



MABxml-1

Ein XML-Schema für das MAB2-Format

Dokumentation

Version 1.2

Vorwort

XML ist eine Technologie, die sich bereits auf breiter Basis (insbesondere als Syntax für Austauschformate) durchgesetzt hat. Viele Software-Anwendungen wurden für XML und die damit verwandten Technologien wie XML-Schema und XSLT konzipiert. So erlaubt beispielsweise das viel beachtete OAI-Protokoll [9] ausschließlich den Austausch von Daten mit XML-konformer Syntax.

Das Maschinelle Austauschformat für Bibliotheken (MAB2) ist noch immer das wichtigste Austauschformat für bibliografische Daten im deutschsprachigen Raum. Es lag also nahe, beide Standards – MAB2 und XML – zusammenzubringen und damit eine Möglichkeit zu schaffen, auch MAB-Daten in eine XML-Struktur zu transportieren.

Mit MARCXML [3] – einem XML-Schema für die Eins-zu-Eins-Übertragung von MARC21-Datensätzen in eine XML-Struktur – hat die Library of Congress diesen Schritt für MARC21 bereits durchgeführt. MABxml soll eine vergleichbare Funktion für MAB2 erfüllen.

Ich möchte allen herzlich danken, die mir bei der Erstellung des Formats geholfen haben und auch jenen, die ihr Interesse an dem Thema gezeigt haben. Besonderen Dank schulde ich meinem Kollegen Reinhold Heuvelmann, der mit viel Geduld alle meine Fragen beantwortete, sowie Herrn Thomas Berger, dessen kritische Beiträge, Korrekturen und Vorschläge mehr als hilfreich waren.

Jürgen Kett

Frankfurt am Main, Dezember 2003

Änderung gegenüber Version 1.1

Aus der Tabelle im Abschnitt 3.3.5 wurde aus den Erläuterungen zum Attribut *id* der folgende Absatz gestrichen:

"Für Datensatz-Elemente gilt hierbei eine besondere Regelung: Die id des Datensatz-Elementes muss der Identifikationsnummer des jeweiligen Datensatzes entsprechen (MAB2-Feld 001)."

Diese Einschränkung ist nicht immer umsetzbar, weil der Typ *xsd:ID* spezielle Anforderungen an die Syntax stellt, die von der Identifikationsnummer des Datensatzes unter Umständen nicht erfüllt wird (Zum Beispiel muss das erste Zeichen ein Buchstabe sein, was wohl bei den wenigsten der zurzeit kursierenden Identifikationsnummern der Fall sein dürfte).

Inhalt:

Vorwort	2
1 Einleitung	4
1.1 Zweck und Anwendungsbereich	4
1.2 Von MAB nach MABxml und umgekehrt	4
1.3 Aufbau des Dokumentes.....	4
2 XML und XML-Schema	5
2.1 Informationsquellen.....	5
2.2 Historie	5
2.3 XML kurz gefasst	6
2.4 Definition von XML-Dokumentstrukturen	7
2.5 Namespaces	8
3 MABxml-1	9
3.1 Zu Beginn ein Beispiel	9
3.2 Die Komponenten einer MABxml-Spezifikation.....	11
3.3 Das MABxml-1-Schema.....	11
3.3.1 Namespace	11
3.3.2 Struktur eines Datensatzes.....	11
3.3.3 Obligatorische Attribute	13
3.3.4 Sammlungen von Datensätzen	13
3.3.5 Optionale Attribute.....	14
3.4 Übertragungsregeln.....	15
3.4.1 Allgemein.....	15
3.4.2 Satzkennung	16
3.4.3 Unterfelder	16
3.4.4 Verbleibende Steuerzeichen.....	16
3.4.5 Lesbarkeit (Einrückungen und Zeilenwechsel).....	17
3.5 Zeichenvorrat und Zeichenkodierung	17
4 FAQ rund um MAB + XML	19
4.1 Darf „normales“ MAB2 über OAI transportiert werden?	19
4.2 Darf der Zeichensatz von MAB oder MAB-Diskette für OAI verwendet werden?	19
4.3 Ist MAB2 XML-konform?	19
4.4 Warum hat MABxml kein Satzkennungselement?	19
Referenzen	21

Abbildungen:

Abb. 1: Struktur eines MABxml-Datensatzes	11
Abb. 2: Die Basiselemente von MABxml	12
Abb. 3: Das Element „datei“	14

1 Einleitung

1.1 Zweck und Anwendungsbereich

MABxml erlaubt den Transport von MAB2-Datensätzen in einer XML-Struktur. Es schließt daher die Lücke zwischen den immer zahlreicher werdenden XML-basierten Technologien und dem in Deutschland wichtigsten Austauschformat für bibliografische Daten.

Mit MABxml kann man MAB-Daten beispielsweise über das Harvesting-Protokoll OAI-PMH [9] austauschen oder mit dem Z39.50-Nachfolger SRW/U [11] suchen und zurückliefern, XSLT-Dokumente für die Beschreibung und Durchführung von Konversionen von MABxml nach verschiedenen anderen Formaten (z.B. nach Dublin Core, MODS oder MARCXML) nutzen und XML-Parser für die Weiterverarbeitung und Validation von MAB-Datensätzen einsetzen.

