.org/2001/XMLSchema"        xmlns="http://www.w3.org/1999/xhtml"
elementFormDefault="qualified" version="1.0" xml:lang="en">
  <xs:annotation>
    <xs:documentation>
      Standard XHTML Schema adapted by IT Solution GmbH
      XHTML 1.0 (Second Edition) Transitional in XML Schema

      This is the same as HTML 4 Transitional except for
      changes due to the differences between XML and SGML.

      Namespace = http://www.w3.org/1999/xhtml

      For further information, see: http://www.w3.org/TR/xhtml1

      Copyright (c) 1998-2002 W3C (MIT, INRIA, Keio),
      All Rights Reserved.

      The DTD version is identified by the PUBLIC and SYSTEM identifiers:

      PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      SYSTEM "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"

      $Id: xhtml1-transitional.xsd,v 1.5 2002/08/28 09:53:29 mimasa Exp $
    </xs:documentation>
  </xs:annotation>
  <xs:import                                namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.w3.org/2001/xml.xsd"/>
  <xs:annotation>
    <xs:documentation>
      ===== Character mnemonic entities =====

      XHTML entity sets are identified by the PUBLIC and SYSTEM identifiers:

      PUBLIC "-//W3C//ENTITIES Latin 1 for XHTML//EN"
      SYSTEM "http://www.w3.org/TR/xhtml1/DTD/xhtml1-lat1.ent"

      PUBLIC "-//W3C//ENTITIES Special for XHTML//EN"
      SYSTEM "http://www.w3.org/TR/xhtml1/DTD/xhtml1-special.ent"

      PUBLIC "-//W3C//ENTITIES Symbols for XHTML//EN"
      SYSTEM "http://www.w3.org/TR/xhtml1/DTD/xhtml1-symbol.ent"
    </xs:documentation>
  </xs:annotation>
  <xs:annotation>
    <xs:documentation>
      ===== Imported Names =====
    </xs:documentation>
  </xs:annotation>
  <xs:simpleType name="Character">
    <xs:annotation>
      <xs:documentation>
        a single character, as per section 2.2 of [XML]
      </xs:documentation>
    </xs:annotation>
  </xs:simpleType>
</xs:schema>
```

¹⁶ Standard-Anzeigeformat zur Bürgerkarten-Umgebung der österreichischen Bürgerkarte; siehe: <http://www.buergerkarte.at/konzept/securitylayer/spezifikation/aktuell/viewerformat/ViewerFormat.html>

```

</xs:annotation>
<xs:restriction base="xs:string">
  <xs:length value="1" fixed="true"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="Number">
  <xs:annotation>
    <xs:documentation>
      one or more digits
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:nonNegativeInteger">
    <xs:pattern value="[0-9]+"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tabindexNumber">
  <xs:annotation>
    <xs:documentation>
      tabindex attribute specifies the position of the current element
      in the tabbing order for the current document. This value must be
      a number between 0 and 32767. User agents should ignore leading zeros.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="Number">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="32767"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="URI">
  <xs:annotation>
    <xs:documentation>
      a Uniform Resource Identifier, see [RFC2396]
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:anyURI"/>
</xs:simpleType>
<xs:simpleType name="Text">
  <xs:annotation>
    <xs:documentation>
      used for titles etc.
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string"/>
</xs:simpleType>
<xs:simpleType name="Length">
  <xs:annotation>
    <xs:documentation>
      nn for pixels or nn% for percentage length
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="[\-+]?(\d+|\d+(\.\d+)?)"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="MultiLength">
  <xs:annotation>
    <xs:documentation>
      pixel, percentage, or relative
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="[\-+]?(\d+|\d+(\.\d+)?)|[1-9]?(\d+)?\%"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Pixels">
  <xs:annotation>
    <xs:documentation>
      integer representing length in pixels
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:nonNegativeInteger"/>

```

```

</xs:simpleType>
<xs:annotation>
  <xs:documentation>
    these are used for image maps
  </xs:documentation>
</xs:annotation>
<xs:simpleType name="Color">
  <xs:annotation>
    <xs:documentation>
      a color using sRGB: #RRGGBB as Hex values

      There are also 16 widely known color names with their sRGB values:

      Black = #000000      Green = #008000
      Silver = #C0C0C0     Lime = #00FF00
      Gray = #808080       Olive = #808000
      White = #FFFFFF      Yellow = #FFFF00
      Maroon = #800000     Navy = #000080
      Red = #FF0000        Blue = #0000FF
      Purple = #800080     Teal = #008080
      Fuchsia = #FF00FF   Aqua = #00FFFF
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="[A-Za-z]+|[0-9A-Fa-f]{3}|[0-9A-Fa-f]{6}"/>
  </xs:restriction>
</xs:simpleType>
<xs:annotation>
  <xs:documentation>
    ===== Generic Attributes =====
  </xs:documentation>
</xs:annotation>
<xs:attributeGroup name="TextAlign">
  <xs:annotation>
    <xs:documentation>
      text alignment for p, div, h1-h6. The default is
      align="left" for ltr headings, "right" for rtl
    </xs:documentation>
  </xs:annotation>
  <xs:attribute name="align">
    <xs:simpleType>
      <xs:restriction base="xs:token">
        <xs:enumeration value="left"/>
        <xs:enumeration value="center"/>
        <xs:enumeration value="right"/>
        <xs:enumeration value="justify"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:attributeGroup>
<xs:annotation>
  <xs:documentation>
    ===== Text Elements =====
  </xs:documentation>
</xs:annotation>
<xs:group name="special.extra">
  <xs:choice>
    <xs:element ref="img"/>
  </xs:choice>
</xs:group>
<xs:group name="special.basic">
  <xs:choice>
    <xs:element ref="br"/>
    <xs:element ref="span"/>
    <xs:element ref="bdo"/>
  </xs:choice>
</xs:group>
<xs:group name="special">
  <xs:choice>
    <xs:group ref="special.basic"/>
    <xs:group ref="special.extra"/>
  </xs:choice>
</xs:group>

```

```

    </xs:choice>
</xs:group>
<xs:group name="fontstyle.extra">
  <xs:choice>
    <xs:element ref="big"/>
    <xs:element ref="small"/>
    <xs:element ref="font"/>
    <xs:element ref="basefont"/>
  </xs:choice>
</xs:group>
<xs:group name="fontstyle.basic">
  <xs:choice>
    <xs:element ref="tt"/>
    <xs:element ref="i"/>
    <xs:element ref="b"/>
    <xs:element ref="u"/>
    <xs:element ref="s"/>
    <xs:element ref="strike"/>
  </xs:choice>
</xs:group>
<xs:group name="fontstyle">
  <xs:choice>
    <xs:group ref="fontstyle.basic"/>
    <xs:group ref="fontstyle.extra"/>
  </xs:choice>
</xs:group>
<xs:group name="phrase.extra">
  <xs:choice>
    <xs:element ref="sub"/>
    <xs:element ref="sup"/>
  </xs:choice>
</xs:group>
<xs:group name="phrase.basic">
  <xs:choice>
    <xs:element ref="em"/>
    <xs:element ref="strong"/>
    <xs:element ref="dfn"/>
    <xs:element ref="code"/>
    <xs:element ref="q"/>
    <xs:element ref="samp"/>
    <xs:element ref="kbd"/>
    <xs:element ref="var"/>
    <xs:element ref="cite"/>
    <xs:element ref="abbr"/>
    <xs:element ref="acronym"/>
  </xs:choice>
</xs:group>
<xs:group name="phrase">
  <xs:choice>
    <xs:group ref="phrase.basic"/>
    <xs:group ref="phrase.extra"/>
  </xs:choice>
</xs:group>
<xs:group name="misc.inline">
  <xs:annotation>
    <xs:documentation>
      these can only occur at block level
    </xs:documentation>
  </xs:annotation>
  <xs:choice>
    <xs:element ref="ins"/>
    <xs:element ref="del"/>
  </xs:choice>
</xs:group>
<xs:group name="misc">
  <xs:annotation>
    <xs:documentation>
      these can only occur at block level
    </xs:documentation>
  </xs:annotation>
  <xs:choice>

```

```

        <xs:group ref="misc.inline"/>
    </xs:choice>
</xs:group>
<xs:group name="inline">
    <xs:choice>
        <xs:element ref="a"/>
        <xs:group ref="special"/>
        <xs:group ref="fontstyle"/>
        <xs:group ref="phrase"/>
    </xs:choice>
</xs:group>
<xs:complexType name="Inline" mixed="true">
    <xs:annotation>
        <xs:documentation>
            "Inline" covers inline or "text-level" element
        </xs:documentation>
    </xs:annotation>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:group ref="inline"/>
        <xs:group ref="misc.inline"/>
        <xs:element ref="p"/>
    </xs:choice>
</xs:complexType>
<xs:annotation>
    <xs:documentation>
        ===== Block level elements =====
    </xs:documentation>
</xs:annotation>
<xs:group name="heading">
    <xs:choice>
        <xs:element ref="h1"/>
        <xs:element ref="h2"/>
        <xs:element ref="h3"/>
        <xs:element ref="h4"/>
        <xs:element ref="h5"/>
        <xs:element ref="h6"/>
    </xs:choice>
</xs:group>
<xs:group name="lists">
    <xs:choice>
        <xs:element ref="ul"/>
        <xs:element ref="ol"/>
        <xs:element ref="dl"/>
    </xs:choice>
</xs:group>
<xs:group name="blocktext">
    <xs:choice>
        <xs:element ref="pre"/>
        <xs:element ref="hr"/>
        <xs:element ref="blockquote"/>
        <xs:element ref="center"/>
    </xs:choice>
</xs:group>
<xs:group name="block">
    <xs:choice>
        <xs:element ref="p"/>
        <xs:group ref="heading"/>
        <xs:element ref="div"/>
        <xs:group ref="lists"/>
        <xs:group ref="blocktext"/>
        <xs:element ref="fieldset"/>
        <xs:element ref="table"/>
    </xs:choice>
</xs:group>
<xs:complexType name="Flow" mixed="true">
    <xs:annotation>
        <xs:documentation>
            "Flow" mixes block and inline and is used for list items etc.
        </xs:documentation>
    </xs:annotation>
    <xs:choice minOccurs="0" maxOccurs="unbounded">

```

