

## Module für Massenamtssignatur (MASS)

### Konzept und Dokumentation

Graz, am 19. Februar 2010

Dr. Thomas Rössler – [thomas.roessler@egiz.gv.at](mailto:thomas.roessler@egiz.gv.at)

DI Arne Tauber – [arne.tauber@egiz.gv.at](mailto:arne.tauber@egiz.gv.at)

#### Zusammenfassung:

Dieses Dokument legt das grundlegende Konzept eines Moduls für Massenamtssignaturen (MASS) fest, welches auf Grundlage einfacher XML-Signaturen die Erstellung von Amtssignaturen auf Basis von Druckstromrohdaten ermöglicht.

Die so erzeugten XML-Signaturen können als Grundlage für Amtssignaturen dienen; die Einbettung von Signaturblöcken bzw. der für Amtssignaturen erforderlichen Layout-Elemente (z.B. Bildmarke, Hinweis, etc.) sind nicht Teil der Lösung. Amtssignaturen, die auf Basis derart erstellter XML-Signaturen gebildet werden, lassen sich vorwiegend über den Weg der Verifizierung prüfen.

Dieses Konzept ist eine konzeptionelle Unterstützung des konkreten Massenamtssignatur-Projekts der Stadt Wien. Desweiteren wird am Ende des Dokuments eine Beschreibung des Basismoduls für Massenamtssignatur der Stadt Wien geliefert.

## Module für Massenamtssignatur (MASS)

### Inhaltsverzeichnis:

Module für Massenamtssignatur (MASS) .....	1
Konzept und Dokumentation .....	1
Revision History .....	3
1 Einleitung.....	4
2 MASS Modul Konzept.....	4
3 Komponenten .....	6
3.1 Komponenten MASSCore .....	7
3.1.1 Initialisierung.....	7
3.1.2 Methoden .....	8
3.2 Komponenten InputProvider.....	8
3.2.1 Initialisierung.....	8
3.2.2 Methoden .....	9
3.3 Komponente SignatureProvider.....	9
3.3.1 Initialisierung.....	9
3.3.2 Methoden .....	10
3.4 Komponente OutputProvider .....	10
3.4.1 Initialisierung.....	11
3.4.2 Methoden .....	11
3.5 Komponente Configuration.....	12
4 Umsetzung .....	13
5 Dokumentation MassSignature Referenzimplementierung.....	15
Referenzen.....	16

## Revision History

Version	Datum	Autor(en)	
1.0.0	05.10.2009	Thomas Rössler	erstellt.
1.0.1	19.02.2010	Arne Tauber	Anpassung Konzept Dokumentation Signaturtool

## 1 Einleitung

Ziel ist es, ein simples Basis-Modul zu erarbeiten, das zur Erstellung von individuellen Lösungen für die Erzeugung von Amtssignaturen auf Basis von Druckstromdaten herangezogen werden kann (Massensignatur).

Die spezifizierte Lösung soll den Druckstrom pro Dokument vereinzeln (d.h. die Rohdaten pro Ausfertigung entnehmen; je nach Druckstrom bedeutet das Vereinzeln auf Basis von XML-Elementen, Zeilenumbrüchen oder sonstiger Delimiter) und die Druckdaten jedes so vereinzelt Dokuments mit einer einfachen XML-Signatur (detached XML Digital Signature) versehen. Um den Bezug zur Druckvorlage herzustellen (d.h. Vordruck auf Papier, etc.), wird die Druckvorlage via Hash-Wert in die zu signierenden Daten eingebettet (Analogon XML+XSL).

Die so erzeugten Signaturen werden in einer Minimalvariante in einem File gespeichert. Da die Prüfung dieser Amtssignaturen nicht vordergründig via Web-Tool gefordert ist und größtenteils seitens Behörde der Weg der Verifizierung gewählt wird, bleiben die Ablage der Signaturen sowie der exakte Prüfprozess im Einzelfall der Behörde überlassen. Dies auch deswegen, da diese Art der Amtssignatur vordergründig für gedruckte Erledigungen gedacht ist. Jedoch schließt diese Art der Amtssignatur eine auch von BürgerInnen selbst durchführbare Verifikation/Rückführung nicht grundsätzlich aus.

Das in diesem Dokument konzipierte Basismodul ist daher sowohl in Bezug auf die Vereinzelnung und die Anpassung an konkrete Druckstrombedingungen als auch in Bezug auf die Ablage und Verarbeitung der erzeugten Signaturen flexibel und adaptiv gestaltet.

Dieses Dokument skizziert das gewählte Konzept und spezifiziert die notwendigen Hauptelemente des sogenannten MASS-Moduls. Desweiteren beinhaltet dieses Dokument eine Anleitung für das entwickelte Basissignaturmodul.

## 2 MASS Modul Konzept

Das Massenamtsignatur-Modul (MASS-Modul) soll möglichst viel Flexibilität für anwendungsspezifische Erweiterungen bieten. Daher wurde die in Abbildung 1 dargestellte Provider-basierte Komponentenstruktur gewählt.