1.2 Von MAB nach MABxml und umgekehrt

Die Grundvoraussetzung für die Brückenfunktion zwischen MAB2 und XML wird durch das Format erfüllt: Die Konversion zwischen MAB2 und MABxml ist intuitiv und leicht zu implementieren. Entsprechende Konversionsprogramme werden auf der MABxml-Informationseite [20] veröffentlicht.

1.3 Aufbau des Dokumentes

Im nächsten Abschnitt wird zunächst ein kurzer Überblick über XML gegeben. Diesen Abschnitt können XML-Erfahrene überspringen.

Im Anschluss wird die aktuelle und vorerst einzige Variante von MABxml, das Format „MABxml-1“, beschrieben und erklärt, wie man MAB/MABxml-Konversionen durchführt. Zum Abschluss gibt es noch einen Abschnitt mit Fragen rund um die Thematik „MAB und XML“.

2 XML und XML-Schema

2.1 Informationsquellen

Am Ende des Dokumentes finden sich einige Referenzen zum Thema XML und XML-Schema. Für eine genaue Spezifikation von XML und XML-Dokumenten verweise ich auf [1]. Ich möchte in diesem Kapitel lediglich einen sehr kurzen Überblick über die Sprache geben. Einfache Tutorials zu XML und XML-Schema gibt es unter [18].

2.2 Historie

XML entstand aus den Erfahrungen mit der Structured Generalized Markup Language (SGML) und der Hyper Text Markup Language (HTML). Beide Sprachen hatten ihre Schwächen: SGML erwies sich für viele Anwendungen als zu komplex, HTML (konzipiert für die Darstellung von Dokumenten im Internet) vermischt Dokumentinhalt/Datenstruktur und deren Darstellung. Als Folge gründete sich 1996 eine Arbeitsgruppe unter der Schirmherrschaft des World-Wide-Web-Konsortiums (W3C) und es entstand die Extensible Markup Language (XML).

Die Extensible Markup Language ist eine Sprache, mit der eine Klasse von Datenobjekten, genannt XML-Dokumente, beschrieben werden können. XML stellt im Hinblick auf die Anwendbarkeit eine bedeutsame Vereinfachung gegenüber SGML dar und bringt gegenüber der Strukturierungsmöglichkeit von Dokumenten eine erhebliche Verbesserung gegenüber HTML.

XML-Dokumente enthalten beides: die Datenstruktur und Dateninhalte. Sie bilden eine Baumstruktur, sind also hierarchisch strukturiert.

Diese Struktur wird durch Markierungen, ein sogenanntes *Markup*, definiert (daher auch „Markup Language“). Diese sind, kurz gesagt, einfache Zeichenfolgen, die in spitze Klammern eingeschlossen werden.

Beispiel 1

```
<autor>Ernst, Klaus</autor>
```

Der Vorteil dieses Systems ist, dass sich die Möglichkeit bietet, die Datenstruktur deskriptiv und für Menschen lesbar zu gestalten. D.h. XML ist kein rechnerinternes Datenmodell, weshalb es sich insbesondere als Austauschformat für lose gekoppelte Systeme (wie das Internet) eignet.

2.3 XML kurz gefasst

Die hierarchisch oberste Struktureinheit in XML ist das Dokument (*document*). Dieses besteht aus sogenannten Elementen (*elements*). Ein Dokument besteht aus mindestens einem Element, dem sog. Wurzelement. Ein Element kann wiederum Unterelemente enthalten oder Inhaltsdaten oder beides. Jedes Element beginnt und endet mit *Markup*, auch genannt *start-tag* bzw. *end-tag*, die die Form „<Elementbezeichnung>“ bzw. „</Elementbezeichnung>“ haben. Darin eingeschlossen sind die Unterelemente, sowie die Inhaltsdaten. Das folgende Beispiel zeigt den möglichen Aufbau eines fiktiven Papieres als XML-Dokument.

Beispiel 2

```
<papier>
  <titel>MABxml v1.0</titel>
  <kapitel>
    <ueberschrift>Einführung</ueberschrift>
    <paragraph>XML wurde durch eine Arbeitsgruppe...</paragraph>
    ...
  </kapitel>
  ...
</papier>
```

Neben der Elementbezeichnung können einem Element noch *Attribute* hinzugefügt und diese mit Werten belegt werden. Die Attribute werden als Liste durch Leerzeichen getrennt an die Elementbezeichnung angeschlossen. Attribute werden in der Regel dazu verwendet, ein Element näher zu spezifizieren. Im folgenden Beispiel werden dem Element *papier* die Attribute *id* zur Identifikation und *status* zur Kennzeichnung des Dokumentstatus beigefügt.

Beispiel 3

```
<papier id = "12345x" status = "Entwurf">Einführung in XML</papier>
```

Es ist auch möglich, zu einem XML-Dokument durch Verwendung der Tags „<!--“ und „-->“ ergänzende Kommentare hinzuzufügen. Diese sind nicht Teil der Datenstruktur, sondern dienen lediglich zur Auskommentierung.

Beispiel 4

```
<!-- dies ist ein kommentar -->
```

2.4 Definition von XML-Dokumentstrukturen

Für den Datenaustausch ist es wichtig, dass die miteinander kommunizierenden Seiten sich auf ein spezielles Austauschformat einigen (insbesondere um Maschinenlesbarkeit zu gewährleisten). Wie oben beschrieben, ist XML lediglich ein syntaktisches Rahmenwerk für die Erstellung von Datenobjekten. Eine Aussage wie „wir nutzen XML als Austauschformat“ sagt daher noch nichts über die eigentliche Datenstruktur aus, sondern nur über gewisse Regeln, denen die Datenstruktur unterworfen ist.

