

IT-Standards für die amtliche Statistik

Table Markup Language (TabML)

Komponente TabML/Layout 1.0

Ein XML-basierter Dokumenttyp zur Beschreibung
statistischer Ergebnistabellen

Autor: Michael Schäfer, <mailto:michael.schaefer@destatis.de>

Status: Spezifikation

Stand: 19.11.2003

© Statistisches Bundesamt Wiesbaden, Deutschland

Table Markup Language (TabML) 1.0

Spezifikation

Beschreibung

TabML (Table Markup Language) ist eine auf [XML \(Extensible Markup Language\) 1.0](#) basierende Auszeichnungssprache für Tabellen. Im Vordergrund der Entwicklung steht der Einsatz in der amtlichen und beschreibenden Statistik. TabML soll darüberhinaus grundsätzlich überall dort Verwendung finden können, wo Tabellen maschinell erzeugt oder gelesen werden.

Dieses Dokument beschreibt u.a. die Elemente von TabML, ihre Attribute und deren Ausprägungen, ihren Inhalt und wie sie voneinander abhängen, und welche Anforderungen Anwendungen erfüllen müssen, die TabML-konforme Dokumente erzeugen oder verarbeiten.

Status dieses Dokuments

Dieses Dokument ist ein unvollständiger und noch nicht fehlerbereinigter Entwurf von TabML 1.0. Es kann daher nicht als Referenz verwendet werden. Insbesondere sind die Beschreibungen der Elemente und Attribute in der vorliegenden Form weder vollständig noch endgültig, und einige Aspekte von TabML werden noch nicht oder nur im Ansatz behandelt. Dazu zählen die Beschreibung der Tabellenmatrizen, die Integration von Graphiken und Stilattributen und einige Regeln für die Anwendungsentwicklung.

1. Inhaltsverzeichnis

1. Inhaltsverzeichnis	3
2. Einführung	6
2.1. Ausgangssituation.....	6
2.2. Lösungsansatz und Ziele	6
2.3. Randbedingungen, Einschränkungen	8
2.4. Verfügbarkeit.....	8
2.5. Implementierung	8
2.6. Namensräume	8
3. Das TabML-Tabellenmodell.....	10
3.1. Grundkonzept des Tabellenmodells.....	10
3.2. Layoutformat	10
3.2.1. Grundsätzliches	10
3.2.2. Struktur	11
3.3. Matrixformat	12
4. TabML Root.....	14
4.1. Beschreibung.....	14
4.2. Elemente.....	14
4.2.1. Wurzelement <code><tml:tabml></code>	14
5. TabML Tabelle	16
5.1. Beschreibung.....	16
5.2. Elemente.....	16
5.2.1. Element <code><tml:table></code>	16
6. TabML Layoutformat.....	17
6.1. Beschreibung.....	17
6.2. Layoutinformation	17
6.3. Funktionen des Layoutformates.....	17
6.4. Grundzüge des Layoutformats.....	17
6.4.1. Layoutformat vs. HTML 4.0.....	18
6.4.2. Strukturierung von Tabellen	19
6.4.2.1. Überblick	19
6.4.2.2. Elementklassen.....	19
6.4.2.3. Segmentierung.....	19
6.4.2.4. Horizontale Segmentierung, Bahnen	19
6.4.2.5. Tabellenfelder über Bahngrenzen	20
6.4.2.6. Seitengenerierung und –nummerierung.....	20
6.4.2.7. Überschrift, Tabellenkopf und Tabellenfuß	21
6.4.2.8. Fußnoten.....	22
6.4.2.9. Definition von Untertabellen	22
6.4.2.10. Werte in Tabellen	22
6.4.2.11. Repräsentation von Sonderzeichen	22
6.4.2.12. Verwendung von Kolonnenbeschreibungen.....	23
6.5. Layoutformat: Root	24
6.5.1. Element <code><lf:layoutFormat></code>	24
6.6. Kolonnenbeschreibung	25
6.6.1. Tabellen-, Bahn- und Kolonnenbreite	25
6.6.1.1. Relative Breitenangaben.....	25
6.6.1.2. Breiten ermitteln und berechnen	26
6.6.1.3. Die Anzahl der Kolonnen berechnen	27
6.6.1.4. Beispiele:	28