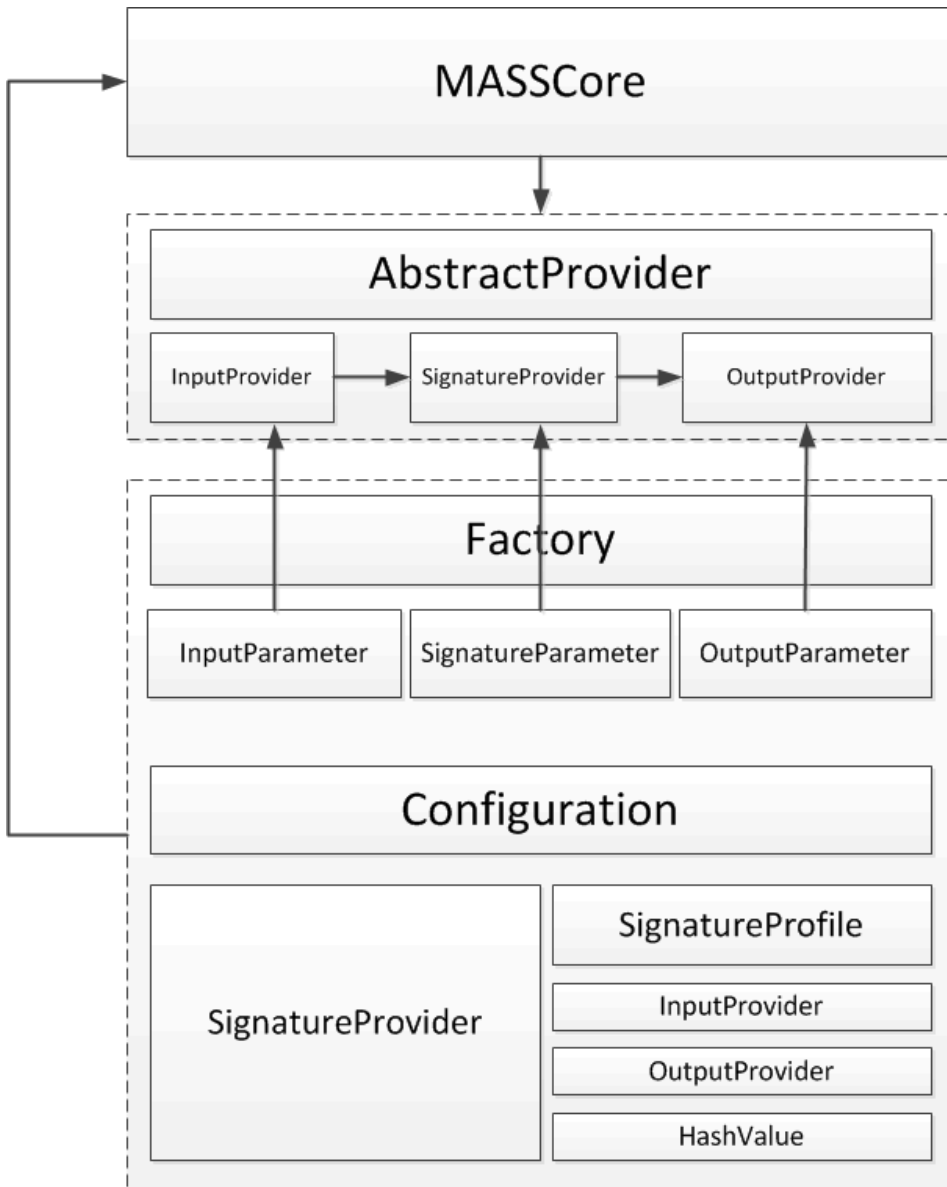


Abbildung 1: MASS Komponenten-Layout

Die wichtigsten Komponenten im Überblick:

- **MASSCore**  
... orchestriert alle Komponenten; ist für die Erstellung der Einzelsignaturen verantwortlich.
- **InputProvider**  
... bereitet den Druckstrom auf; die dabei herausgelösten Einzeldatensätze werden der MASSCore-Komponente zur Signatur bereitgestellt.
- **SignatureProvider**  
... vom InputProvider bereitgestellte Daten werden vom SignatureProvider signiert und anschließend über das MASSCore Modul an den Outputprovider weitergereicht.
- **OutputProvider**  
... legt die erzeugten Einzelsignaturen ab; die genaue Form der Ablage ist anwendungsspezifisch.
- **Factory**

## Module für Massenamtssignatur (MASS)

... ist für die Instanziierung aller Komponenten (außer MASSCore) zuständig. Das MASSCore Modul wird mit der Factory instanziiert.

- **Configuration**

... globale Konfiguration aller Komponenten inklusive Signaturprovider und Signatureprofile mit InputProvider und OutputProvider.

Die nachfolgenden Abschnitte skizzieren die Komponenten im Detail.

Federführendes Element ist die MASSCore-Komponente. Der Gesamtablauf des Massensignaturvorgangs gliedert sich in 3 Phasen:

### 1. Initialisierungsphase

Auf Veranlassung durch die AnwenderIn bzw. des äußeren Anwendungssystems wird die MASSCore-Komponente initialisiert. Im Zuge der Initialisierung wird die Gesamtkonfiguration sowie das konkrete Signaturprofil und die einzelnen Parameter der Provider festgelegt. Die MASSCore-Komponente initialisiert die Instanziierung und Initialisierung der konkreten, anwendungsspezifischen InputProvider, Signaturprovider und OutputProvider.