Um eindeutige Datenstrukturen für XML-Dokumente zu definieren, wurden zwei Beschreibungssprachen entworfen:

- 1.) **Document Type Definition (DTD)** [1]: Diese ermöglicht die Definition der in einem XML-Dokument zu verwendenden Elemente und Attribute, sowie der Hierarchie, in der diese Elemente auftreten dürfen. Außerdem können Einschränkungen für die Inhalte und Verwendung von Elementen und Attribute definiert werden.
- 2.) **XML Schema** [2]: Nachdem DTDs sich für viele Anwendungen als nicht mächtig genug herausstellten, entwickelte man mit *XML Schema* eine wesentlich komplexere Beschreibungssprache, in der die Funktionalität der DTD vollständig enthalten ist. Diese erlaubt u.a. durch Typdefinitionen die präzise Festlegung von Inhalten. So können beispielsweise Muster (*patterns*) dazu verwendet werden, für ein Element nur Einträge zu erlauben, die dem Muster entsprechen (z.B. nutzbar für Datumsfelder).

Diese Strukturbeschreibungen können dazu verwendet werden, einerseits das Austauschformat eindeutig zu spezifizieren und andererseits beim Austausch die Gültigkeit einer Anfrage bzw. Antwort zu überprüfen. Hierzu muss das zu überprüfende Dokument auf das jeweilige Schema bzw. die jeweilige DTD verweisen.

Das folgende Beispiel (Quelle [18]) zeigt wie eine Referenz auf ein XML-Schema – in diesem Falle das Schema „note.xsd“ – aussieht.

Beispiel 5 Referenz eines Instanzdokumentes auf das zugehörige Schema

```
<?xml version = "1.0"? >
<note xmlns = "http://www.w3schools.com"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation = "http://www.w3schools.com note.xsd" >

<to> Tove </to>
<from> Jani </from>
<heading> Reminder </heading>
<body> Don't forget me this weekend! </body>
</note>
```

2.5 Namespaces

XML-Namespaces wurden für den Fall definiert, in dem in einem XML-Dokument Teile von einem anderen XML-Dokument aufgenommen werden sollen. Das Problem ist, dass so Elementnamen und Attribute doppelt auftauchen können und dabei doppeldeutig für den Leser oder das Verarbeitungsprogramm werden.

Nehmen wir an, wir hätten ein XML-Format zur Speicherung von MARC21-Datensätzen und eines zur Speicherung von MAB2-Datensätzen definiert und beide Formatdefinitionen enthielten das Element „record“. Wollten wir nun ein XML-Dokument erstellen, das beide Formate parallel nutzt, kämen wir in einen Namenskonflikt: Bedeutet die Zeichenkette „<record>“, dass ein MARC21-Datensatz folgt oder, dass nun ein MAB2-Datensatz folgt. Um solche Namenskonflikte zu lösen, wurden Namensräume (sog. Namespaces) definiert [17]. Für jede Formatbeschreibung (also eine DTD oder ein XML-Schema), die in einem XML-Dokument verwendet wird, kann ein Präfix definiert werden. Mit diesem Präfix kann der Autor für jedes Element kennzeichnen, auf welche Formatbeschreibung es sich bezieht. Für unser Beispiel wären „marc21“ und „mab2“ denkbare Präfixe. Das Element *record* würde dann entweder mit <marc21:record> oder mit <mab2:record> bezeichnet. Durch diese Definition weiß der Leser bzw. das Verarbeitungsprogramm, ob es sich um das Element *record* aus der MAB2- oder jenes aus der MARC21-Formatdefinition handelt.

In Beispiel 5 wurde dieses Konzept bereits verwendet. Das Attribut `xmlns` legt die im Dokument verwendeten Namespaces fest.

`xmlns:xyz = „Namespace A“` bedeutet, dass alle Elemente und Attribute des Dokumentes, die den Präfix `xyz` verwenden (z.B. `xyz:record`) zum Namespace *A* gehören. Steht `xmlns` ohne Suffix, (`xmlns = „Namespace B“`) dann bedeutet dies, dass alle Elemente und Attribute ohne Präfix zum Namespace *B* gehören. Dies ist der sog. Default-Namespaces des Dokumentes. In Beispiel 5 ist das „`http://www.w3schools.com`“.