6.6.2. Tabelle der Elementtypen	30
6.6.3. Element <code><lf:column></code>	30
6.6.4. Element <code><lf:columnGroup></code>	31
6.6.5. Element <code><lf:columnStructure></code>	32
6.6.6. Element <code><lf:slice></code>	33
6.7. Stilattribute	34
6.7.1. Warum Stilattribute?	34
6.7.2. Verarbeitung von Stilattributen.....	34
6.7.3. Definition und Vererbung	35
6.7.4. Ausrichtung von Daten in Zellen	35
6.7.5. Tabelle der Elementtypen	36
6.7.6. Element <code><lf:styleDefinition></code>	36
6.7.7. Element <code><lf:styleDocumentDefaults></code>	37
6.7.8. Element <code><lf:styleElementDefaults></code>	38
6.7.9. Element <code><lf:style.Defaults></code>	40
6.7.10. Elemente <code><lf:style.elementtype></code>	41
6.7.10.1. Blockebene	41
6.7.10.2. Zellebene	42
6.7.10.3. Datenebene	43
6.7.10.4. Attribute.....	45
6.8. Blockelemente: vertikale Segmentierung	46
6.8.1. Beschreibung	46
6.8.2. Tabelle der Elementtypen	46
6.8.3. Element <code><lf:frame></code>	47
6.8.4. Element <code><lf:title></code>	49
6.8.5. Element <code><lf:axis></code>	50
6.8.6. Element <code><lf:body></code>	51
6.8.7. Element <code><lf:data></code>	52
6.8.8. Element <code><lf:foot></code>	53
6.8.9. Element <code><lf:fr></code>	54
6.8.10. Element <code><lf:head></code>	56
6.8.11. Element <code><lf:row></code>	57
6.8.12. Element <code><lf:subtitle></code>	59
6.9. Inline-Elemente: Tabellenzellen	60
6.9.1. Beschreibung	60
6.9.2. Tabelle der Elementtypen	60
6.9.3. Zuordnung von Zell-Elementtypen zu Block-Level-Elementtypen.....	60
6.9.4. Element <code><lf:b></code>	62
6.9.5. Element <code><lf:d></code>	63
6.9.6. Element <code><lf:f></code>	64
6.9.7. Element <code><lf:h></code>	65
6.9.8. Element <code><lf:r></code>	66
6.9.9. Element <code><lf:s></code>	67
6.9.10. Element <code><lf:v></code>	68
6.10. Datenelemente: Strukturierung von Zelleninhalt	69
6.10.1. Beschreibung	69
6.10.2. Tabelle der Elementtypen	69
6.10.3. Element <code><lf:bc></code>	70
6.10.4. Element <code><lf:br></code>	71

6.10.5. Element <lf:dt>.....	71
6.10.6. Element <lf:fId>.....	72
6.10.7. Element <lf:fRef>	73
6.10.8. Element <lf:fText>	74
6.10.9. Element <lf:pTag>	75
6.10.10. Element <lf:pNum>	76
6.10.11. Element <lf:sc>.....	77
6.11. Attributklassen	78
6.11.1. Beschreibung	78
6.11.2. Attribute der Kolonnenbeschreibung.....	78
6.11.3. Allgemeine Attribute von Blockelementen.....	78
6.11.4. Allgemeine Attribute von Tabellenzellen	78
6.11.5. Verarbeitung von Stilattributen steuern.....	78
6.11.6. Stilattribute	78
6.11.6.1. Rahmen	79
6.11.6.2. Farbe und Hintergrund	79
6.11.6.3. Textgestaltung	79
6.11.7. Ausrichtung von Text in einer Zelle.....	80