### 2. Signaturphase

Diese Phase kapselt die iterativen Teile des Massensignaturvorgangs. Die AnwenderIn bzw. das äußere Anwendungssystem startet den Signaturvorgang durch Aufruf der entsprechenden Start-Methode der MASSCore-Komponente. Diese holt daraufhin das nächste zu signierende Datenelement vom zuvor instanziierten InputProvider ab, signiert dieses mit dem Signaturprovider und übergibt die resultierende XML-Signatur dem OutputProvider. Der OutputProvider übernimmt und verarbeitet die erstellte Signatur. Dieser Signaturvorgang wiederholt sich solange, solange der konkrete InputProvider weitere Daten zur Signatur liefern kann. Dies entspricht im wesentlichen einem Iterator-Designpattern; das führende System ist die MASSCore-Komponente. Zu jedem iterativen Schritt stehen dem InputProvider eingangsabhängige InputParameter, dem SignaturProvider InputParameter-abhängige Signaturparameter und dem OutputProvider eingangsabhängige und Signaturparameter-abhängige Outputparameter zur Verfügung. D.h. jedem Parameter stehen die Parameter des vorausgehenden Schritts zur Verfügung.

### 3. Abschlußphase

Nach Erstellung der letzten Signatur, d.h. sobald der InputProvider keine weiteren Signaturdaten liefern kann, veranlasst die MASSCore-Komponente den Abschluss des gesamten Signaturprozesses. Dazu ruft die MASSCore-Komponente entsprechende Finalize-Methoden bei den beteiligten InputProvider-, Signaturprovider- und OutputProvider-Instanzen auf. Im Zuge dieser Finalisierung können bspw. anwendungsspezifische Abschlussschritte, wie etwa das Schließen von Datenbankverbindungen, falls Provider auf diese direkt zugegriffen haben, etc. durchgeführt werden.

## 3 Komponenten

Dieser Abschnitt erläutert die einzelnen Komponenten im Detail.

### **3.1 Komponenten MASSCore**

Diese Komponente stellt den Kern des MASS-Moduls dar. Über entsprechende Methoden dieser Komponente wird der Signaturvorgang initiiert.

Im Signaturprozess selbst ist die MASSCore Komponente für die Erzeugung von Einzelsignaturen verantwortlich. Über den Aufruf einer definierten Start-Methode wird die MASSCore Komponente dazu veranlasst einzelne Datensätze—geliefert von einem konfigurierten InputProvider und deren Ausprägung ist anwendungsspezifisch—mit einer Signatur über den konfigurierten SignatureProvider zu versehen. Die resultierenden Signaturen werden einzeln über das Interface OutputProviderInterface an einen konkreten OutputProvider übergeben, z.B. OutputProviderXML oder OutputProviderText, der die Nachbereitung bzw. das Abspeichern der erzeugten Einzelsignaturen veranlasst.

#### **3.1.1 Initialisierung**

Die MASSCore Methode wird über eine Factory unter Einbeziehung der gemeinsamen Konfiguration (Komponente Configuration) initialisiert. Aus dieser Konfiguration müssen die folgenden, für die MASSCore Komponente relevante Vorgaben erkennbar sein:

##### **a) Signaturprofile (ProfileID)**

Die Konfiguration legt alle relevanten Elemente eines sogenannten Signaturprofiles fest, welche sind wie folgt:

###### **1) InputProvider**

Die konkrete InputProvider Komponente muss anhand der Angaben in der Konfiguration instanzierbar sein. Die MASSCore Komponente instanziiert den konkreten InputProvider, zum Beispiel InputProviderXML, via Reflection-Mechanismen bzw. nach dem Factory Pattern (die konkrete InputProvider-Implementierung muss zur Kompilierzeit nicht feststehen). Der Name der InputProvider-Klasse muss im Konfigurationsprofile angegeben sein oder kann auch allenfalls für eine bestimmte Ausprägung fix vorgegeben sein.

###### **2) OutputProvider**

Die konkrete OutputProvider Komponente muss anhand der Angaben in der Konfiguration instanzierbar sein. Die MASSCore Komponente instanziiert den konkreten OutputProvider, zum Beispiel OutputProviderXML, Reflection-Mechanismen bzw. nach dem Factory Pattern (die konkrete OutputProvider-Implementierung muss zur Kompilierzeit nicht feststehen). Der Name der OutputProvider-Klasse muss im Konfigurationsprofile angegeben sein oder kann auch allenfalls für eine bestimmte Ausprägung fix vorgegeben sein.

###### **3) HashValue**

Dieser Wert legt den fixen Teil des zu signierenden Dokuments fest und wird als fester Bestandteil in die zu signierenden Daten mit aufgenommen.

##### **b) Signatureprovider**

Die konkrete SignatureProvider Komponente muss anhand der Angaben in der Konfiguration instanzierbar sein. Die MASSCore Komponente instanziiert den konkreten SignatureProvider, zum Beispiel XSectProvider, via Reflection-Mechanismen (die konkrete SignatureProvider-Implementierung muss zur Kompilierzeit nicht feststehen). Der Name der SignatureProvider-Klasse muss im Konfigurationsfile angegeben sein oder kann auch allenfalls für eine bestimmte Ausprägung fix vorgegeben sein.