```

        <xs:group ref="block"/>
        <xs:element ref="form"/>
        <xs:group ref="inline"/>
        <xs:group ref="misc"/>
    </xs:choice>
</xs:complexType>
<xs:element name="blockquote">
    <xs:complexType mixed="true">
        <xs:complexContent mixed="true">
            <xs:extension base="Flow"/>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="p">
    <xs:complexType mixed="true">
        <xs:complexContent mixed="true">
            <xs:extension base="Inline">
                <xs:attributeGroup ref="TextAlign"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="hr">
    <xs:complexType>
        <xs:attribute name="align">
            <xs:simpleType>
                <xs:restriction base="xs:token">
                    <xs:enumeration value="left"/>
                    <xs:enumeration value="center"/>
                    <xs:enumeration value="right"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="noshade">
            <xs:simpleType>
                <xs:restriction base="xs:token">
                    <xs:enumeration value="noshade"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="size" type="Pixels"/>
        <xs:attribute name="width" type="Length"/>
    </xs:complexType>
</xs:element>
<xs:annotation>
    <xs:documentation>
        ===== Content models for exclusions =====
    </xs:documentation>
</xs:annotation>
<xs:complexType name="a.content" mixed="true">
    <xs:annotation>
        <xs:documentation>
            a elements use "Inline" excluding a
        </xs:documentation>
    </xs:annotation>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:group ref="special"/>
        <xs:group ref="fontstyle"/>
        <xs:group ref="phrase"/>
        <xs:group ref="misc.inline"/>
    </xs:choice>
</xs:complexType>
<xs:complexType name="pre.content" mixed="true">
    <xs:annotation>
        <xs:documentation>
            pre uses "Inline" excluding img, object, applet, big, small,
            font, or basefont
        </xs:documentation>
    </xs:annotation>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="a"/>
    </xs:choice>
</xs:complexType>

```

```

        <xs:group ref="special.basic"/>
        <xs:group ref="fontstyle.basic"/>
        <xs:group ref="phrase.basic"/>
        <xs:group ref="misc.inline"/>
    </xs:choice>
</xs:complexType>
<xs:annotation>
    <xs:documentation>
        ===== Document Structure =====
    </xs:documentation>
</xs:annotation>
<xs:element name="html">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="head" minOccurs="0"/>
            <xs:element ref="body"/>
        </xs:sequence>
        <xs:attribute name="version" type="xs:string"/>
    </xs:complexType>
</xs:element>
<xs:annotation>
    <xs:documentation>
        ===== Document Head =====
    </xs:documentation>
</xs:annotation>
<xs:element name="head">
    <xs:annotation>
        <xs:documentation>
            content model is "head.misc" combined with a single
            title and an optional base element in any order
        </xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:sequence>
            <xs:choice>
                <xs:sequence>
                    <xs:element ref="title"/>
                </xs:sequence>
            </xs:choice>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="title" type="xs:string">
    <xs:annotation>
        <xs:documentation>
            The title element is not considered part of the flow of text.
            It should be displayed, for example as the page header or
            window title. Exactly one title is required per document.
        </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:annotation>
    <xs:documentation>
        ===== Document Body =====
    </xs:documentation>
</xs:annotation>
<xs:element name="body">
    <xs:complexType mixed="true">
        <xs:complexContent mixed="true">
            <xs:extension base="Flow">
                <xs:attribute name="background" type="URI"/>
                <xs:attribute name="bgcolor" type="Color"/>
                <xs:attribute name="text" type="Color"/>
                <xs:attribute name="link" type="Color"/>
                <xs:attribute name="vlink" type="Color"/>
                <xs:attribute name="alink" type="Color"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="div">

```

```

<xs:annotation>
  <xs:documentation>
    generic language/style container
  </xs:documentation>
</xs:annotation>
<xs:complexType mixed="true">
  <xs:complexContent mixed="true">
    <xs:extension base="Flow">
      <xs:attributeGroup ref="TextAlign"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
</xs:element>
<xs:annotation>
  <xs:documentation>
    ===== Headings =====
  
```

There are six levels of headings from h1 (the most important) to h6 (the least important).

```

  </xs:documentation>
</xs:annotation>
<xs:element name="h1">
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline">
        <xs:attributeGroup ref="TextAlign"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="h2">
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline">
        <xs:attributeGroup ref="TextAlign"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="h3">
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline">
        <xs:attributeGroup ref="TextAlign"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="h4">
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline">
        <xs:attributeGroup ref="TextAlign"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="h5">
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline">
        <xs:attributeGroup ref="TextAlign"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="h6">
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline">
        <xs:attributeGroup ref="TextAlign"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>

```

```

        </xs:extension>
    </xs:complexContent>
</xs:complexType>
</xs:element>
<xs:annotation>
    <xs:documentation>
        ===== Lists =====
    </xs:documentation>
</xs:annotation>
<xs:simpleType name="ULStyle">
    <xs:annotation>
        <xs:documentation>
            Unordered list bullet styles
        </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
        <xs:enumeration value="disc"/>
        <xs:enumeration value="square"/>
        <xs:enumeration value="circle"/>
    </xs:restriction>
</xs:simpleType>
<xs:element name="ul">
    <xs:annotation>
        <xs:documentation>
            Unordered list
        </xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="li" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="type" type="ULStyle"/>
        <xs:attribute name="compact">
            <xs:simpleType>
                <xs:restriction base="xs:token">
                    <xs:enumeration value="compact"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
</xs:element>
<xs:simpleType name="OLStyle">
    <xs:annotation>
        <xs:documentation>
            Ordered list numbering style

            1  arabic numbers      1, 2, 3, ...
            a  lower alpha       a, b, c, ...
            A  upper alpha       A, B, C, ...
            i  lower roman       i, ii, iii, ...
            I  upper roman       I, II, III, ...

            The style is applied to the sequence number which by default
            is reset to 1 for the first list item in an ordered list.
        </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
</xs:simpleType>
<xs:element name="ol">
    <xs:annotation>
        <xs:documentation>
            Ordered (numbered) list
        </xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="li" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="type" type="OLStyle"/>
        <xs:attribute name="compact">
            <xs:simpleType>

```

```

        <xs:restriction base="xs:token">
            <xs:enumeration value="compact"/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
<xs:attribute name="start" type="Number"/>
</xs:complexType>
</xs:element>
<xs:simpleType name="LIStyle">
    <xs:annotation>
        <xs:documentation>
            LIStyle is constrained to: "(ULStyle|OLStyle)"
        </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string"/>
</xs:simpleType>
<xs:element name="li">
    <xs:annotation>
        <xs:documentation>
            list item
        </xs:documentation>
    </xs:annotation>
    <xs:complexType mixed="true">
        <xs:complexContent mixed="true">
            <xs:extension base="Flow">
                <xs:attribute name="type" type="LIStyle"/>
                <xs:attribute name="value" type="Number"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:annotation>
    <xs:documentation>
        definition lists - dt for term, dd for its definition
    </xs:documentation>
</xs:annotation>
<xs:element name="dl">
    <xs:complexType>
        <xs:choice maxOccurs="unbounded">
            <xs:element ref="dt"/>
            <xs:element ref="dd"/>
        </xs:choice>
        <xs:attribute name="compact">
            <xs:simpleType>
                <xs:restriction base="xs:token">
                    <xs:enumeration value="compact"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
</xs:element>
<xs:element name="dt">
    <xs:complexType mixed="true">
        <xs:complexContent mixed="true">
            <xs:extension base="Inline"/>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="dd">
    <xs:complexType mixed="true">
        <xs:complexContent mixed="true">
            <xs:extension base="Flow"/>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:annotation>
    <xs:documentation>
        ===== Preformatted Text =====
    </xs:documentation>
</xs:annotation>
<xs:element name="pre">

```

```

<xs:annotation>
  <xs:documentation>
    content is "Inline" excluding
      "img|object|applet|big|small|sub|sup|font|basefont"
  </xs:documentation>
</xs:annotation>
<xs:complexType mixed="true">
  <xs:complexContent mixed="true">
    <xs:extension base="pre.content"/>
  </xs:complexContent>
</xs:complexType>
</xs:element>
<xs:annotation>
  <xs:documentation>
    ===== Text alignment =====
  </xs:documentation>
</xs:annotation>
<xs:element name="center">
  <xs:annotation>
    <xs:documentation>
      center content
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Flow"/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:annotation>
  <xs:documentation>
    ===== Inserted/Deleted Text =====

    ins/del are allowed in block and inline content, but its
    inappropriate to include block content within an ins element
    occurring in inline content.
  </xs:documentation>
</xs:annotation>
<xs:element name="ins">
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Flow"/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="del">
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Flow"/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:annotation>
  <xs:documentation>
    ===== The Anchor Element =====
  </xs:documentation>
</xs:annotation>
<xs:annotation>
  <xs:documentation>
    ===== Inline Elements =====
  </xs:documentation>
</xs:annotation>
<xs:element name="span">
  <xs:annotation>
    <xs:documentation>
      generic language/style container
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline"/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>

```

```

    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="bdo">
  <xs:annotation>
    <xs:documentation>
      I18N BiDi over-ride
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline">
        <xs:attribute name="dir" use="required">
          <xs:simpleType>
            <xs:restriction base="xs:token">
              <xs:enumeration value="ltr"/>
              <xs:enumeration value="rtl"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="br">
  <xs:annotation>
    <xs:documentation>
      forced line break
    </xs:documentation>
  </xs:annotation>
  <xs:complexType/>
</xs:element>
<xs:element name="em">
  <xs:annotation>
    <xs:documentation>
      emphasis
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline"/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="strong">
  <xs:annotation>
    <xs:documentation>
      strong emphasis
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline"/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="dfn">
  <xs:annotation>
    <xs:documentation>
      definitional
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline"/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="code">
  <xs:annotation>
    <xs:documentation>

```