Der gleiche Satz im Format MABxml-1:

```

<datensatz typ="h" status="n" mabVersion="M2.0">
  <feld nr="001" ind="">187056</feld>
  <feld nr="002" ind="a">19890420</feld>
  <feld nr="003" ind="">20021120131909</feld>
  <feld nr="025" ind="z">187056</feld>
  <feld nr="030" ind="">b|1dcz|z|||37</feld>
  <feld nr="036" ind="a">DE</feld>
  <feld nr="037" ind="a">de</feld>
  <feld nr="050" ind="">|||||||g</feld>
  <feld nr="052" ind="">p||||||z|||||||</feld>
  <feld nr="070" ind="">9000</feld>
  <feld nr="070" ind="a">292</feld>
  <feld nr="070" ind="b">1200</feld>
  <feld nr="331" ind="">Journal of neurology für Testzwecke und Validation Lokaldaten Heise</feld>
  <feld nr="335" ind="">official journal of the European Neurological Society</feld>
  <feld nr="341" ind="">Zeitschrift für Neurologie</feld>
  <feld nr="370" ind="a">Organ d. Deutschen Gesellschaft für Neurologie u. d. Deutschen Gesellschaft
für Neurochirurgie</feld>
  <feld nr="370" ind="a">Zeitschrift für Neurologie. <ns>1974-</ns></feld>
  <feld nr="370" ind="a">Insert for neurologists</feld>
  <feld nr="376" ind="b">JNRYA</feld>
  <feld nr="376" ind="b">ZSNUA</feld>
  <feld nr="376" ind="">J. Neurol.</feld>
  <feld nr="376" ind="">J. Neurol. (Berlin)</feld>
  <feld nr="405" ind="">207.1974 -</feld>
  <feld nr="410" ind="">Berlin ; Heidelberg [u.a.]</feld>
  <feld nr="412" ind="">Springer</feld>
  <feld nr="425" ind="b">1975</feld>
  <feld nr="507" ind="">Zusatz teils: Organ d. Deutschen Gesellschaft für Neurologie u. d. Deutschen
Gesellschaft für Neurochirurgie</feld>
  <feld nr="524" ind="">Ungezählte Beil.: Insert for neurologists; Supplement</feld>
  <feld nr="537" ind="">16!NA für Internetausg.</feld>
  <feld nr="542" ind="a">ISSN 0340-5354</feld>
  <feld nr="542" ind="a">ISSN 0012-1037</feld>
  <feld nr="542" ind="a">ISSN 0939-1517</feld>
  <feld nr="652" ind="a">
  <uf code="a">Computerdatei im Fernzugriff</uf>
  </feld>
  <feld nr="700" ind="z">|870</feld>
</datensatz>

```

3.2 Die Komponenten einer MABxml-Spezifikation

Das Format MABxml wird durch zwei Dokumente spezifiziert:

- das *MABxml-Schema*
- und die *Übertragungsregeln* zwischen MAB2 und MABxml

Die Übertragungsregeln sind notwendig, weil für MABxml im Wesentlichen dieselben Regeln gelten sollen wie für MAB2. D.h. statt einer neuen detaillierten Formatbeschreibung bzw. einem komplexen Schema, das genaue feldspezifische Vorgaben macht, sollen die bisherigen MAB2-Regeln durch einige einfache Übertragungsregeln für MABxml nutzbar gemacht werden.

Das MABxml-Schema gibt lediglich einen groben syntaktischen Rahmen vor, sagt aber beispielsweise nur wenig über die semantischen Aspekte des Formates aus.

3.3 Das MABxml-1-Schema

3.3.1 Namespace

Der Namespace von MABxml-1 ist zurzeit:

<http://www.ddb.de/professionell/mabxml/mabxml-1.xsd>

Dies ist auch gleichzeitig die URL, unter der das Schema zu finden ist. In Kürze wird der Namespace durch eine URN [19] ersetzt.

3.3.2 Struktur eines Datensatzes

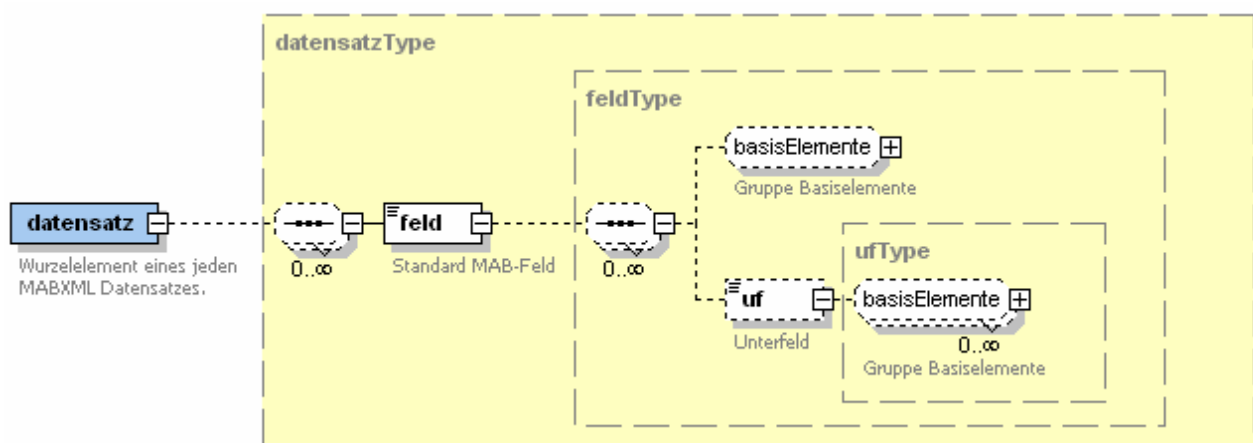


Abb. 1: Struktur eines MABxml-Datensatzes¹

¹ (Grafik erzeugt mit XMLSpy® (www.altova.de))