### 3.1.2 Methoden

Im wesentlichen muss die MASSCore-Komponente die folgenden Konstruktor/Methoden zur Verfügung stellen:

a) Konstruktor

Die Initialisierung wird von der AnwenderIn bzw. vom äußeren Anwendungssystem ausgelöst. Der Aufruf des Konstruktors veranlasst die Instanziierung der konkreten Input-, Signatur-/OutputProvider. Als Parameter soll eine Factory angegeben werden, welche für die Instanziierung aller Komponenten sorgt.

b) Initialisierungsmethode `initialize`

Initialisiert sämtliche Provider.

c) Start-Methode `start`

Die Start-Methode wird von der AnwenderIn bzw. vom äußeren Anwendungssystem zum Starten des eigentlichen Signaturprozesses aufgerufen. Daraufhin wird die MASSCore-Komponente alle vom konkreten InputProvider zur Verfügung gestellten Signaturdaten einzeln mittels Einzelsignaturen über den SignatureProvider versehen. Alle Vorgaben für die zu erstellenden XML-Signaturen werden in der Profil-Definition im Rahmen der Initialisierung angeführt. Die resultierenden Signaturen werden anschließend vom OutputProvider verarbeitet.

d) Finalisierungsmethode `finalize`

Finalisiert sämtliche Provider.

## 3.2 Komponenten InputProvider

Die InputProvider Komponente bereitet den zu signierenden Datenstrom auf. Anhand der in der gemeinsamen Konfiguration angegebenen Informationen muss der konkrete InputProvider in der Lage sein, den Massensignaturstrom – gem. den anwendungsspezifischen Vorgaben – in Einzelsignaturanfragen auflösen zu können. Die MASSCore Komponente wird einem Iterator-Pattern folgend iterativ die zu signierenden Daten für jede Signatur einzeln abfragen. Der InputProvider muss diese zur Verfügung stellen.

### 3.2.1 Initialisierung

Zur Konfiguration des InputProviders muss die gemeinsame Konfiguration (Komponente Configuration) herangezogen werden. Die Konfiguration kann dabei providerspezifische Konfigurationselemente beinhalten. InputProvider-spezifische Angaben sind in der Komponente Configuration gekapselt angeführt.

Der konkrete InputProvider wird über Reflection-Mechanismen bzw. über eine Factory von der MASSCore Komponente instanziiert. Im Rahmen einer dezidierten Implementierung kann dieser auch fix vorgegeben werden.

Die Komponente InputProvider ist eine abstrakte Komponente und muss durch konkrete InputProvider-Implementierungen implementiert werden (zum Beispiel InputProviderXML oder InputProviderText). InputProvider, Signaturprovider und OutputProvider haben besonders bezüglich Initialisierung eine Reihe von Gemeinsamkeiten. Beide Komponenten werden von einer abstrakten Provider-Komponente abgeleitet, die diese Gemeinsamkeiten kapselt.

### 3.2.2 Methoden

Die vom InputProvider intern zur Verfügung gestellten Methoden sind providerspezifisch. Es werden hierzu keine weiteren Vorgaben getroffen.

Aus Sicht der MASSCore-Komponente sind alle für das gewählte Iterator-Designpattern notwendigen Methoden zur Verfügung zu stellen:

a) Methode `next`

Methode returniert die nächsten, zu signierenden Daten. Die zu signierenden Daten werden als binäre Daten übergeben. Sollten keine weiteren Daten zur Signatur existieren, so wird die Signaturerzeugung von der MASSCore Komponente beendet und die Finalisierung der einzelnen Komponenten wird gestartet. Die MASSCore Komponente ruft im Zuge des Iterationsschrittes diese Methode auf, um die zu signierenden Daten für die nächste Einzelsignatur vom InputProvider abzurufen.

b) Methode `hasNext`

Diese Methode gibt nur boolesche Werte zurück. Sie liefert `true`, solange weitere Einzeldatensätze zur Signatur vorliegen. Andernfalls lautet der Rückgabewert `false`.

Zusätzliche Methoden (vererbt von AbstractProvider):

c) Methode `initialize`

Die Initialisierung wird von der MASSCore Komponente ausgelöst. Der Aufruf der `init`-Methode veranlasst die Instanzierung des konkreten Providers. Die Implementierung der `initialize`-Methode ist überwiegend providerspezifisch.

d) Methode `configure`

Die Konfiguration wird von der MASSCore Komponente ausgelöst. Der Aufruf der `configure`-Methode veranlasst die Konfiguration des konkreten Providers. Die Implementierung der `configure`-Methode ist überwiegend providerspezifisch. Als Parameter können eine Reihe von Konfigurationsparametern in Form von Properties übergeben werden.

e) Methode `doFinal`

Diese Methode wird von der MASSCore-Komponente zum Abschluss des Gesamtprozesses aufgerufen. Im Zuge der `finalize`-Methode kann der Provider alle abschließenden Tätigkeiten vornehmen, wie das Schließen offener Verbindungen, Dateien, etc. Die Implementierung der `finalize`-Methode ist überwiegend providerspezifisch.

## 3.3 Komponente *SignatureProvider*

Die *SignatureProvider* Komponente ist für die eigentliche Erzeugung der Signatur zuständig. Von der MASSCore Komponente werden die einzelnen zu signierenden Datensatz in Form von Byte-Arrays an die *SignatureProvider* Komponente übergeben, welche für die Signaturerzeugung zuständig ist und die Signatur in Form eines Byte-Arrays an den OutputProvider übergeben muss.