```

    program code
  </xs:documentation>
</xs:annotation>
<xs:complexType mixed="true">
  <xs:complexContent mixed="true">
    <xs:extension base="Inline"/>
  </xs:complexContent>
</xs:complexType>
</xs:element>
<xs:element name="samp">
  <xs:annotation>
    <xs:documentation>
      sample
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline"/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="kbd">
  <xs:annotation>
    <xs:documentation>
      something user would type
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline"/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="var">
  <xs:annotation>
    <xs:documentation>
      variable
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline"/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="cite">
  <xs:annotation>
    <xs:documentation>
      citation
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline"/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="abbr">
  <xs:annotation>
    <xs:documentation>
      abbreviation
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline"/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="acronym">
  <xs:annotation>

```

```

    <xs:documentation>
    acronym
  </xs:documentation>
</xs:annotation>
<xs:complexType mixed="true">
  <xs:complexContent mixed="true">
    <xs:extension base="Inline" />
  </xs:complexContent>
</xs:complexType>
</xs:element>
<xs:element name="q">
  <xs:annotation>
    <xs:documentation>
    inlined quote
  </xs:documentation>
</xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline" />
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="sub">
  <xs:annotation>
    <xs:documentation>
    subscript
  </xs:documentation>
</xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline" />
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="sup">
  <xs:annotation>
    <xs:documentation>
    superscript
  </xs:documentation>
</xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline" />
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="tt">
  <xs:annotation>
    <xs:documentation>
    fixed pitch font
  </xs:documentation>
</xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline" />
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="i">
  <xs:annotation>
    <xs:documentation>
    italic font
  </xs:documentation>
</xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline" />
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="b">

```

```

<xs:annotation>
  <xs:documentation>
    bold font
  </xs:documentation>
</xs:annotation>
<xs:complexType mixed="true">
  <xs:complexContent mixed="true">
    <xs:extension base="Inline"/>
  </xs:complexContent>
</xs:complexType>
</xs:element>
<xs:element name="big">
  <xs:annotation>
    <xs:documentation>
      bigger font
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline"/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="small">
  <xs:annotation>
    <xs:documentation>
      smaller font
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline"/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="u">
  <xs:annotation>
    <xs:documentation>
      underline
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline"/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="s">
  <xs:annotation>
    <xs:documentation>
      strike-through
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline"/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="strike">
  <xs:annotation>
    <xs:documentation>
      strike-through
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline"/>
    </xs:complexContent>
  </xs:complexType>
</xs:element>

```

```

<xs:element name="basefont">
  <xs:annotation>
    <xs:documentation>
      base font size
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:attribute name="size" use="required"/>
    <xs:attribute name="color" type="Color"/>
    <xs:attribute name="face"/>
  </xs:complexType>
</xs:element>
<xs:element name="font">
  <xs:annotation>
    <xs:documentation>
      local change to font
    </xs:documentation>
  </xs:annotation>
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline">
        <xs:attribute name="size"/>
        <xs:attribute name="color" type="Color"/>
        <xs:attribute name="face"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:annotation>
  <xs:documentation>
    ===== Images =====

    To avoid accessibility problems for people who aren't
    able to see the image, you should provide a text
    description using the alt and longdesc attributes.
    In addition, avoid the use of server-side image maps.
  </xs:documentation>
</xs:annotation>
<xs:simpleType name="ImgAlign">
  <xs:annotation>
    <xs:documentation>
      used for object, applet, img, input and iframe
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:token">
    <xs:enumeration value="top"/>
    <xs:enumeration value="middle"/>
    <xs:enumeration value="bottom"/>
    <xs:enumeration value="left"/>
    <xs:enumeration value="right"/>
  </xs:restriction>
</xs:simpleType>
<xs:element name="img">
  <xs:complexType>
    <xs:attribute name="src" type="URI" use="required"/>
    <xs:attribute name="height" type="Length"/>
    <xs:attribute name="width" type="Length"/>
    <xs:attribute name="border" type="Length"/>
    <xs:attribute name="hspace" type="Pixels"/>
    <xs:attribute name="vspace" type="Pixels"/>
    <xs:attribute name="align" type="ImgAlign"/>
  </xs:complexType>
</xs:element>
<xs:annotation>
  <xs:documentation>
    ===== Tables =====

    Derived from IETF HTML table standard, see [RFC1942]
  </xs:documentation>
</xs:annotation>
<xs:simpleType name="TFrame">

```

```

<xs:annotation>
  <xs:documentation>
    The border attribute sets the thickness of the frame around the
    table. The default units are screen pixels.

    The frame attribute specifies which parts of the frame around
    the table should be rendered. The values are not the same as
    CALS to avoid a name clash with the valign attribute.
  </xs:documentation>
</xs:annotation>
<xs:restriction base="xs:token">
  <xs:enumeration value="void"/>
  <xs:enumeration value="above"/>
  <xs:enumeration value="below"/>
  <xs:enumeration value="hsides"/>
  <xs:enumeration value="lhs"/>
  <xs:enumeration value="rhs"/>
  <xs:enumeration value="vsides"/>
  <xs:enumeration value="box"/>
  <xs:enumeration value="border"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="TRules">
  <xs:annotation>
    <xs:documentation>
      The rules attribute defines which rules to draw between cells:

      If rules is absent then assume:
        "none" if border is absent or border="0" otherwise "all"
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:token">
    <xs:enumeration value="none"/>
    <xs:enumeration value="groups"/>
    <xs:enumeration value="rows"/>
    <xs:enumeration value="cols"/>
    <xs:enumeration value="all"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="TAlign">
  <xs:annotation>
    <xs:documentation>
      horizontal placement of table relative to document
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:token">
    <xs:enumeration value="left"/>
    <xs:enumeration value="center"/>
    <xs:enumeration value="right"/>
  </xs:restriction>
</xs:simpleType>
<xs:attributeGroup name="cellhalign">
  <xs:annotation>
    <xs:documentation>
      horizontal alignment attributes for cell contents

      char          alignment char, e.g. char=':'
      charoff       offset for alignment char
    </xs:documentation>
  </xs:annotation>
  <xs:attribute name="align">
    <xs:simpleType>
      <xs:restriction base="xs:token">
        <xs:enumeration value="left"/>
        <xs:enumeration value="center"/>
        <xs:enumeration value="right"/>
        <xs:enumeration value="justify"/>
        <xs:enumeration value="char"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>

```

```

    <xs:attribute name="char" type="Character"/>
    <xs:attribute name="charoff" type="Length"/>
</xs:attributeGroup>
<xs:attributeGroup name="cellvalign">
  <xs:annotation>
    <xs:documentation>
      vertical alignment attributes for cell contents
    </xs:documentation>
  </xs:annotation>
  <xs:attribute name="valign">
    <xs:simpleType>
      <xs:restriction base="xs:token">
        <xs:enumeration value="top"/>
        <xs:enumeration value="middle"/>
        <xs:enumeration value="bottom"/>
        <xs:enumeration value="baseline"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:attributeGroup>
<xs:element name="table">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="caption" minOccurs="0"/>
      <xs:choice>
        <xs:element ref="col" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="colgroup" minOccurs="0" maxOccurs="unbounded"/>
      </xs:choice>
      <xs:element ref="thead" minOccurs="0"/>
      <xs:element ref="tfoot" minOccurs="0"/>
      <xs:choice>
        <xs:element ref="tbody" maxOccurs="unbounded"/>
        <xs:element ref="tr" maxOccurs="unbounded"/>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="width" type="Length"/>
    <xs:attribute name="border" type="Pixels"/>
    <xs:attribute name="frame" type="TFrame"/>
    <xs:attribute name="rules" type="TRules"/>
    <xs:attribute name="cellspacing" type="Length"/>
    <xs:attribute name="cellpadding" type="Length"/>
    <xs:attribute name="align" type="TAlign"/>
    <xs:attribute name="bgcolor" type="Color"/>
  </xs:complexType>
</xs:element>
<xs:simpleType name="CAAlign">
  <xs:restriction base="xs:token">
    <xs:enumeration value="top"/>
    <xs:enumeration value="bottom"/>
    <xs:enumeration value="left"/>
    <xs:enumeration value="right"/>
  </xs:restriction>
</xs:simpleType>
<xs:element name="caption">
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Inline">
        <xs:attribute name="align" type="CAAlign"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:annotation>
  <xs:documentation>
    Use thead to duplicate headers when breaking table
    across page boundaries, or for static headers when
    tbody sections are rendered in scrolling panel.
  </xs:documentation>

```

Use tfoot to duplicate footers when breaking table across page boundaries, or for static footers when tbody sections are rendered in scrolling panel.

Use multiple tbody sections when rules are needed between groups of table rows.

```
</xs:documentation>
</xs:annotation>
<xs:element name="thead">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="tr" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attributeGroup ref="cellhalign"/>
    <xs:attributeGroup ref="cellvalign"/>
  </xs:complexType>
</xs:element>
<xs:element name="tfoot">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="tr" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attributeGroup ref="cellhalign"/>
    <xs:attributeGroup ref="cellvalign"/>
  </xs:complexType>
</xs:element>
<xs:element name="tbody">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="tr" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attributeGroup ref="cellhalign"/>
    <xs:attributeGroup ref="cellvalign"/>
  </xs:complexType>
</xs:element>
<xs:element name="colgroup">
  <xs:annotation>
    <xs:documentation>
      colgroup groups a set of col elements. It allows you to group
      several semantically related columns together.
    </xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="col" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="span" type="Number"/>
    <xs:attribute name="width" type="MultiLength"/>
    <xs:attributeGroup ref="cellhalign"/>
    <xs:attributeGroup ref="cellvalign"/>
  </xs:complexType>
</xs:element>
<xs:element name="col">
  <xs:annotation>
    <xs:documentation>
      col elements define the alignment properties for cells in
      one or more columns.
```

The width attribute specifies the width of the columns, e.g.

width=64	width in screen pixels
width=0.5*	relative width of 0.5

The span attribute causes the attributes of one col element to apply to more than one column.