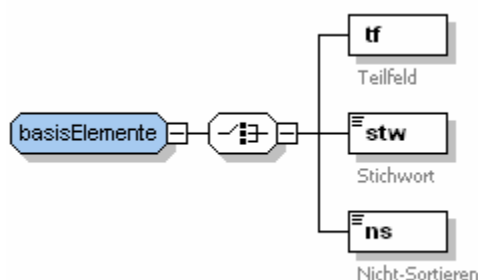


Abb. 2: Die Basiselemente von MABxml²

Abb. 1 zeigt die Struktur eines MABxml-Datensatzes. Der Inhalt eines Datensatzes wird im Element *datensatz* abgelegt. Das Element *datensatz* verfügt u.a. über die Attribute *typ*, *status* und *mabVersion* zur Speicherung von Informationen aus der Satzkenung (mehr dazu in Abschnitt 3.4.2). Jeder Datensatz (Element „*datensatz*“) enthält eine beliebige Anzahl an variablen Feldern (Element „*feld*“). Das Element *feld* enthält die Daten eines Feldes. Die Feldnummer wird im Attribut „*nr*“, der Indikator im Attribut „*ind*“ abgelegt. Ein Feld kann eine beliebige Mischung aus den folgenden Komponenten enthalten:

1. **Text**, d.h. Inhaltsdaten, wie beispielsweise einen Namen, Code oder eine Identifikationsnummer.

Beispiel 6 Verwendung des Elements *feld* zum Beschreiben von Feldern

MAB2	MABxml Ebene 1
089_Gesamtschule	<pre><feld nr="089" ind=" " > Gesamtschule </feld></pre>

1. Die Elemente *tf*, *stw*, und *ns* (dies sind die Elemente der Gruppierung „Basiselemente“, vgl. Abb. 2). Diese ersetzen die MAB-Steuerzeichen Teilfeldtrennzeichen, Stichwortzeichen und Nicht-Sortierzeichen.

Beispiel 7 Verwendung von *tf* als Ersatz für das Teilfeldtrennzeichen

MAB2	MABxml Ebene 1
089_Schuljahr 10. #Erweiterungskurs. #Hauptbd.	<pre><feld nr="089" ind=" " > Schuljahr 10.<tf/>Erweiterungskurs<tf/>Hauptbd. </feld></pre>

² (Grafik erzeugt mit XMLSpy® (www.altova.de))

3. Das Element *uf*. Dieses Element ersetzt das Unterfeldkennzeichen. Es umschließt den Inhalt des Unterfeldes. Der Unterfeld-Code wird nicht zusammen mit dem Inhalt, sondern im Attribut *code* abgelegt. Genauso wie das Element *feld* darf das Element *uf* neben den Inhaltsdaten auch die Unterelemente *ns*, *stw* und *tf* enthalten.

Beispiel 8 Verwendung von *uf* zum Beschreiben von Unterfeldern

MAB2	MABxml Ebene 1
652a\$aDiskette	<feld nr="652" ind="a"> <uf code="a">Diskette</uf> </feld>

3.3.3 Obligatorische Attribute

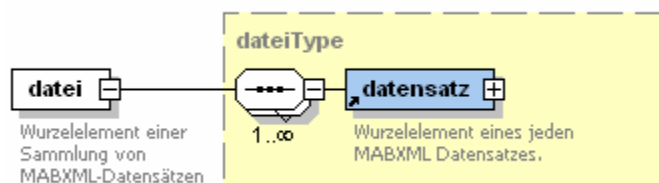
Die folgende Tabelle listet die *obligatorischen* Attribute und deren Bedeutung auf.

Element	Attribut	Bedeutung	Vorkommen
datensatz	typ	Typ des Datensatzes (MAB2-Satzkennung, Pos. 23), Aufzählungstyp mit den Werten: h, y, u, v, p, t, k,...	obligatorisch
	status	Status des Datensatzes (MAB2-Satzkennung, Pos. 5), Aufzählungstyp mit den Werten: c, d, n, p, u, v	obligatorisch
	mabVersion	Die den Übertragungsregeln zugrunde liegende MAB-Version (Satzkennung Pos. 6-9). Wird zur Zeit konstant durch den Wert „M2.0“ (d.h. MAB2) belegt.	obligatorisch
feld	nr	die Feldnummer eines MAB2-Feldes	obligatorisch
	ind	der Indikator eines MAB2-Feldes	obligatorisch
uf	code	der Code eines Unterfeldes. Das Unterfeldkennzeichen wird hierbei weggelassen	obligatorisch

Tabelle 1: Obligatorische Attribute

3.3.4 Sammlungen von Datensätzen

Es ist auch möglich mehrere Datensätze in einer XML-Datei abzulegen. Hierzu wurde das Element „*datei*“ definiert. Es fungiert als Container für beliebig viele *datensatz*-Elemente. Ein solches Element ist notwendig, da jede gültige XML-Datei genau ein Wurzelement, d.h. ein Element, das alle anderen Elemente als Unterelement enthält, haben muss.

Abb. 3: Das Element „datei“³

Beispiel 9

```
<datei>
  <datensatz typ="h" status="c" mabVersion="M2.0">
    ...
  </datensatz>
  <datensatz typ="u" status="n" mabVersion="M2.0">
    ...
  </datensatz>
  <datensatz typ="h" status="n" mabVersion="M2.0">
    ...
  </datensatz>
</datei>
```

3.3.5 Optionale Attribute

Neben den beschriebenen obligatorischen Attributen verfügt MABxml-1 auch noch über einige optionale Attribute. Diese sind nicht für die Übertragung von MAB-Daten auf XML notwendig, können aber für Personen, die in erster Line mit MABxml arbeiten wollen und nicht mit MAB2, die Auswertung von Datensätzen erleichtern.