### 3.3.1 Initialisierung

Zur Konfiguration des *SignatureProvider* muss die gemeinsame Konfiguration (Komponente Configuration) herangezogen werden. Die Konfiguration kann dabei providerspezifische Konfigurationselemente beinhalten. *SignatureProvider*-spezifische Angaben sind in der Komponente Configuration gekapselt angeführt.

## Module für Massenamtsignatur (MASS)

Der konkrete SignatureProvider wird über Reflection-Mechanismen bzw. über eine Factory von der MASSCore Komponente instanziiert. Im Falle einer dezidierten Implementierung kann diese auch fix vorgegeben sein.

Die Komponente SignatureProvider ist eine abstrakte Komponente und muss durch konkrete SignatureProvider-Implementierungen umgesetzt werden (zum Beispiel XsectSignatureProvider).

InputProvider, SignatureProvider und OutputProvider haben besonders bezüglich Initialisierung eine Reihe von Gemeinsamkeiten. Beide Komponenten werden von einer abstrakten Provider-Komponente abgeleitet, die diese Gemeinsamkeiten kapselt.

### 3.3.2 Methoden

Die vom SignatureProvider intern zur Verfügung gestellten Methoden sind providerspezifisch. Es werden hierzu keine weiteren Vorgaben getroffen.

Aus Sicht der MASSCore-Komponente sind für die gewählte Architektur folgende Methoden erforderlich:

a) Methode `sign`

Diese Methode wird von der MASSCore Komponente aufgerufen, um eine Einzelsignatur zu erstellen. Der SignatureProvider übernimmt dabei das zu signierende Byte-Array und die Signaturparameter um die Signatur zu erzeugen. Die Methode liefert als Result ebenso ein Byte-Array zurück.

Zusätzliche Methoden (teilweise vererbt von AbstractProvider):

b) Methode `setHashValue`

Diese Methode dient dazu, um den Hashwert für den fixen Teil des zu signierenden Dokuments festzulegen.

c) Methode `initialize`

Die Initialisierung wird von der MASSCore Komponente ausgelöst. Der Aufruf der `init`-Methode veranlasst die Instanzierung des konkreten Providers. Die Implementierung der `initialize`-Methode ist überwiegend providerspezifisch.

d) Methode `configure`

Die Konfiguration wird von der MASSCore Komponente ausgelöst. Der Aufruf der `configure`-Methode veranlasst die Konfiguration des konkreten Providers. Die Implementierung der `configure`-Methode ist überwiegend providerspezifisch. Als Parameter können eine Reihe von Konfigurationsparametern in Form von Properties übergeben werden.

e) Methode `doFinal`

Diese Methode wird von der MASSCore-Komponente zum Abschluss des Gesamtprozesses aufgerufen. Im Zuge der `finalize`-Methode kann der Provider alle abschließenden Tätigkeiten vornehmen, wie das Schließen offener Verbindungen, Dateien, etc. Die Implementierung der `finalize`-Methode ist überwiegend providerspezifisch.

### 3.4 Komponente OutputProvider

Die OutputProvider Komponente bereitet das Ergebnis des Signaturvorgangs auf. Anhand der in der gemeinsamen Konfiguration angegebenen Informationen muss der konkrete OutputProvider in der Lage sein, die einzelnen, von der MASSCore-Komponente erzeugten Signaturen in ein

anwendungsspezifisches Gesamtergebnis überzuführen. Die Einzelsignaturen werden von der MASSCore-Komponente erzeugt und an den OutputProvider übergeben.

### 3.4.1 Initialisierung

Zur Konfiguration des OutputProvider muss die gemeinsame Konfiguration (Komponente Configuration) herangezogen werden. Die Konfiguration kann dabei providerspezifische Konfigurationselemente beinhalten. OutputProvider-spezifische Angaben sind in der Komponente Configuration gekapselt angeführt.

Der konkrete OutputProvider wird über Reflection-Mechanismen bzw. über eine Factory von der MASSCore Komponente instanziiert. Im Falle einer dezidierten Implementierung kann dieser auch fix vorgegeben werden.

Die Komponente OutputProvider ist eine abstrakte Komponente und muss durch konkrete OutputProvider-Implementierungen implementiert werden (zum Beispiel OutputProviderXML oder OutputProviderText).

InputProvider, SignatureProvider und OutputProvider haben besonders bezüglich Initialisierung eine Reihe von Gemeinsamkeiten. Beide Komponenten werden von einer abstrakten Provider-Komponente abgeleitet, die diese Gemeinsamkeiten kapselt.

### 3.4.2 Methoden

Die vom OutputProvider intern zur Verfügung gestellten Methoden sind providerspezifisch. Es werden hierzu keine weiteren Vorgaben getroffen.

Aus Sicht der MASSCore-Komponente sind für die gewählte Architektur folgende Methoden erforderlich:

a) Methode `append`

Diese Methode wird von der MASSCore Komponente aufgerufen, um eine erstellte Einzelsignatur dem konkreten OutputProvider zu übergeben. Der OutputProvider verarbeitet die übergebene Einzelsignatur applikations- bzw. providerspezifische (z.B. Ablage im File-Systeme, Datenbank, etc.) weiter. Als Übergabeparameter wird die erzeugte Einzelsignatur als Byte-Array sowie adaptierbare OutputParameter übergeben.