```
</xs:documentation>
</xs:annotation>
<xs:complexType>
  <xs:attribute name="span" type="Number"/>
  <xs:attribute name="width" type="MultiLength"/>
  <xs:attributeGroup ref="cellhalign"/>
  <xs:attributeGroup ref="cellvalign"/>
</xs:complexType>
</xs:element>
```

```

<xs:element name="tr">
  <xs:complexType>
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="th"/>
      <xs:element ref="td"/>
    </xs:choice>
    <xs:attributeGroup ref="cellhalign"/>
    <xs:attributeGroup ref="cellvalign"/>
    <xs:attribute name="bgcolor" type="Color"/>
  </xs:complexType>
</xs:element>
<xs:annotation>
  <xs:documentation>
    th is for headers, td for data and for cells acting as both
  </xs:documentation>
</xs:annotation>
<xs:element name="th">
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Flow">
        <xs:attribute name="rowspan" type="Number"/>
        <xs:attribute name="colspan" type="Number"/>
        <xs:attributeGroup ref="cellhalign"/>
        <xs:attributeGroup ref="cellvalign"/>
        <xs:attribute name="nowrap">
          <xs:simpleType>
            <xs:restriction base="xs:token">
              <xs:enumeration value="nowrap"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="bgcolor" type="Color"/>
        <xs:attribute name="width" type="Length"/>
        <xs:attribute name="height" type="Length"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
<xs:element name="td">
  <xs:complexType mixed="true">
    <xs:complexContent mixed="true">
      <xs:extension base="Flow">
        <xs:attribute name="text" type="Color"/>
        <xs:attribute name="rowspan" type="Number"/>
        <xs:attribute name="colspan" type="Number"/>
        <xs:attributeGroup ref="cellhalign"/>
        <xs:attributeGroup ref="cellvalign"/>
        <xs:attribute name="nowrap">
          <xs:simpleType>
            <xs:restriction base="xs:token">
              <xs:enumeration value="nowrap"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="bgcolor" type="Color"/>
        <xs:attribute name="width" type="Length"/>
        <xs:attribute name="height" type="Length"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
</xs:schema>

```

2. SCHEMA 2 (SLXHTML12)

Das Schema zur XHTML/CSS2 Formatprüfung entspricht der Spezifikation 1.2 des Viewerformates zum Security Layer:

(Auszug aus Spezifikation Standard-Anzeigeformat zur Bürgerkarten-Umgebung der österreichischen Bürgerkarte Version 1.2.1 vom 01.03.2005)

2.1. Profil von XHMTL 1.1

Dieser Abschnitt beschreibt das Profil für die XML-Struktur des Standard-Anzeigeformats. Der Aufbau dieses Abschnitts orientiert sich an den in [\[XHMTL 1.1\]](#), Abschnitt 5 definierten Modulen: Für jedes Modul werden die Einschränkungen hinsichtlich der im Standard-Anzeigeformat erlaubten Elemente und Attribute angegeben.

Die zum Abschluß dieses Abschnitts referenzierten XML-Schemata dienen als normative Zusammenfassung der Einschränkungen. Sie orientiert sich an dem in [\[XHMTL MOD\]](#) vorgeschlagenen Mechanismus zur Erstellung eines XHMTL-Dialekts. Instanzdokumente, die nicht extakt diesem Schema entsprechen, müssen von einer [Bürgerkarten-Umgebung](#) zurückgewiesen werden.

2.1.1. Einschränkungen gegenüber XHMTL 1.1

Attribute Collections

Vergleiche [\[XHMTL 1.1\]](#), Abschnitt 5.1.

Aus der Attributsammlung `Core` wird das Attribut `title` gestrichen. Die Attributsammlungen `Events`, `Style` und `I18N` werden zur Gänze entleert. Die Attributsammlung `Common` ist damit ident mit `Core`.

Core Modules

Vergleiche [\[XHMTL 1.1\]](#), Abschnitt 5.2.

Structure Module

Das Attribut `profile` zum Element `head` wird gestrichen. Das Inhaltsmodell des Elements `head` wird auf `(title, style)` verändert.

Das Attribut `version` zum Element `html` wird auf den Wert `//www.buergerkarte.at//DOCUMENT SLXHTML 1.2//DE` fixiert.

Die Elemente `body` und `title` bleiben sowohl hinsichtlich Inhaltsmodell als auch hinsichtlich der Attribute unverändert.

Text Module

Das Attribut `cite` zum Element `blockquote` wird gestrichen.

Die Elemente `br`, `code`, `div`, `em`, `h1`, `h2`, `h3`, `h4`, `h5`, `h6`, `p`, `pre`, `span` und `strong` bleiben sowohl hinsichtlich Inhaltsmodell als auch hinsichtlich der Attribute unverändert.

Die Content Sets `Heading` und `Flow` bleiben unverändert; der Content Set `Block` ändert sich zu `(blockquote|div|p|pre)`; der Content Set `Inline` ändert sich zu `(br|cite|code|em|span|strong)`. Damit werden die Elemente `abbr`, `acronym`, `address`, `dnf`, `kbd`, `q`, `samp` und `var` nicht verwendet.

Hypertext Module

Dieses Modul wird nicht verwendet.

List Module

Dieses Modul wird unverändert verwendet.

Applet Module

Dieses Modul wird nicht verwendet.

Text Extension Module

Presentation Module

Das Element `hr` bleibt sowohl hinsichtlich Inhaltsmodell als auch hinsichtlich der Attribute unverändert. Die Elemente `tt`, `i`, `b`, `big`, `small`, `sup`, `sub` werden nicht verwendet.

Der Content Set `Block` wird somit um das Element `hr` erweitert.

Edit Module

Dieses Modul wird nicht verwendet.

Bi-directional Text Module

Dieses Modul wird nicht verwendet.

Forms Modules

Diese Module werden nicht verwendet.

Table Modules

Basic Table Module

Dieses Modul wird nicht verwendet.

Table Module

Das Inhaltsmodell bleibt für alle Elemente unverändert.

Die Attribute `summary`, `width`, `border`, `cellpadding`, `cellspacing`, `datapagesize` sowie die Attributgruppen `frame` und `rules` zum Element `table` werden gestrichen.

Die Attribute `abbr`, `axis`, `headers`, sowie die Attributgruppen `scope`, `CellHAlign` und `CellVAlign` zu den Elementen `th` und `td` werden gestrichen.

Die Attributgruppen `CellHAlign` und `CellVAlign` zu den Elementen `tr`, `thead`, `tfoot` und `tbody` werden gestrichen.

Die Attribute `span` und `width` sowie die Attributgruppen `CellHAlign` und `CellVAlign` zu den Elementen `col` und `colgroup` werden gestrichen.

Mit den Attributen `colspan` und `rowspan` zu den Elementen `th` und `td` ist es möglich, überlappende Tabellenbereiche zu erzeugen. Enthält ein Instanzdokument eine Tabelle mit einem solchen überlappenden Bereich, MUSS das Instanzdokument von der Anzeigekomponente zurückwiesen werden.

Image Module

Die Attribute `longdesc`, `height` und `width` zum Element `img` werden gestrichen. Das Inhaltsmodell für das Element `img` bleibt unverändert.

Referenziert ein Instanzdokument mittels des Elements `img` ein oder mehrere Bilder, so MUSS die [Bürgerkarten-Umgebung](#) im Fall der Signaturerstellung für jedes Bild eine der beiden folgenden Möglichkeiten wählen:

- Die [Bürgerkarten-Umgebung](#) zeigt das referenzierte Bild als Teil des Instanzdokuments an und inkludiert eine Referenz auf das Bild entsprechend der Methode in [Abschnitt 4](#) in die Signatur.
- Die [Bürgerkarten-Umgebung](#) zeigt nicht das referenzierte Bild als Teil des Instanzdokuments an, sondern stattdessen den Text des Attributs `alt`. In diesem Fall wird die entsprechende Bildreferenz jedoch nicht in die Signatur inkludiert.

Anmerkung: Die zweite Variante kann von der [Bürgerkarten-Umgebung](#) gewählt werden, wenn sie das referenzierte Bild nicht auflösen kann (z.B. Broken Link), oder aber wenn die Darstellung von Bildern im Darstellungskontext keinen Sinn ergibt (z.B. bei einer [Bürgerkarten-Umgebung](#) für Blinde).

Im Fall der Signaturprüfung MUSS die [Bürgerkarten-Umgebung](#) für jedes Bild, das in einem Instanzdokument referenziert werden, wie folgt vorgehen:

- Wurde eine Referenz auf das Bild in die zu prüfende Signatur entsprechend dem [Abschnitt 4](#) inkludiert, MUSS die [Bürgerkarten-Umgebung](#) dieses Bild als Teil des Instanzdokuments darstellen. Ist dies nicht möglich (z.B. weil Auflösung scheitert, die aufgelösten Daten nicht interpretiert werden können, oder die Darstellung von Bildern nicht durchführbar ist - [Bürgerkarten-Umgebung](#) für Blinde), MUSS sie das Instanzdokument zurückweisen. Das Attribut `alt` zum Element `img` DARF in diesem Fall NICHT angezeigt werden.
- Fehlt für ein referenziertes Bild hingegen die korrespondierende Referenz entsprechend dem [Abschnitt 4](#) in der zu prüfenden Signatur, DARF die [Bürgerkarten-Umgebung](#) dieses Bild NICHT als Teil des Instanzdokuments darstellen. Stattdessen MUSS sie den Text des Attributs `alt` darstellen.

Das Bildformat [[JPEG](#)] MUSS von der Anzeigekomponente einer [Bürgerkarten-Umgebung](#) unterstützt werden. Enthält eine [[JPEG](#)]-Datei jedoch Marker vom Typ `TEM`, `JPG`, `JPGn` ($n \geq 0$), `RSTn` ($n \geq 0$) oder `APPn` ($n > 0$), MUSS das Instanzdokument von der [Bürgerkarten-Umgebung](#) zurückgewiesen werden.

Das Bildformat [\[GIF\]](#) MUSS von der Anzeigekomponente einer [Bürgerkarten-Umgebung](#) ebenfalls unterstützt werden. Enthält eine [\[GIF\]](#)-Datei jedoch mehrere Bilder (*Animated GIF*) oder eine *Application Extension*, MUSS das Instanzdokument von der [Bürgerkarten-Umgebung](#) zurückgewiesen werden. Eine [\[GIF\]](#)-Datei DARF Erweiterungen vom Typ *Comment Extension* sowie *Plain Text Extension* beinhalten, der Inhalt dieser Erweiterungen DARF jedoch NICHT angezeigt werden.

Weitere Bildformate DÜRFEN von der Anzeigekomponente einer [Bürgerkarten-Umgebung](#) unterstützt werden. Die Bürgerkarten-Umgebung MUSS dabei jedoch verhindern, dass dynamischen Inhalte angezeigt werden (z.B. durch animierte Bilder) .

Client-side Image Map Module

Dieses Modul wird nicht verwendet.

Server-side Image Map Module

Dieses Modul wird nicht verwendet.

Object Module

Dieses Modul wird nicht verwendet.

Frames Module

Dieses Modul wird nicht verwendet.

Target Module

Dieses Modul wird nicht verwendet.

Iframe Module

Dieses Modul wird nicht verwendet.

Intrinsic Events Module

Dieses Modul wird nicht verwendet.

Metainformation Module