2. Einführung

2.1. Ausgangssituation

In den vergangenen Jahrzehnten stellten Großrechnersysteme die technische Basis für den weit überwiegenenden Teil der amtlichen statistischen Produktionssysteme. Zwar befindet sich auch hier schon seit längerem die IT-Landschaft mit zunehmender Geschwindigkeit im Umbruch, doch werden die meisten statistischen Tabellen – das Schlüsselprodukt der amtlichen Statistik – immer noch auf Großrechnern erzeugt, oft von Programmen, die schon existierten, als andere Plattformen noch keine Verwendung fanden. Konsequenterweise werden Tabellen dabei in Formaten erzeugt, die für die Verwertung auf Großrechnersystemen ausgelegt sind. Meist ist dabei der Ausdruck in Papierform die einzige Form der Verwertung.

Die bisher unternommenen Anstrengungen zielten i.d.R. auf eine andere Art der Weiterverarbeitung, nicht aber auf eine grundlegende Veränderung der Tabellenformate. Diese, für den Ausdruck mit Zeilen- und Laserdruckern „optimiert“, sind jedoch nur sehr schwer für andere Nachverarbeitungsarten geeignet. Meist ist es unmöglich, solche Tabellen mit generischen Programmen zu verarbeiten, da alle Informationen über die Struktur und den Inhalt (Metadaten) in den Anwendungen verborgen sind.

Gleichzeitig werden an die Repräsentation und Verfügbarkeit statistischer Ergebnisse höhere Anforderungen gestellt. Tabellen sollen schneller und in einer Vielzahl unterschiedlicher Formate zur Verfügung stehen, nicht nur beim Endkunden, sondern auch bei den internen Abnehmern wie den Fachabteilungen, die die Weiterverarbeitung zunehmend in eigener Regie betreiben, um wiederum den eigenen Anforderungen oder denen ihrer Kunden zu genügen. Das Produkt Tabelle tritt dabei in zwei Formen auf: als Endprodukt für die interne oder externe Präsentation statistischer Ergebnisse und als vorwiegend internes oder technisches Zwischenprodukt, z.B. für die Speicherung in Informationssystemen.

2.2. Lösungsansatz und Ziele

Die Lösung zu diesem Problem kann nicht darin bestehen, vorhandene Anwendungen zu erweitern, um Tabellen in allen möglichen, letztendlich wieder anwendungsspezifischen Formaten zu erzeugen. Sie liegt darin, Tabellen so zu erzeugen, daß sie von generischen Anwendungen in solche Formate konvertiert werden können, also in einem quasi „neutralen“ Format. Dies reduziert die Aufwände der Softwarepflege dauerhaft auf ein Minimum und erweitert umgekehrt die Möglichkeiten, weitere Formate bei Bedarf zu bedienen. Zugleich erhöht dies die Flexibilität bei der Tabellenproduktion, Damit dies erreicht werden kann, müssen die Tabellen mit den in den Anwendungen verfügbaren Metadaten ausgestattet werden, und zwar in einem einheitlichen Format. Dieses Format zu formulieren ist das primäre Ziel von TabML. Darüberhinaus sollen folgende Einzelziele erreicht werden:

- Allgemeine Erweiterung der Verarbeitungsmöglichkeiten von Tabellen.

- Interaktive Verarbeitung von Tabellen mit einem hohem Grad an Sicherheit und Unterstützung.
- Leichte Konvertierung von Tabellen in verschiedenste Formate .
- Integration von Tabellen in eine Metadatenverwaltung.