Zusätzliche Methoden:

b) Methode `initialize`

Die Initialisierung wird von der MASSCore Komponente ausgelöst. Der Aufruf der `init`-Methode veranlasst die Instanzierung des konkreten Providers. Die Implementierung der `initialize`-Methode ist überwiegend providerspezifisch.

c) Methode `configure`

Die Konfiguration wird von der MASSCore Komponente ausgelöst. Der Aufruf der `configure`-Methode veranlasst die Konfiguration des konkreten Providers. Die Implementierung der `configure`-Methode ist überwiegend providerspezifisch. Als Parameter können eine Reihe von Konfigurationsparametern in Form von Properties übergeben werden.

d) Methode `doFinal`

Diese Methode wird von der MASSCore-Komponente zum Abschluss des Gesamtprozesses aufgerufen. Im Zuge der `finalize`-Methode kann der Provider alle abschließenden Tätigkeiten

vornehmen, wie das Schließen offener Verbindungen, Dateien, etc. Die Implementierung der finalize-Methode ist überwiegend providerspezifisch.

### 3.5 Komponente Configuration

Die Komponente Configuration ist für die Gesamtkonfiguration des MASS-Systems verantwortlich. Die Configuration-Komponente steht somit allen beteiligten Komponenten zur Verfügung.

Als Basis für die Konfiguration wird ein XML- oder Properties-File vorgeschlagen. Dieses File enthält sowohl globale Vorgaben als auch applikationsspezifische Vorgaben. Als technische Basis soll das Apache Commons Configuration Framework oder das Java Properties Package dienen.

Die Konfiguration soll hierarchisch aufgebaut werden. Das heißt jedes Konfigurationselement selbst kann weitere, detailliertere Konfigurationsangaben bedingen, die entsprechend gekapselt mitgegeben werden (z.B. konkrete Provider-Konfiguration etc.). Für einfache Basisimplementierungen kann auch eine flache Konfiguration verwendet werden.

Die vorgeschlagene Konfiguration selbst soll aus zwei Hauptblöcken bestehen:

1. Globale Einstellungen

Hierin sind globale Vorgaben möglich, die alle beteiligten Komponenten betreffen.

Im Rahmen des globalen Konfigurationsblocks können auch globale default-Vorgaben festgelegt werden, die für alle nachfolgend definierten Signaturprofile gelten, sofern das konkrete Signaturprofil die default-Werte nicht explizit überschreibt.

2. Signaturprofile

Das Konfigurationsfile definiert eine beliebige Anzahl von Signaturprofilen. Jedes Profil ist in einem Block gekapselt und wird mit einem Konfigurationsfile-weitem, eindeutigen Profile-Identifizier (profileID) gekennzeichnet. Jedes Profil muss zumindest festlegen:

- a. InputProvider-Implementierung

Für jedes Signaturprofil muss eine konkrete InputProvider-Implementierung festgelegt werden. Die konkrete InputProvider-Implementierung wird mittels Klassen-Namen angegeben werden, sodass diese im Zuge der Initialisierungsphase instanziiert werden kann.

- b. InputProvider-spezifische Konfigurationen

Bei jedem Signaturprofil können für die konkrete InputProvider-Instanz beliebige, provider-spezifische Konfigurationen mitgegeben werden, z.B. Zugangsdaten für Datenbankzugriffe, File-Bezeichnungen, Delimiter von Einzeldatensätze, etc.. Diese Konfigurationselemente müssen nur dem jeweiligen InputProvider bekannt sein. Von allen anderen Komponenten wird der provider-spezifische Konfigurationsblock ignoriert.

- c. OutputProvider-Implementierung

Für jedes Signaturprofil muss eine konkrete OutputProvider-Implementierung festgelegt werden. Die konkrete OutputProvider-Implementierung kann mittels Klassen-Namen angegeben werden, sodass diese im Zuge der Initialisierungsphase instanziiert werden kann.

- d. OutputProvider-spezifische Konfigurationen

Bei jedem Signaturprofil können für die konkrete OutputProvider -Instanz beliebige, provider-spezifische Konfigurationen mitgegeben werden, z.B. Zugangsdaten für Datenbankzugriffe, File-Bezeichnungen, Delimiter von Einzeldatensätze, etc.. Diese Konfigurationselemente müssen nur dem jeweiligen OutputProvider bekannt sein.

Von allen anderen Komponenten wird der provider-spezifische Konfigurationsblock ignoriert.

### e. SignatureProvider-Implementierung

Für jedes Signaturprofil muss eine konkrete SignatureProvider-Implementierung festgelegt werden. Die konkrete SignatureProvider-Implementierung wird mittels Klassen-Namen angegeben werden, sodass diese im Zuge der Initialisierungsphase instanziiert kann. Allenfalls können auch globale SignatureProvider in der Konfiguration definiert werden, die dann im Signaturprofil nur mehr mittels ID referenziert werden.

### f. SignatureProvider-spezifische Konfigurationen

Bei jedem Signaturprofil können für die konkrete SignatureProvider -Instanz beliebige, provider-spezifische Konfigurationen mitgegeben werden, z.B. Zugangsdaten für Datenbankzugriffe, File-Bezeichnungen, Delimiter von Einzeldatensätze, etc.. Diese Konfigurationselemente müssen nur dem jeweiligen SignatureProvider bekannt sein. Von allen anderen Komponenten wird der provider-spezifische Konfigurationsblock ignoriert. Allenfalls können im Zuge einer globalen SignatureProvider Definition diese Angaben in den entsprechendne globalen SignatureProvider-Definitionen definiert werden.