Dieses Modul wird nicht verwendet.

Scripting Module

Dieses Modul wird nicht verwendet.

Style Sheet Module

Die Attribute `title` und `xml:space` zum Element `style` werden gestrichen. Das Attribut `type` zum Element `style` wird vorgeschrieben und auf den Wert `text/css` fixiert. Das Attribut `media` zum Element `style` wird ebenfalls vorgeschrieben und auf den Wert `screen` fixiert. Das Inhaltsmodell für das Element `style` bleibt unverändert.

Anmerkung: Für den erlaubten Inhalt des Elements `style` siehe [Abschnitt 3](#).

Style Attribute Module

Dieses Modul wird nicht verwendet.

Link Module

Dieses Modul wird nicht verwendet.

Base Module

Dieses Modul wird nicht verwendet.

Name Identification Module

Dieses Modul wird nicht verwendet.

Legacy Module

Dieses Modul wird nicht verwendet.

2.1.2. XML-Schema für das Standard-Anzeigeformat

Die im vorigen Abschnitt angegebenen Einschränkungen sind als XML-Schema zusammengefasst und formalisiert. Dieses XML-Schema ist als normativ anzusehen und hat bei Unklarheiten Vorrang gegenüber dieser textuellen Beschreibung. Das XML-Schema besteht aus einer Vielzahl von einzelnen Dateien und kann in gepackter Form von folgender Adresse heruntergeladen werden:

[\[slxhtml.schemas.zip\]](#)

Die Hauptschema-Datei kann weiters direkt von folgender Adresse geladen werden:

[\[schema/slxhtml.xsd\]](#)

2.1.3. Sonstiges

Kommentare

Kommentare sind ein Bestandteil von [\[XHTML 1.1\]](#) mit einer klar festgelegten Bedeutung. Aus dieser Bedeutung geht hervor, dass es sich dabei um Informationen handelt, die nicht für die Anzeige vorgesehen sind. Aus diesem Grund DARF ein Instanzdokument Kommentare enthalten. Die Kommentare DÜRFEN von der Anzeigekomponente einer [Bürgerkarten-Umgebung](#) NICHT angezeigt werden.

Rendering außerhalb der Zeichenebene

Bestandteile eines Instanzdokuments, die zwar laut [\[XHTML 1.1\]](#) von einer Anzeige dargestellt werden dürfen, für die aber kein verpflichtendes Rendering auf der Zeichenebene vorgesehen ist, DÜRFEN von der Anzeigekomponente einer [Bürgerkarten-Umgebung](#) NICHT angezeigt werden. Als Beispiel sei hier der Inhalt des Elements `title` ([\[XHTML 1.1\]](#) schlägt eine Anzeige in der Titelleiste des Anzeigefensters vor) erwähnt.

2.2. Profil von CSS 2

Das in diesem Abschnitt angegebene Profil von [\[CSS 2\]](#) legt zunächst jenes Mindestmaß der CSS2-Syntax fest, die von jeder [Bürgerkarten-Umgebung](#) im Rahmen der Anzeige eines Dokuments, das diesem Standard-Anzeigeformat entspricht, verarbeitet werden können MUSS bzw. SOLLTE.

Darüber hinaus DARF jede [Bürgerkarten-Umgebung](#) auch die nicht in diesem Profil angegebenen syntaktischen Konstrukte aus CSS2 verarbeiten und zur Anzeige bringen, wenn dies in den folgenden Abschnitten nicht ausdrücklich untersagt wird.

Kann eine [Bürgerkarten-Umgebung](#) syntaktische Konstrukte, die in dieser Spezifikation als OPTIONAL oder EMPFOHLEN qualifiziert sind, nicht interpretieren, MUSS sie das entsprechende Instanzdokument zurückweisen.

Anwendungsentwickler, die Instanzdokumente schreiben, welche diesem Standard-Anzeigeformat genügen, SOLLTEN nur die in diesem Profil als ERFORDERLICH oder EMPFOHLEN bezeichneten syntaktischen Konstrukte verwenden.

2.2.1. Einbindung von CSS-Formaten in das Standard-Anzeigeformat

[\[CSS 2\]](#) definiert eine Reihe von Möglichkeiten, wie CSS-Formate in eine Instanz eines XHTML-Dokuments eingebunden werden können. Dieses Profil beschränkt diese Möglichkeiten auf eine einzige, und zwar die Einbindung mittels des XHTML-Elements `style`. Alle anderen Möglichkeiten (Definition über das XHTML-Attribut `style`, sowie die Einbindung externer CSS-Dateien über das XHTML-Element `link` bzw. über die CSS-Regel `@import`) DARF eine [Bürgerkarten-Umgebung](#) NICHT verarbeiten, d. h. das entsprechende Instanzdokument MUSS zurückgewiesen werden.

2.2.2. Anwendung von CSS-Formaten durch die Bürgerkarten-Umgebung

Die Bestimmung der von der Anzeige-Komponente der [Bürgerkarten-Umgebung](#) zu verwendenden CSS-Formate erfolgt in zwei Schritten:

- Als Ausgangspunkt müssen die Formate des Default-CSS-Stylesheets (vergleiche [Abschnitt 3.2.1](#)) verwendet werden.
- Existieren im Instanzdokument benutzerdefinierte CSS-Formate (vergleiche [Abschnitt 3.1](#)), MÜSSEN diese benutzerdefinierten Formate jene des Default-CSS-Stylesheets überschreiben.

Die aus diesen zwei Schritten resultierenden CSS-Formate MÜSSEN von der Anzeigekomponente der [Bürgerkarten-Umgebung](#) für die Anzeige des Instanzdokuments berücksichtigt werden.

Default-CSS-Stylesheet

Der als Ausgangspunkt zur Bestimmung der von der [Bürgerkarten-Umgebung](#) bei der Anzeige anzuwendenden CSS-Formate zu verwendende Default-CSS-Stylesheet ist normativer Bestandteil dieser Spezifikation und kann unter folgender Adresse heruntergeladen werden:

[\[slxhtml.default.css\]](#)

Anmerkung: Es ist nicht notwendig, dass die Anzeigekomponente der [Bürgerkarten-Umgebung](#) den Default-Stylesheet interpretieren können muss. Vielmehr *MUSS* sie dafür Sorge tragen, dass das angezeigte Instanzdokument so aussieht, als sei es unter Interpretation des Default-Stylesheets entstanden. Insofern ist es kein Widerspruch, dass im Default-Stylesheet CSS-Eigenschaften und Eigenschaftswerte verwendet werden, die laut dieser Spezifikation nicht verpflichtend unterstützt werden müssen.

2.2.3. @-Regeln

Hinsichtlich der in [\[CSS 2\]](#) definierten @-Regeln gelten für eine [Bürgerkarten-Umgebung](#) folgende Regeln:

@charset

Die @charset Regel zur Angabe der Zeichenkodierung eines externen Stylesheets DARF in einem Instanzdokument NICHT verwendet werden, da diese Spezifikation ausschließlich die Verwendung von eingebetteten Stylesheets erlaubt (siehe [Abschnitt 3.1](#)), und die @charset Regel in solchen Stylesheets NICHT verwendet werden DARF (vgl. [\[CSS 2\]](#), Abschnitt 4.4). Die Anzeigekomponente der [Bürgerkarten-Umgebung](#) MUSS ein Instanzdokument, das eine @charset Regel enthält, zurückweisen.

@import

Die @import Regel zur Einbindung externer CSS-Dateien DARF in einem Instanzdokument NICHT verwendet werden (vgl. [\[CSS 2\]](#), Abschnitt 6.3, sowie [Abschnitt 3.1](#) dieses Dokuments). Die Anzeigekomponente der [Bürgerkarten-Umgebung](#) MUSS ein Instanzdokument, das eine @import Regel enthält, zurückweisen.

@media

Die @media Regel zur Angabe von Stylesheet-Information für bestimmte Ausgabemedien DARF in einem Instanzdokument NICHT verwendet werden (vgl. [\[CSS 2\]](#), Abschnitt 7). Die Anzeigekomponente der [Bürgerkarten-Umgebung](#) MUSS ein Instanzdokument, das eine @media Regel enthält, zurückweisen.

@page

Die @page Regel zur Festlegung der Seiteneigenschaften für seitenorientierte Ausgabemedien DARF von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [\[CSS 2\]](#), Abschnitt 13, sowie [Abschnitt 3.5.7](#) dieses Dokuments).

@font-face

Die @font-face Regel zur Beschreibung bzw. Referenzierung zusätzlicher Schriftfamilien (vgl. [\[CSS 2\]](#), Abschnitt 15.3) DARF in einem Instanzdokument NICHT verwendet werden. Die

Anzeigekomponente der [Bürgerkarten-Umgebung](#) MUSS ein Instanzdokument, das eine `@font-face` Regel enthält, zurückweisen.

2.2.4. CSS-Selektoren

[[CSS 2](#)] definiert in Abschnitt 5 eine Reihe von Regeln für den Aufbau von Selektoren. Dieser Abschnitt legt fest, welche dieser Regeln eine [Bürgerkarten-Umgebung](#) verarbeiten können MUSS; und zwar sind das:

- universal selectors
- type selectors
- class selectors
- ID selectors

Die übrigen Selektoren DÜRFEN unterstützt werden:

- descendant selectors
- child selectors
- adjacent selectors
- *attribute selectors* (Ausnahme: *class selectors*)
- pseudo classes
- pseudo elements

2.2.5. CSS-Eigenschaften

Wertangaben

Dieser Abschnitt legt die generellen Anforderungen hinsichtlich der zu unterstützenden Wertangaben für CSS-Eigenschaften fest. Sie gelten für alle anwendbaren CSS-Eigenschaften, außer es werden für eine spezielle CSS-Eigenschaft explizit andere Angaben gemacht.

Längenangaben

Eine [Bürgerkarten-Umgebung](#) MUSS eine Längenangabe in den Einheiten `in`, `mm`, `cm`, `pc` und `px` und SOLLTE eine Längenangabe in den relativen Einheiten `ex` und `em` für eine CSS-Eigenschaft unterstützen, wenn laut [[CSS 2](#)] eine solche für diese Eigenschaft möglich ist.

Prozentwerte

Eine [Bürgerkarten-Umgebung](#) MUSS eine prozentuelle Wertangabe für eine CSS-Eigenschaft unterstützen, wenn laut [[CSS 2](#)] eine solche für diese Eigenschaft möglich ist.

Farben

Eine [Bürgerkarten-Umgebung](#) MUSS für eine CSS-Eigenschaft sämtliche in [[CSS 2](#)], Abschnitt 4.3.6 aufgeführten Möglichkeiten zur Angabe einer Farbe unterstützen, wenn laut [[CSS 2](#)] eine solche für diese Eigenschaft möglich ist.

Die Ausnahme bilden die Systemfarben (vgl. [[CSS 2](#)], Abschnitt 18.2); diese DÜRFEN in einem Instanzdokument nicht verwendet werden, um Abhängigkeiten von den Systemumgebung auszuschließen. Widrigenfalls MUSS das Instanzdokument von der [Bürgerkarten-Umgebung](#) zurückgewiesen werden.

Eerbe und automatische Werte

Für Eigenschaften, die laut dieser Spezifikation von einer [Bürgerkarten-Umgebung](#) unterstützt werden MÜSSEN oder SOLLTEN, SOLLTEN jedenfalls auch die Eigenschaftswerte `inherit` bzw. `auto` unterstützt werden, wenn laut [[CSS 2](#)] solche Werte für die Eigenschaft möglich sind.

Abstände und Rahmen

Randabstände

Die Eigenschaften `margin-top`, `margin-bottom`, `margin-left` und `margin-right` MÜSSEN von einer [Bürgerkarten-Umgebung](#) unterstützt werden. Prozentuell angegebene Werte (vgl. [Abschnitt 3.5.1.2](#)) SOLLTEN unterstützt werden.

Die Eigenschaft `margin` DARF von einer [Bürgerkarten-Umgebung](#) unterstützt werden.

Ein Instanzdokument DARF in den oben genannten Eigenschaften NICHT negative Wert enthalten. Widrigenfalls MUSS es von der [Bürgerkarten-Umgebung](#) zurückgewiesen werden.