2.3. Randbedingungen, Einschränkungen

Die Entwicklung (und der Einsatz) von TabML geschieht unter zwei gegensätzlichen Randbedingungen: vielen Anwendungen den Einsatz von TabML zu ermöglichen bei gleichzeitig großer Beschreibungstiefe. Dies erfordert, daß viele Elemente optional bleiben müssen. Andererseits werden, was die *maschinelle* Tabellenerstellung betrifft, nur wenige Anwendungen – wenn überhaupt – das volle Potential von TabML ausschöpfen können, indem sie alle Elemente *automatisch* bis in die feinsten Verästelungen beschreiben. Dies gilt z.B. für die allgemeinen Metadaten und die Stilattribute des Layoutformates. Es ist davon auszugehen, daß in vielen Fällen eine Nachbearbeitung gewünscht oder notwendig ist, abhängig u.a. davon, wie individuell eine Tabelle gestaltet werden soll. Trotzdem ist auch bei automatischer Weiterverarbeitung eine erhebliche Verbesserung gegenüber den bisherigen Verfahren zu erwarten, da i.d.R. ein deutliches Plus an Metadaten für generische Anwendungen nutzbar sein wird. Andererseits ist z.B. eine Anwendung denkbar, die völlig unstrukturierte Tabellen aus Großrechneranwendungen einliest und interaktiv gesteuert TabML-Strukturen und -Metadaten hinzufügt.

2.4. Verfügbarkeit

2.5. Implementierung

TabML 1.0 wird parallel als XML-DTD und als XML Schema implementiert. Die XML-DTD verwendet vordefinierte Präfixe um die Bindung der Elementtypen an ihre Namensräume zu simulieren (s. 2.6)

Für die semantische Implementierung von TabML, also die Benamung der Elemente, Attribute und Attributwerte, wurde die englische Sprache gewählt, um den möglichen Austausch von Dokumenten mit ausländischen Institutionen (z.B. den nationalen statistischen Ämtern oder Eurostat) zu erleichtern.

2.6. Namensräume

TabML 1.0 ist ein aus modularen Komponenten aufgebauter Dokumenttyp und verwendet Elementtypen aus mehreren [Namensräumen](#). TabML-konforme Dokumente müssen Element- und ggf. Attributnamen entsprechend den durch das [Worldwide Web Consortium \(W3C\)](#) veröffentlichten Empfehlungen für [Namensräume](#) ihrem jeweiligen Namensraum eindeutig zuordnen.

[Namensräume](#) werden nicht durch XML-DTDs unterstützt. Für die Validierung von TabML-Dokumenten per DTD wird deshalb eine DTD bereitgestellt, in der alle betroffenen Namen durch ein vorgegebenes Präfix erweitert sind. Es gilt als *best practice* und wird deshalb empfohlen, nur diese vorgegebenen Präfixe zu verwenden, auch wenn Dokumente mittels [XML Schema](#) oder einer anderen Methode, die [Namensräume](#) unterstützt, validiert werden.

TabML-1.0 verwendet folgende Namensräume und vordefinierten Präfixe:

Komponente	Namensraum-URI	Präfix
TabML	http://www.destatis.de/schema/tabml/1.0	tml
TabML-Layoutformat	http://www.destatis.de/schema/tabml-layout/1.0	lf

Namensraum-URI und Präfix sind in den Elementtypbeschreibungen enthalten.

Ein TabML-Dokument sollte folgende Namensraumdeklarationen enthalten; es ist *best practice*, keinen Defaultnamensraum zu verwenden. Es wird außerdem empfohlen, alle Namensraumdeklarationen für das gesamte Dokument im Wurzelement `<tml:tabml>` vorzunehmen:

```
<tml:tabml xmlns:tml="http://www.destatis.de/schema/tabml/1.0"
           xmlns:lf="http://www.destatis.de/schema/tabml-layout/1.0"
           version="1.0"
           >
```

Die TabML-DTD erzwingt die Deklaration im Wurzelement, indem sie alle Namensraumattribute als erforderlich (`#REQUIRED`) erklärt.