### g. Hashwert auf Vordruck/Layoutmaske

Um den Bezug zur Druckvorlage herzustellen, die ebenfalls für die Amtssignatur relevante Angaben enthält, wird ein Hash-Wert der Druckvorlage bzw. einer Layoutmaske dem Profil als Konfigurationsparameter hinterlegt. Dieser Hash-Wert muss jeder Signatur, entweder als zusätzliche Referenz oder im Rahmen der zu signierenden Daten, beigebracht werden. Nur so ist eine Verkettung der Vorlage/Layoutmaske mit den Druckrohdaten gewährleistet. Die Erstellung des Hashwertes kann bspw. durch Hashen eines qualitativ hochwertigen Scans der Vorlage/Layoutmaske erfolgen. Der Scan sowie der Hash-Vorgang müssen nachvollziehbar bzw. mit den Signaturdaten gemeinsam archiviert werden.

Jedes Signaturprofil ist somit mit einem Vordruck bzw. mit einer Layoutmaske verknüpft.

Die konkrete Festlegung der Konfigurationsstruktur erfolgt im Zuge der Implementierung.

## 4 Umsetzung

Im Zuge einer prototypischen Umsetzung werden folgende Komponenten implementiert:

### 1. Konfiguration

auf Basis eines textbasierten Konfigurationsfiles

### 2. MASSCore Komponente

Basiskernkomponente.

### 3. XML- und file-basierter InputProvider *InputProviderXML*

Provider öffnet ein per Konfiguration angegebenes File, entnimmt einzelne Datensätze, die aufgrund eines vorkonfigurierten Delimiters (via Konfigurationsfile) separiert werden, und stellt diese der MASSCore-Komponente zur Verfügung.

### 4. XML-basierter *XSectSignaturetProvider*

## Module für Massenamtssignatur (MASS)

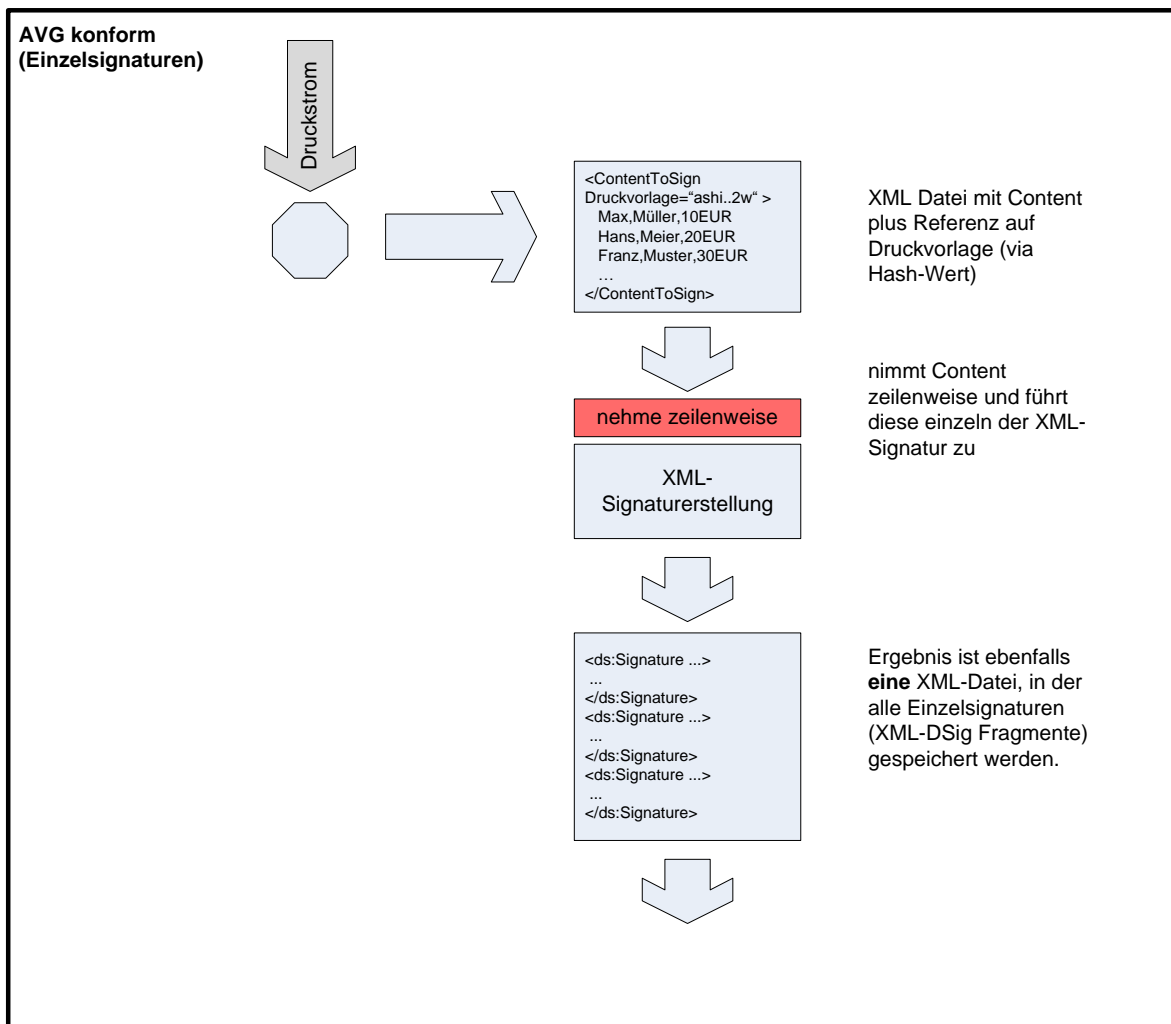
Erzeugt mittels dem XSECT SignatureProvider enveloping XML Signaturen, welche den Hashwert als XML Attribut und die signierten Daten Base64 kodiert in die resultierende Signatur einbettet.