Die Eigenschaften `padding-top`, `padding-bottom`, `padding-left` und `padding-right` MÜSSEN von einer [Bürgerkarten-Umgebung](#) unterstützt werden. Prozentuell angegebene Werte (vgl. [Abschnitt 3.5.1.2](#)) SOLLTEN unterstützt werden.

Die Eigenschaft `padding` DARF von einer [Bürgerkarten-Umgebung](#) unterstützt werden.

Ein Instanzdokument DARF in den oben genannten Eigenschaften NICHT negative Werte enthalten. Widrigenfalls MUSS es von der [Bürgerkarten-Umgebung](#) zurückgewiesen werden.

Rahmendicke

Die Eigenschaften `border-top-width`, `border-bottom-width`, `border-left-width`, `border-right-width` und `border-width` SOLLTEN von einer [Bürgerkarten-Umgebung](#) unterstützt werden. Werden die Eigenschaften unterstützt, SOLLTEN auch die vordefinierten Werte `thin`, `medium` und `thick` unterstützt werden (vgl. [\[CSS 2\]](#), Abschnitt 8.5.1).

Rahmenfarbe

Die Eigenschaften `border-top-color`, `border-bottom-color`, `border-left-color`, `border-right-color` und `border-color` SOLLTEN von einer [Bürgerkarten-Umgebung](#) unterstützt werden. Der vordefinierte Wert `transparent` für die Eigenschaft `border-color` DARF unterstützt werden (vgl. [\[CSS 2\]](#), Abschnitt 8.5.2).

Rahmenstil

Die Eigenschaften `border-top-style`, `border-bottom-style`, `border-left-style`, `border-right-style` und `border-style` SOLLTEN von einer [Bürgerkarten-Umgebung](#) unterstützt werden. Werden die Eigenschaften unterstützt, SOLLTEN die vordefinierten Werte `none`, `dashed`, `dotted`, `solid` und `double` unterstützt werden; alle übrigen Werte DÜRFEN unterstützt werden (vgl. [\[CSS 2\]](#), Abschnitt 8.5.3).

Kurze Schreibweise

Die Eigenschaften für die verkürzte Schreibweise der Rahmeneigenschaften (`border-top`, `border-bottom`, `border-left`, `border-right` und `border` - vgl. [\[CSS 2\]](#), Abschnitt 8.5.4) SOLLTEN von einer [Bürgerkarten-Umgebung](#) unterstützt werden. Die EMPFOHLENEN Werte ergeben sich aus den drei vorigen Abschnitten.

Positionierung von Boxen

Die Eigenschaft zur Steuerung des Box-Typs (`display`) DARF von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [\[CSS 2\]](#), Abschnitt 9.2).

Die Eigenschaft zur Festlegung des Positionierungsschemas für eine Box (`position`, vgl. [\[CSS 2\]](#), Abschnitt 9.3) DARF in einem Instanzdokument NICHT enthalten sein, um Überlappungen von Inhalten auszuschließen. Widrigenfalls MUSS das Instanzdokument von der [Bürgerkarten-Umgebung](#) zurückgewiesen werden.

Die Eigenschaften zur Festlegung der Abstände einer Box (`top`, `bottom`, `left`, `right`; vgl. [\[CSS 2\]](#), Abschnitt 9.3) DÜRFEN in einem Instanzdokument NICHT enthalten sein, um Überlappungen von Inhalten auszuschließen. Widrigenfalls MUSS das Instanzdokument von der [Bürgerkarten-Umgebung](#) zurückgewiesen werden.

Anmerkung: Es wäre nicht unbedingt notwendig, diese Eigenschaften explizit zu verbieten, da sie nur für Elemente anzuwenden ist, die positioniert sind, für die also auch die Eigenschaft `position` gesetzt ist. Es ist jedoch auch nicht sinnvoll, diese Eigenschaften im Stylesheet des Instanzdokuments quasi als Zombi auftreten zu lassen.

Die Eigenschaften zur Festlegung des Umfließens von Boxen (`float`, `clear`) DÜRFEN von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [\[CSS 2\]](#), Abschnitt 9.5).

Die Eigenschaft zur Festlegung der Anordnung von Boxen in der Tiefe (`z-index`; vgl. [\[CSS 2\]](#), Abschnitt 9.9) DARF in einem Instanzdokument NICHT enthalten sein, um Überlappungen von Inhalten auszuschließen. Widrigenfalls MUSS das Instanzdokument von der [Bürgerkarten-Umgebung](#) zurückgewiesen werden.

Anmerkung: Es wäre nicht unbedingt notwendig, diese Eigenschaft explizit zu verbieten, da sie nur für Elemente anzuwenden ist, die positioniert sind, für die also auch die Eigenschaft `position` gesetzt ist. Es ist jedoch auch nicht sinnvoll, diese Eigenschaft im Stylesheet des Instanzdokuments quasi als Zombi auftreten zu lassen.

Die Eigenschaften zur Steuerung der Textrichtung (`direction`, `unicode-bidi`) DÜRFEN von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [\[CSS 2\]](#), Abschnitt 9.10).

Darstellung von Boxen

Die Eigenschaften zur Angabe von Breite und Höhe einer Box (`width`, `height`, vgl. [\[CSS 2\]](#), Abschnitte 10.2 und 10.5) DÜRFEN von einer [Bürgerkarten-Umgebung](#) NICHT unterstützt werden, um Überlappungen von Inhalten auszuschließen. Widrigenfalls MUSS das Instanzdokument von der [Bürgerkarten-Umgebung](#) zurückgewiesen werden.

Die Eigenschaften zur Angabe der minimalen Breite und Höhe einer Box (`min-width`, `min-height`) DÜRFEN von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [\[CSS 2\]](#), Abschnitte 10.4 und 10.7).

Die Eigenschaften zur Angabe der maximalen Breite und Höhe einer Box (`max-width`, `max-height`, vgl. [\[CSS 2\]](#), Abschnitte 10.4 und 10.7) DÜRFEN von einer [Bürgerkarten-Umgebung](#) NICHT unterstützt werden, um Überlappungen von Inhalten auszuschließen. Widrigenfalls MUSS das Instanzdokument von der [Bürgerkarten-Umgebung](#) zurückgewiesen werden.

Die Eigenschaften zur Angabe der Zeilenhöhe (`line-height`, `vertical-align`) SOLLTEN von einer [Bürgerkarte](#) unterstützt werden (vgl. [\[CSS 2\]](#), Abschnitt 10.8). Die einzige Ausnahme bildet die Eigenschaft `vertical-align`: Hier MUSS eine [Bürgerkarten-Umgebung](#) jedenfalls die Werte `sub` und `super` interpretieren können.

Sichtbarer Bereich von Boxen

Die Eigenschaft zur Angabe der Sichtbarkeit einer Box (`visibility`) DARF von einer [Bürgerkarten-Umgebung](#) unterstützt werden - vgl. [\[CSS 2\]](#), Abschnitt 11).

Die Eigenschaften zur Steuerung des sichtbaren Bereichs einer Box (`overflow`, `clip`; vgl. [\[CSS 2\]](#), Abschnitt 11) DÜRFEN in einem Instanzdokument NICHT enthalten sein, um versteckte Inhalte auszuschließen. Widrigenfalls MUSS das Instanzdokument von der [Bürgerkarten-Umgebung](#) zurückgewiesen werden.

Generierter Inhalt, Nummerierung, Listen

Die Eigenschaft für die Generierung von Inhalten (`content`) DARF von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [\[CSS 2\]](#), Abschnitte 12.2).

Die Eigenschaft für die Darstellung von Anführungszeichen (`quotes`) DARF von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [\[CSS 2\]](#), Abschnitte 12.3).

Die Eigenschaften für die automatische Nummerierung (`counter-reset`, `counter-increment`) DÜRFEN von einer [Bürgerkarten-Umgebung](#) unterstützt werden - vgl. [CSS 2], Abschnitt 12.5).

Marker-Abstand

Die Eigenschaft zur Bestimmung des Abstandes zwischen einem Marker und der zugehörigen Box (`marker-offset`) DARF von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [CSS 2], Abschnitt 12.6.1).

Listenzeichen

Für die Eigenschaft zur Auswahl des Listenzeichens (`list-style-type`) MUSS eine [Bürgerkarten-Umgebung](#) die Werte `none`, `disc`, `circle`, `square`, `decimal`, `decimal-leading-zero`, `lower-roman`, `upper-roman`, `lower-alpha`, `lower-latin`, `upper-alpha` und `upper-latin` unterstützen. Die übrigen Werte DÜRFEN unterstützt werden (vgl. [CSS 2], Abschnitt 12.6.2).

Position des Listenzeichens

Die Eigenschaft zur Positionierung des Listenzeichens in Bezug auf die zugehörige Box (`list-style-position`) SOLLTE von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [CSS 2], Abschnitt 12.6.2).

Bild als Listenzeichen

Die Eigenschaft zur Auswahl eines Bildes als Listenzeichen (`list-style-image`) DARF von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [CSS 2], Abschnitt 12.6.2). Unterstützt die [Bürgerkarten-Umgebung](#) diese Eigenschaft, so MUSS sie bezüglich der Einbindung des Bildes in die Signatur wie im [Abschnitt 2.1.7](#) beschrieben vorgehen.

Kurzschreibweise

Die Eigenschaft für die Kurzschreibweise der Listeneigenschaften (`list-style`) SOLLTE von einer [Bürgerkarten-Umgebung](#) unterstützt werden. Die EMPFOHLENEN Werte ergeben sich aus den obenstehenden Erläuterungen zu den Eigenschaften `list-style-type`, `list-style-position` und `list-style-image` (vgl. [CSS 2], Abschnitt 12.6.2).

Seitenorientierte Medien

Die Eigenschaften für seitenorientierte Medien DÜRFEN von einer [Bürgerkarten-Umgebung](#) unterstützt werden (`size`, `marks`, `page-break-before`, `page-break-inside`, `page-break-after`, `page`, `orphans`, `widows` - vgl. [CSS 2], Abschnitt 13).

Farben und Hintergrund

Die Eigenschaft zur Bestimmung der Vordergrundfarbe des Inhalts eines Elements (`color`) MUSS von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [CSS 2], Abschnitt 14.1).

Die Eigenschaft zur Bestimmung der Hintergrundfarbe des Inhalts eines Elements (`background-color`; vgl. [CSS 2], Abschnitt 14.2.1) MUSS von einer [Bürgerkarten-Umgebung](#) unterstützt werden.

Die Eigenschaften zur Auswahl und Steuerung eines Bildes als Hintergrund (`background-image`, `background-repeat`, `background-position`, `background-attachment`; vgl. [CSS 2], Abschnitt 14.2.1) DÜRFEN in einem Instanzdokument NICHT enthalten sein, um Überlappungen von Inhalten auszuschließen. Widrigenfalls MUSS das Instanzdokument von der [Bürgerkarten-Umgebung](#) zurückgewiesen werden.

Die Eigenschaft für die Kurzschreibweise der Hintergrundeigenschaften (`background`) SOLLTE von einer [Bürgerkarten-Umgebung](#) unterstützt werden. Die EMPFOHLENEN Werte ergeben sich aus den obenstehenden Erläuterungen zur Eigenschaft `background-color` (vgl. [\[CSS 2\]](#), Abschnitt 14.2.1). Enthält die Eigenschaft Werte zur Auswahl und Steuerung eines Bildes als Hintergrund, MUSS das Instanzdokument von der [Bürgerkarten-Umgebung](#) zurückgewiesen werden.

Schriften

Schriftfamilie