3. Das TabML-Tabellenmodell

3.1. Grundkonzept des Tabellenmodells

Kern des TabML-Konzepts ist die Idee, daß eine Tabelle in zwei „Formaten“ existieren kann, die zwei Zuständen während des Prozesses der Tabellenerstellung entsprechen: der Erzeugung der Tabellenmatrix im Speicher und der Ausgabe der Matrix in einem druckfähigen Format, also mit komplettem Layout. In vielen Fällen weicht das Layout einer Tabelle auch in Bezug auf die Werte inhaltlich von der Tabellenmatrix ab, beispielsweise, wenn die Matrix Hilfszellen für eine Geheimhaltung enthält. Mit TabML sollen beide Formate – als [Matrixformat](#) und [Layoutformat](#) bezeichnet – dargestellt werden können, wobei mit jedem Format durchaus unterschiedliche Ziele verfolgt werden. Prinzipiell kann eine Tabelle in beiden Formaten in einem TabML-Dokument gespeichert werden, wobei beachtet werden muß, daß dies i.d.R. die Zwischenpufferung eines der Formate erfordert. Ein anderes Problem ist die Entstehung von Inkonsistenzen zwischen den Formaten, wenn Änderungen nicht parallel erfolgen.

3.2. Layoutformat

3.2.1. Grundsätzliches

Durch das Layout erhält eine Tabelle ein Erscheinungsbild, dem besonders bei Veröffentlichungstabellen konkrete Vorstellungen und Gestaltungsabsichten zugrunde liegen. Meist jedoch gehen im Layout - vor allem bei der traditionellen Erzeugung druckfähiger Tabellen auf Großrechnersystemen - wichtige, oft fast alle, Strukturinformationen und Metadaten verloren. Die Folge ist, daß z.B. eine druckfertige Tabelle nur schwer in andere Formate konvertiert werden kann, schon gar nicht mit Hilfe generischer Programme. Das Layoutformat von TabML soll helfen, diese Beschränkungen zu überwinden, indem es einerseits versucht, die vollständige Layoutinformation zu bewahren, und andererseits die Tabelle so zu strukturieren und mit Metadaten auszustatten, daß generische Programme sinnvoll auf die Bestandteile der Tabelle und ihre Attribute zugreifen können, auch, um die ursprüngliche Gestaltung zu verändern.

Hauptzweck des Layoutformates ist es, die Konvertierung in verschiedene Präsentationsformate, wie PDF, RTF und HTML, zu erleichtern, ohne jeweils den kompletten, oft komplexen und tabellenspezifischen Prozeß der Layoutgenerierung wiederholen zu müssen. Das Layoutformat besitzt so einen eher statischen Charakter. Sein Zweck ist nicht, umfangreiche Manipulationen am Inhalt einer Tabelle, speziell das Entfernen oder Hinzufügen von Bestandteilen, zu erlauben; dem widerspricht auch, daß mit einem Layout versehene Tabellen den Charakter eines Endproduktes haben. Im Vordergrund steht die Möglichkeit, „fertige“ Tabellen schnell, sicher, mit geringem Aufwand und durch generische Anwendungen (Konverter) in verschiedenen Formaten und auf verschiedenen Medien speichern oder präsentieren zu können. Darüber hinaus bietet auch das Layoutformat eingeschränkte Möglichkeiten, Informationen für die Auswertung von Tabellen bereitzustellen.