Element	Attribut	Bedeutung	Vorkommen
datei	dateiart	Art der Datei. Aufzählungstyp mit den Werten: TITEL, PND, GKD, SWD, NOTAT, LOKAL, ADRESS	optional
datei, datensatz, feld, uf	id	Attribut vom Typ xsd:ID. Dieses Attribut ermöglicht es dem Autor eines XML-Dokumentes, einem Element eine ID, die innerhalb des Dokumentes eindeutig ist, zuzuordnen.	optional

Das MAB2-Format sieht für Titeldaten, Normdaten und Lokaldaten verschiedene Austauschdateien vor. Die Verwendung des Attributs *dateiart* wird empfohlen, wenn eine MABxml-Datei nur Datensätze einer bestimmten Art enthält.

³ (Grafik erzeugt mit XMLSpy® (www.altova.de))

Das Attribut *id*, über das jedes Element und Unterelement verfügt, dient der Identifikation eines einzelnen Elementes. Dadurch wird es dem Autor einer XML-Datei möglich, an einer anderen Stelle des Dokumentes auf dieses spezielle Element zu verweisen. Interessant kann diese Eigenschaft werden, wenn ein MABxml-Datensatz im Kontext eines allgemeineren XML-Dokumentes eingesetzt wird. Beispiel 10 zeigt skizzenhaft einen solchen Anwendungsfall. Mehr zu diesem Thema gibt es unter [10].

Beispiel 10 Pseudo-Code

```
<dokument>

  <!-- verschiedene Titelsätze des Autors Daniel Haase -->
  <mab:datensatz id="12340" typ="h" status="n" mabVersion="M2.0">
    ...
  </mab:datensatz>
  <mab:datensatz id="12341" typ="h" status="n" mabVersion="M2.0">
    ...
  </mab:datensatz>
  ...

  <!-- Personendatensatz des Autors Daniel Haase (nicht PND) -->
  <autor>
    <name>Haase, Daniel</name>
    <geb>10-03-1960</geb>
    <werke>
      <!-- Verweise auf die oben angegebenen Titelsätze -->
      <titelsatz href="12340"/>
      <titelsatz href="12341"/>
      ...
    </werke>
  </autor>

</dokument>
```

3.4 Übertragungsregeln

3.4.1 Allgemein

Zur Konversion von MAB2-Datensätzen nach MABxml wurde eine einfache Regelliste erstellt: die Übertragungsregeln MAB/MABxml-1 [22].

Die Übertragungsregeln für MABxml-1 sind einfach: Wenn nicht anders vermerkt, werden die Felder und Attribute identisch zu MAB2 eingesetzt und belegt. Die in MAB2 durch die Steuerzeichen „Satzendezeichen“, „Feldendezeichen“ und „Unterfeldkennzeichen“ gegebene Datensatzstruktur wird in eine hierarchische XML-Struktur übertragen: Der Inhalt eines Datensatzes wird durch das *datensatz*-Tag umschlossen, der Inhalt eines Feldes durch das *feld*-Tag und der Inhalt eines Unterfeldes durch das *uf*-Tag. Die oben genannten Steuerzeichen fallen weg.

3.4.2 Satzkennung

Die beiden Attribute des *Datensatz*-Elementes ersetzen das MAB2-Satzkennungsfeld. Alle anderen MAB2-Satzkennungsfelder sind entweder konstant oder für ein XML-Dokument irrelevant (eine ausführlichere Begründung folgt in 4.1). Bei einer Konversion von MABxml nach MAB2 müssen diese fehlenden Felder berechnet werden.

3.4.3 Unterfelder

Eine weitere Ausnahme bildet die Behandlung von Unterfeldern. Die Übertragung von MAB2 kann wie folgt formuliert werden: Kommt in einem MAB-Feld ein Unterfeld vor, dann:

- verwirf das Unterfeldkennzeichen
- umschließe den Inhalt des Unterfeldes (mit Ausnahme des Unterfeld-Codes) durch die Tags `<uf code=""> ... </uf>`
- und lege den Code des Unterfelds im Attribut *code* ab.

3.4.4 Verbleibende Steuerzeichen

Die verbleibenden MAB2-Steuerzeichen „Nichtsortierzeichen“, „Teilfeldtrennzeichen“ und „Stichwortzeichen“ werden durch die Tags `<ns></ns>` (Nichtsortierzeichen), `<stw></stw>` (Stichwortzeichen) und das leere Element `<tf/>` (Teilfeldtrennzeichen) ersetzt.

Beispiel 11 unvollständiger Datensatz:

```
<datensatz typ="h" status="n" mabVersion="M2.0" >
  <feld nr="089" ind=" " >Schuljahr 10.<tf/>Erweiterungskurs.<tf/>Hauptbd.</feld>
  <feld nr="100" ind="b" >Ernst, Klaus</feld>
  <feld nr="200" ind=" " >Institut für Geschichtswissenschaft</feld>
  <feld nr="652" ind="a" >
    <uf code="a" >Diskette</uf>
  </feld>
</datensatz>
```

3.4.5 Lesbarkeit (Einrückungen und Zeilenwechsel)

Um das MABxml-Dokument lesbarer zu gestalten, können vor und nach den Elementen *datei*, *datensatz*, *feld* und *uf* beliebig viele Zeilenwechsel und Einrückungen durch TAB- oder Leerzeichen eingefügt werden.