Für die Eigenschaft zur Auswahl der Schriftfamilie (`font-family`) MUSS eine [Bürgerkarten-Umgebung](#) die vordefinierten Werte `serif`, `san-serif` und `monospaced` für die allgemeinen Schriftfamilien unterstützen. Alle übrigen Werte DÜRFEN von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [\[CSS 2\]](#), Abschnitt 15.2.2).

Wird im Instanzdokument eine Schriftfamilie, die durch die [Bürgerkarten-Umgebung](#) nicht dargestellt werden kann, bevorzugt spezifiziert, DARF die [Bürgerkarten-Umgebung](#) das Instanzdokument trotzdem zur Anzeige bringen, wenn eine weitere, darstellbare Schriftfamilie als Alternative spezifiziert wurde. Lautet die Spezifikation im Instanzdokument beispielsweise `font-family: "Times New Roman", serif`, DARF die [Bürgerkarten-Umgebung](#) das Instanzdokument zur Anzeige bringen, auch wenn sie die Schriftfamilie *Times New Roman* nicht kennt (`serif` muss sie ja jedenfalls unterstützen).

Schriftstil

Die Eigenschaften zur Bestimmung des Schriftstils (`font-style`) sowie des Schriftgewichts (`font-weight`) MÜSSEN von einer [Bürgerkarten-Umgebung](#) unterstützt werden. Die Werte `normal` und `italic` MÜSSEN, der Wert `oblique` SOLLTE unterstützt werden.

Die Eigenschaft zur Bestimmung der Schriftvariante (`font-variant`) SOLLTE, jene zur Bestimmung der Schriftstreckung (`font-stretch`) DARF von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [\[CSS 2\]](#), Abschnitt 15.2.3).

Schriftgröße

Die Eigenschaft zur Angabe der Schriftgröße (`font-size`) MUSS von einer [Bürgerkarten-Umgebung](#) unterstützt werden. Die Eigenschaft für die Spezifikation des Streckungsverhältnisses (`font-size-adjust`) DARF von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [\[CSS 2\]](#), Abschnitt 15.2.4).

Kurzschreibweise

Die Eigenschaft für die Kurzschreibweise der Schrifteigenschaften (`font`) SOLLTE von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [\[CSS 2\]](#), Abschnitt 15.2.5). Die EMPFOHLENEN Werte ergeben sich aus den obenstehenden Erläuterungen zu den Eigenschaften `font-style`, `font-variant`, `font-weight`, `font-size` und `font-family`, sowie aus den Erläuterungen zur Eigenschaft `line-height` in [Abschnitt 3.5.4.2](#).

Die zusätzlichen vordefinierten Werte mit Bezug auf die verwendeten Systemschriftarten (`caption`, `icon`, etc.) DÜRFEN in einem Instanzdokument NICHT enthalten sein, um Abhängigkeiten von der Systemumgebung auszuschließen. Widrigenfalls MUSS das Instanzdokument von der [Bürgerkarten-Umgebung](#) zurückgewiesen werden.

Darstellung von Text

Nicht darstellbare Zeichen

Enthält der Text eines Instanzdokuments ein Zeichen, das die [Bürgerkarten-Umgebung](#) nicht anzeigen kann, MUSS das Instanzdokument von der [Bürgerkarten-Umgebung](#) zurückgewiesen werden. Eine alternative Darstellung des Zeichens durch einen Platzhalter DARF NICHT erfolgen.

Einrückung

Die Eigenschaft zur Einrückung der ersten Textzeile eines Blocks (`text-indent`) SOLLTE von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [\[CSS 2\]](#), Abschnitt 16.1).

Ausrichtung

Für die Eigenschaft zur Ausrichtung des Inhalts eines Blocks (`text-align`) MUSS eine [Bürgerkarten-Umgebung](#) die Werte `left`, `right` und `center` unterstützen. Der Wert `justified` SOLLTE, die Angabe eines String-Werts DARF unterstützt werden (vgl. [CSS 2], Abschnitt 16.2).

Verzierung

Für die Eigenschaft zur Verzierung eines Texts (`text-decoration`; vgl. [CSS 2], Abschnitt 16.3.1) MUSS eine [Bürgerkarten-Umgebung](#) die Werte `none`, `underline` und `line-through` unterstützen.

Der Wert `blink` DARF in einem Instanzdokument NICHT enthalten sein. Widrigenfalls MUSS das Instanzdokument von der [Bürgerkarten-Umgebung](#) zurückgewiesen werden.

Die übrigen Werte DÜRFEN von einer [Bürgerkarten-Umgebung](#) unterstützt werden.

Schatten

Die Eigenschaft zur Angabe eines Textschattens (`text-shadow`) DARF von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [CSS 2], Abschnitt 16.3.2).

Wortabstand und Zeichenabstand

Die Eigenschaften für Wortabstand und Zeichenabstand (`word-spacing`, `letter-spacing`) SOLLTEN von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [CSS 2], Abschnitt 16.4). Negative Werte DÜRFEN in einem Instanzdokument NICHT enthalten sein, um Überlappungen von Inhalten auszuschließen. Widrigenfalls MUSS das Instanzdokument von der [Bürgerkarten-Umgebung](#) zurückgewiesen werden.

Kapitalisierung

Die Eigenschaft zur Angabe der Kapitalisierung von Text eines Elements (`text-transform`) DARF von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [CSS 2], Abschnitt 16.5).

Whitespace

Die Eigenschaft zur Behandlung von White Space im Text eines Elements (`white-space`) SOLLTE von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [CSS 2], Abschnitt 16.6).

Tabellen

Position der Beschriftung

Für die Eigenschaft zur Angabe der Position für die Beschriftung einer Tabelle (`caption-side`) SOLLTE eine [Bürgerkarten-Umgebung](#) die Eigenschaften `top` und `bottom` unterstützen; die Eigenschaften `left` und `right` DÜRFEN unterstützt werden (vgl. [CSS 2], Abschnitt 17.4.1).

Layout-Algorithmus

Die Eigenschaft zur Festlegung des Layout-Algorithmus für eine Tabelle (`table-layout`) DARF von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [CSS 2], Abschnitt 17.5.2).

Der Wert `fixed` DARF jedoch NICHT unterstützt werden, da es mit dem damit selektierten Layout-Algorithmus zu Überlappungen von Inhalten kommen kann.

Generell MUSS von der Anzeigekomponente der [Bürgerkarten-Umgebung](#) ein solcher Layout-Algorithmus für eine Tabelle verwendet werden, der keinen *Overflow* erzeugt, mit dem also der Inhalt eines jeden Tabellenelements so gerendert wird, dass er nicht über das Tabellenelement hinausragt. Ein Beispiel für einen solchen Algorithmus findet sich in [CSS 2], Abschnitt 17.5.2, Unterabschnitt *Automatic table layout*.

Ränder

Die Eigenschaften für die Darstellung von Rändern in Tabellen (`border-collapse`, `border-spacing`, `empty-cells`) DÜRFEN von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [CSS 2], Abschnitt 17.6).

Sprachausgabe

Die Eigenschaft zur Steuerung der Sprachausgabe der Spaltenbeschriftung einer Tabelle (`speech-header`) DARF von einer [Bürgerkarten-Umgebung](#) unterstützt werden (vgl. [CSS 2], Abschnitt 17.7.1).

Benutzer-Schnittstelle

Form des Cursors

Die Eigenschaft zur Steuerung der Form des Cursors (`cursor`; vgl. [\[CSS 2\]](#), Abschnitt 18.1) DARF in einem Instanzdokument NICHT enthalten sein. Widrigenfalls MUSS das Instanzdokument von der [Bürgerkarten-Umgebung](#) zurückgewiesen werden.

Anmerkung: Prinzipiell ist es nicht notwendig, diese Eigenschaft zu verbieten, ihre Verwendung macht aber auch keinen Sinn, da sie nicht auf die Zeichnungsebene bezogen ist. Im Sinne einer möglichst schlanken Spezifikation wird die Eigenschaft daher verboten.

Konturen

Die Eigenschaften zur Festlegung von Konturen von Elementen (`outline-width`, `outline-style`, `outline-color` und `outline`; vgl. [\[CSS 2\]](#), Abschnitt 18.4) DÜRFEN in einem Instanzdokument nicht enthalten sein. Widrigenfalls MUSS das Instanzdokument von der [Bürgerkarten-Umgebung](#) zurückgewiesen werden.

Anmerkung: Prinzipiell ist es nicht notwendig, diese Eigenschaften zu verbieten, ihre Verwendung macht aber auch keinen Sinn, da sie auf Elemente bezogen sind, die in einem Instanzdokument nicht enthalten sein dürfen. Im Sinne einer möglichst schlanken Spezifikation wird die Eigenschaft daher verboten.

Sprachausgabe

Die Eigenschaften für die Sprachausgabe eines Dokuments (vgl. [\[CSS 2\]](#), Abschnitt 19) DÜRFEN von einer [Bürgerkarten-Umgebung](#) unterstützt werden.

2.3. Bilder im Standard-Anzeigeformat

Wie aus den [Abschnitten 2](#) und [3](#) hervorgeht, erlaubt das in diesem Dokument spezifizierte Standard-Anzeigeformat auf zweifache Weise die Integration von Bildern:

mittels ``-Tag (vgl. [Abschnitt 2.1.7](#));

mittels CSS-Eigenschaft zur Auswahl eines Bildes als Listenzeichen (vgl. [Abschnitt 3.5.6.4](#)).

Nachdem die somit integrierten Bilder nicht direkt in das Instanzdokument eingebunden, sondern nur mittels URI referenziert werden, müssen die referenzierten Bilddaten als zusätzliche Daten neben dem eigentlichen Instanzdokument in die XML-Signatur aufgenommen werden. Die nachfolgenden Abschnitte spezifizieren diese Integration.

2.3.1. Integration in die XML-Signatur

Für jedes Bild, dass auf eine der zwei oben erwähnten Arten in einem Instanzdokument referenziert wird und mitsigniert werden soll (vergleiche die Alternativen in [Abschnitt 2.1.7](#)), MUSS in die XML-Signatur (neben dem `dsig:Reference` Element für das Instanzdokument) ein weiteres `dsig:Reference` Element aufgenommen werden. Dabei sind folgende Regeln einzuhalten:

- Das Attribut `URI` des `dsig:Reference` Elements MUSS jene URI beinhalten, mit der das entsprechende Bild im Instanzdokument referenziert wird;
- Das Attribut `Type` des `dsig:Reference` Elements ist zu verwenden und MUSS wie folgt aufgebaut sein:
`http://www.buergerkarte.at/specifications/Security-Layer/20031031?Name=SignedImage&InstanceDocRef=x`
Der Platzhalter `x` in der letzten Zeile ist durch den Index jenes `dsig:Reference` Elements zu ersetzen, das zum Integrieren des Instanzdokuments in die Signatur verwendet wird. Die erste `dsig:Reference` der Signatur trägt den Index 0. Wird das selbe Bild in mehrere Instanzdokumente eingebunden, sind die Indizes der Instanzdokument-Referenzen durch Beistriche voneinander zu trennen.

Beispiel

Das folgende Instanzdokument soll von der [Bürgerkarten-Umgebung](#) signiert werden:

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Beispiel</title>
    <style type="text/css">
      ul { list-style-image: url("http://example.com/list-style.gif") }
```

```

</style>
</head>
<body>
  <p>
    Dieses Bild wird mitsigniert:
    
  </p>
  <ul>
    <li>Das ist ein mittels Bild dargestellter Listenpunkt.</li>
  </ul>
</body>
</html>