### 5. XML- und file-basierter OutputProvider *OutputProviderXML*

Provider öffnet ein File, dessen Bezeichnung als providerspezifischer Parameter aus der Konfiguration entnommen wird, und schreibt die erstellten XML-Signaturen gesammelt in dieses File.

Die Umsetzung weiterer Input-/OutputProvider ist nicht geplant. Da das Projekt als OpenSource-Projekt im Rahmen der OpenSource Plattform des Digitalen:Österreich geführt wird, sind beliebige Erweiterungen durch Dritte einfach möglich.

Die prototypische Umsetzung soll im wesentlichen den in Abbildung 2 gezeigten Prozess realisieren.



**Abbildung 2: Musterprozess für prototypische Umsetzung**

Weitere technische Eckdaten der Umsetzung:

- OpenSource Projekt gemäß EUPL (berücksichtigt kostenfreie Nutzung der referenzierten Bibliotheken)
- Basis: Java 1.5

c) OpenSource Projekt im Rahmen egovlabs.gv.at: vorgeschlagener Projektitel MASS

## 5 Dokumentation MassSignature Referenzimplementierung

Dieser Abschnitt dokumentiert die Referenzimplementierung des Massensignaturmoduls für die Stadt Wien.

Aufgerufen wird die Referenzimplementierung über die dem Projekt beiliegende sign.bat mit zusätzlichem Parameter des Konfigurationsfiles oder über die at.gv.egiz.wien.MassSignature Klasse mit den Argumenten „-c CONFIG\_FILE“

Variante 1: sign.bat conf/sample-conf.txt

Variante 2: java -c ...classpath... at.gv.egiz.wien.MassSignature -c conf/sample-conf.txt

Die Konfigurationsdatei enthält folgende Parameter:

<i>Parameter</i>	<i>Bechreibung</i>	<i>Optional</i>
<b>InputFile</b>	Absolute oder relative Angabe der InputDatei für die Signaturdaten. Jeweils eine Zeile des InputFiles wird als Signaturdaten herangezogen.	Nein
<b>OutputDirectory</b>	Verzeichnis, wo die erzeugten Signaturen geschrieben werden.	Nein
<b>PKCS12File</b>	PKCS#12 Datei des Software-Signaturzertifikats.	Nein
<b>PKCS12Password</b>	Passwort für die Zertifikatsdatei.	Nein
<b>HashValue</b>	Hashwert des fixen Teils des Dokuments.	Nein
<b>ReferenceNumberLocation</b>	Gibt den Ort der Geschäftszahl innerhalb der Zeile (der im InputFile gelesenen Zeile) an. Wert muss die Form „x,y“ haben. X gibt das x-te Zeichen innerhalb der Zeile und y die Länge der Geschäftszahl an. Bsp. 205,12	Ja
<b>NamePattern</b>	Gibt ein konfigurierbares Pattern für den Dateinamen an, der in das OutputDirectory für die jeweilige Signatur geschrieben wird. Folgende Platzhalter sind definiert:  GZ = Geschäftszahl  yyyy = Jahr  MM = Monat	Ja

## Module für Massenamtsignatur (MASS)

	dd = Tag
	HH = Stunde
	mm = Minute
	ss = Sekunde
	Alle Nicht-Datumswerte müssen in Hochkomma „“ gesetzt werden. Bspw.
	'TEST-GZ-'yyyyMMdd'-HHmmss'.xml'
<b>IncludeSignatureCertificate</b>	Gibt an, ob das Signaturzertifikat in die resultierende Signatur mit aufgenommen werden soll. Ja

### Beispiel:

```
# Sample text-based configuration file
InputFile=test/input/RVCST_Rohdaten.txt
OutputDirectory=test/output
PKCS12File=conf/signaturekeys.p12
PKCS12Password=test
HashValue=A4F3CD
ReferenceNumberLocation=205,12
NamePattern='TEST-GZ-'yyyyMMdd'-HHmmss'.xml'
IncludeSignatureCertificate=true
```

## Referenzen

- [1] JSR 105: XML Digital Signature APIs, Abgerufen aus dem World Wide Web am 01. 10. 2009 unter <http://www.icp.org/en/jsr/detail?id=105>.
- [2] Eastlake, Donald, Reagle, Joseph und Solo, David: XML-Signature Syntax and Processing. W3C Recommendation, Februar 2002. Abgerufen aus dem World Wide Web am 01. 10. 2009 unter <http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/>.