Empfohlen wird die folgende Formatierung: Alle Felder, Unterfelder und Datensätze werden durch einen Zeilenwechsel davor und danach voneinander und von ihren Oberelementen getrennt. Zusätzlich werden sie durch Einrückung (durch TAB- oder Leerzeichen) gegenüber dem jeweiligen Oberelement optisch abgesetzt. Elemente derselben Hierarchieebene werden hierbei gleich weit eingerückt.

Diese Formatierung wurde ohne nähere Erläuterung in allen Beispielen angewandt.

Achtung: Alle Programme, die MABxml-Dokumente verarbeiten, müssen dementsprechend alle *Blank-*, *TAB-*, *Carriage-Return-* und *Line-Feed-*Zeichen, die vor und nach den Elementen *datei*, *datensatz*, *feld* und *uf* vorkommen ignorieren. Sie sind nicht inhaltlicher Bestandteil des Datensatzes.

3.5 Zeichenvorrat und Zeichenkodierung

Als Zeichenvorrat steht in XML1.0 das Character-Set ISO 10646 [13] (Unicode [12])⁴ zur Verfügung. Ausgenommen sind hierbei der größte Teil der ersten 30 Zeichen (darunter auch das MAB2-Feldendezeichen, Satzendezeichen und Unterfeldkennzeichen – hierzu aber später mehr).

Die Zeichenkodierung – also der Mechanismus Zeichen in ein Bit-Muster zu kodieren – kann für jedes Dokument variieren. Allerdings muss jeder XML-Prozessor zumindest mit den Kodierungen UTF-8 und UTF-16 umgehen können. D.h. dass man nur unter Verwendung dieser beiden Kodierungen mit einer breiten Unterstützung durch Dritt-Software rechnen kann.

⁴ Der Zusammenhang zwischen ISO 10646 und Unicode ist etwas komplizierter. Ich verweise diesbezüglich auf [14].

Dennoch bietet XML die Möglichkeit der Verwendung anderer Kodierungen. Der Name der Kodierung sollte dann dem Attribut „encoding“ zugewiesen werden, da sonst das Verarbeitungsprogramm standardmäßig von der Kodierung UTF-8 ausgeht (vgl. Beispiel 12).

Beispiel 12

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

In MAB2 können verschiedene Zeichenkodierungen verwendet werden (siehe MAB2-Feld 030, Position 3), mit Abstand am verbreitetsten sind allerdings die Kodierung von „*ISO 646 + ISO 5426*“ sowie die Kodierung von „*MAB-Diskette*“ (DOS-Zeichensatztable 850).

Der MAB2-Zeichenvorrat entspricht den Zeichen aus ISO646 + ISO 5426-1983 oder im Falle von „MAB-Diskette“ den ASCII-Zeichen. ISO 10646 ist eine Obermenge dieses Zeichenvorrats⁵. *ISO 646 + ISO 5426* und die *DOS-Zeichensatztable 850* können also als spezielle Kodierungen von *ISO 10646* aufgefasst und im Rahmen von XML verwendet werden.

Will man die beiden genannten MAB2-Zeichenkodierungen auch für MABxml-Datensätze verwenden, sollten im XML-Header die in Tabelle 2 aufgeführten „encoding“-Bezeichnungen verwendet werden.

encoding	steht für
x-MAB	ISO 646 + ISO 5426 mit den im Anhang A der MAB2-Spezifikation angegebenen Ausnahmen.
IBM850	DOS-Zeichensatztable 850 (Zeichensatz MAB-Diskette), offizielle Bezeichnung nach IANA [16].

Tabelle 2: „encoding“-Bezeichnungen für MAB2-Zeichenkodierungen

Beispiel 13

```
<?xml version="1.0" encoding="x-MAB"?>
```

Der Präfix „x-“ wurde im Falle von „x-MAB“ aufgrund der folgenden Empfehlung vorangestellt: „It is recommended that character encodings registered (as charsets) with the Internet Assigned Numbers Authority [16], other than those just listed, be referred to using their registered names; other encodings should use names starting with an "x-" prefix.“ [1]

⁵ Die einzige Ausnahme bildet hier das Trema- und das Umlautzeichen und alle ihre Kombinationen mit Grundbuchstaben. Diese sind in ISO 10646 jeweils nur noch durch ein einzelnes Zeichen vertreten. Umlautzeichen und Trema unterscheiden sich allerdings nur in ihrer Semantik, sind optisch aber identisch. [15]

Warnung: Es sei deutlich darauf hingewiesen, dass die Verwendung der Kodierungen x-MAB einer breiten Unterstützung durch Software-Tools im Wege steht. Es empfiehlt sich, für XML-Dokumente grundsätzlich die Kodierung UTF-8 zu verwenden, denn der XML-Standard schreibt vor, dass XML-Programme diese Kodierung unterstützen **müssen**. Andere international gebräuchliche Kodierungen wie „ISO-8859-1“ und „IBM-850“ (also MAB-Diskette) werden häufig ebenfalls unterstützt und können im Zweifelsfall praktische Alternativen zu UTF-8 sein.

4 FAQ rund um MAB + XML

4.1 Darf „normales“ MAB2 über OAI transportiert werden?

Das OAI-Protokoll nutzt XML als Syntax. D.h. die Anfragen und Antworten sind XML-Dokumente. Daher muss der in der Antwort zurückgelieferte Datensatz XML-konform sein. Das ist aber für MAB2 leider nicht der Fall (siehe Frage 4.2).