```

Die dafür notwendige XML-Signatur sieht skizzenhaft wie folgt aus (drei Punkte ... signalisieren Auslassungen aus Gründen der Übersichtlichkeit):

```

<?xml version="1.0" encoding="UTF-8"?>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    ...
    <Reference URI="http://example.com/instanceDocument.xhtml">
      ...
    </Reference>
    <Reference URI=http://example.com/image.gif"
      Type="...ayer/20031031?Name=SignedImage&InstanceDocRef=0">
    </Reference>
    <Reference URI="http://example.com/list-style.gif"
      Type="...ayer/20031031?Name=SignedImage&InstanceDocRef=0">
    </Reference>
  </SignedInfo>
  ...
</Signature>

```

Anmerkung: Der Wert des Attributs URI der ersten Reference wurde willkürlich gewählt. Der tatsächlich zu verwendende Wert hängt vom Signaturerstellungsrequest ab, der an die [Bürgerkarten-Umgebung](#) gesendet wurde.

2.3.2. Prozessmodell für Signaturerstellung und Signaturprüfung

Die beiden nachfolgenden Abschnitte beschreiben das von einer [Bürgerkarten-Umgebung](#) einzuhaltende Prozessmodell bei der Verarbeitung von Dokumenten, die dem in diesem Dokument festgeschriebenen Standard-Anzeigeformat genügen, im Rahmen von Signaturerstellung bzw. Signaturprüfung.

Signaturerstellung

- Für jedes Instanzdokument, das signiert werden soll, führe folgende Schritte durch:
 - Prüfe, ob das Instanzdokument den Regeln dieser Spezifikation entspricht.
 - Füge ein `dsig:Reference` Element für das Instanz-Dokument in die XML-Signatur ein.
 - Prüfe, ob das Instanzdokument referenzierte Bilder enthält. Falls ja, führe für jedes referenzierte Bild, das mitsigniert werden soll, folgende Schritte durch:
 - zusätzlichen `dsig:Reference` in der XML- Prüfe, ob das referenzierte Bild bereits mit einem Signatur berücksichtigt wurde.
 - Falls ja, erweitere das Attribut `Type` dieses `dsig:Reference` Elements um den Index des mit dem Instanzdokument korrespondierenden `dsig:Reference` Elements, wenn der Index nicht schon berücksichtigt wurde.
 - Falls nein, füge ein zusätzliches `dsig:Reference` Element für das Bild in die XML-Signatur ein.
 - Fahre in der Erstellung der XML-Signatur wie für Dokumente anderer Formate fort.

Signaturprüfung

- Prüfe die Gültigkeit der XML-Signatur an Hand des Prozessmodells aus [[XMLDSIG](#)].
- Für jedes durch die XML-Signatur abgedeckte Instanzdokument führe folgende Schritte durch:

- Prüfe, ob das Instanzdokument referenzierte Bilder enthält. Falls ja, führe für jedes referenzierte Bild folgende Schritte durch:
 - Prüfe, ob das referenzierte Bild mit einer entsprechenden zusätzlichen dsig:Reference-Element in der XML-Signatur berücksichtigt ist.
 - Melde all jene referenzierten Bilder als Ergebnis der Signaturprüfung, die mit einer entsprechenden zusätzlichen dsig:Reference-Element in der XML-Signatur berücksichtigt ist.

(Anmerkung: Die [Bürgerkarten-Umgebung](#) kann mit dieser Information dann entsprechend den Vorgaben aus [Abschnitt 2.1.7](#) weiterverfahren).