4.2 Darf der Zeichensatz von MAB oder MAB-Diskette für OAI verwendet werden?

Die OAI-Spezifikation gibt UTF-8 als einzige legale Zeichenkodierung vor. MAB-Datensätze müssen für den Transport über OAI also in jedem Fall nach Unicode konvertiert und dann mit UTF-8 kodiert werden.

4.3 Ist MAB2 XML-konform?

MAB2 benutzt leider Zeichen, die mit XML1.0 nicht vereinbar sind. Es handelt sich hierbei um das Satzende-, Feldende- und Unterfeldeinleitungszeichen. Diese Zeichen (genauso wie die meisten der ersten 32 Unicode-Zeichen) sind keine legalen Zeichen in XML1.0.

MAB2-Datensätze also einfach mit in ein XML-Tag zu packen – z.B. <datensatz>(MAB2-Datensatz)</datensatz> führt zu keinem korrekten XML-Dokument.

4.4 Warum hat MABxml kein Satzkelement?

Anders als MARCXML, das ein *leader*-Element enthält, gibt es in MABxml kein Satzkelement. Stattdessen transportiert MABxml einen Teil der Informationen über die beiden *datensatz*-Attribute *typ*, *status* und *mabVersion*. Die übrigen Satzkelementpositionen wurden aus den folgenden Gründen weggelassen:

- Die Länge eines einzelnen Satzes spielt für die Verarbeitung in der heutigen Zeit keine große Rolle mehr und kann bei Bedarf problemlos von einem Rechner ermittelt werden.
- Eine Datenanfangsadresse ist für das XML-Format irrelevant.
- Die Indikatorlänge wird bereits implizit im XML-Schema festgesetzt.
- Alle anderen Werte sind Konstanten.

In MARCXML wurde ein ähnlicher Ansatz verfolgt, aber nicht konsequent durchgeführt: Auch hier gibt es optional die Möglichkeit, den Record-Type als Attribut des Elementes *record* anzugeben, z.B.: `<record type="Bibliographic"> ...</record>`

Zusätzlich steht der Record-Type dann aber noch im leader-Element von record. Desweiteren wurde die Definition des Leaders verändert: "MARC structural elements, such as the length of field and starting position of field data in directory entries are not needed in the XML record. Leader data positions not needed in the XML environment are retained as place holders or carried as blanks [3]."

Die Satzkennung gänzlich über Attribute umzusetzen und überflüssige Elemente auszulassen, erschien als die sauberere Lösung.

Referenzen

- [1] *Extensible Markup Language (XML) 1.0 (Second Edition)*. W3C. Internet: <http://www.w3.org/TR/REC-xml>.
- [2] *XML Schema Homepage*. W3C. Internet: <http://www.w3.org/XML/Schema>.
- [3] *MARCXML Homepage*. Library of Congress. Internet: <http://www.loc.gov/standards/marcxml/>
- [4] *MARC STANDARDS*. Library of Congress. Internet: <http://www.loc.gov/marc/>
- [5] XSL Transformations (XSLT), Version 1.0. W3C. Internet : <http://www.w3.org/TR/1999/REC-xslt-19991116>
- [6] *XML Schema Best Practices Homepage*. Xfront. Internet: <http://www.xfront.com/BestPracticesHomepage.html>
- [7] *Global versus Local*. Xfront. Internet: <http://www.xfront.com/GlobalVersusLocal.pdf>
- [8] *RAC und MAB oder AACR und MARC?*. Renate Gömpel, Die Deutsche Bibliothek. 29.August 2002. Internet: http://www.ddb.de/professionell/pdf/sbb_workshop_290802.pdf
- [9] Open Archives Initiative Homepage. Internet: <http://www.openarchives.org/>
- [10] *XML Linking Technologies*. Eric van der Vlist. 04.Oktober 2000. Internet: <http://www.xml.com/pub/a/2000/10/04/linking/>
- [11] SRW/U Homepage. ZING Initiative. Internet: <http://www.loc.gov/z3950/agency/zing/srw/background.html>
- [12] Unicode Homepage. Internet: <http://www.unicode.org>
- [13] ISO Online. Internet: <http://www.iso.org>
- [14] *FAQ – Unicode and ISO 646*. Unicode Homepage. Internet: http://www.unicode.org/faq/unicode_iso.html
- [15] *Zeichenkonkordanz MAB2-Zeichensatz – Unicode / ISO 10646*. R. Heuvelmann. Die Deutsche Bibliothek. Internet: http://www.ddb.de/professionell/pdf/mab_unic.pdf
- [16] IANA Charset Registry. <http://www.iana.org/assignments/character-sets>
- [17] *Namespaces in XML*. W3C. Internet: <http://www.w3.org/TR/REC-xml-names/>
- [18] W3Schools - Full Web Building Tutorials. Internet: <http://www.w3schools.com>
- [19] Persistent Identifier – eindeutige Bezeichner für digitale Objekte. Internet: <http://www.persistent-identifizier.de>
- [20] MABxml-Informationssseite. Internet: <http://www.ddb.de/professionell/mabxml.htm>
- [21] XML-Schema von MABxml-1. Internet: <http://www.ddb.de/professionell/mabxml/mabxml-1.xsd>
- [22] Übertragungsregeln MAB/MABxml-1. Internet: http://www.ddb.de/professionell/pdf/mabxml_1_uebertr.pdf