3.2.2. Struktur

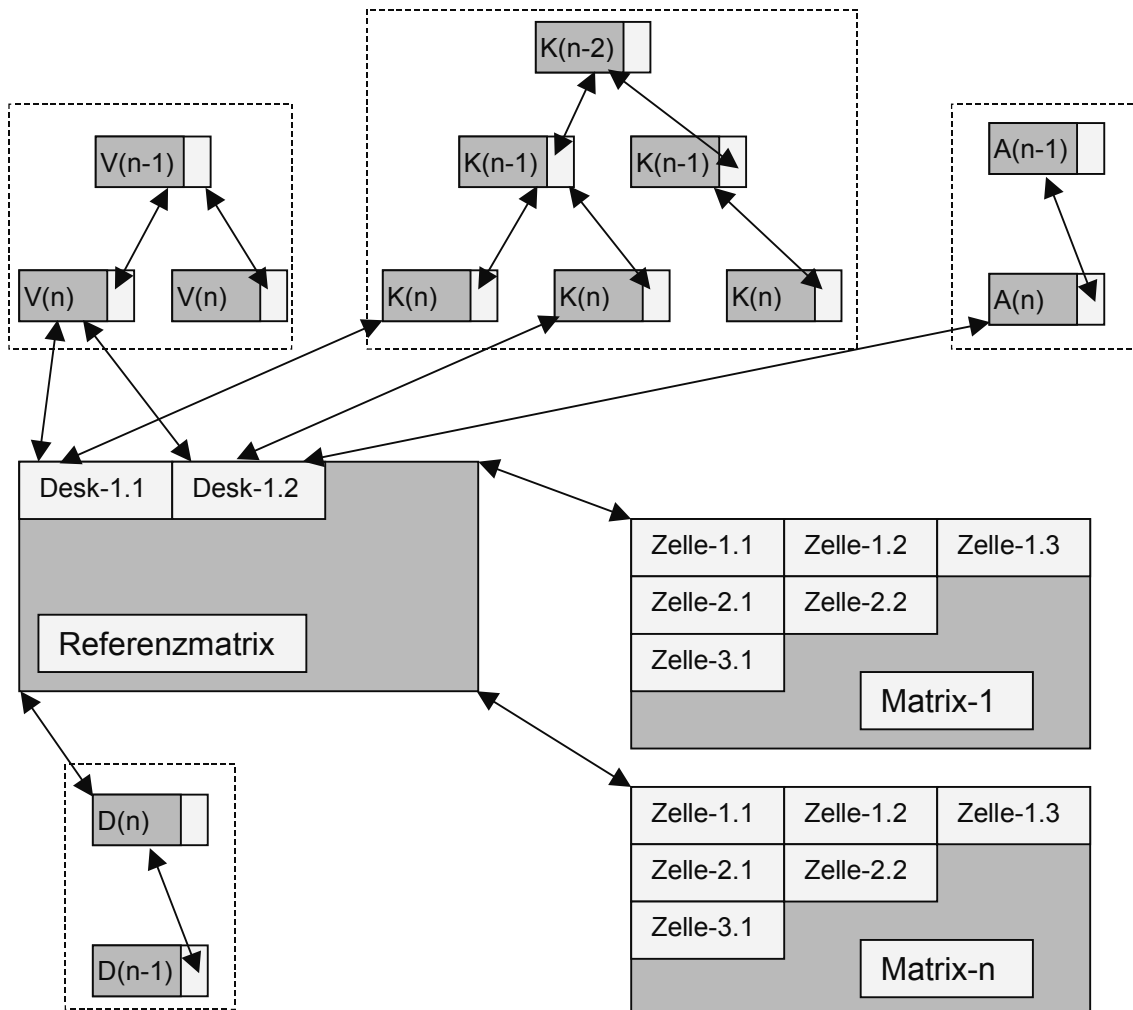
Das Layoutformat beruht – in der aktuellen Version – auf der traditionellen Tabellengestaltung, welche für die Veröffentlichung mittels Druckmedien gedacht ist, doch auf graphische Elemente (weitgehend) verzichtet, und den technischen Bedingungen der Produktionssysteme, die in der Großrechnerwelt entstanden sind. Die einzigen hier noch vorhandenen Strukturinformation ist die horizontale Einteilung in Seiten und Zeilen. Diese werden im Layoutformat durch weitere Strukturelemente und die vertikale Einteilung in Kolonnen ergänzt. Kolonneneinteilungen werden getrennt von den Tabellenseiten beschrieben und von diesen referenziert. Da beliebig viele Kolonneneinteilungen erlaubt sind, können sie seiten- und abschnittsweise variieren, was die Abbildung sogenannter „gestürzter“ Tabellen ermöglicht.

Eine Tabellenseite wird in hierarchisch angeordnete logische Abschnitte unterteilt. Darunter ist ein zu HTML analoges kombiniertes Zeilen- und Zellenmodell implementiert. Eine Zelle wird in einer Zeile definiert und kann sich über mehrere Zeilen und Kolonnen erstrecken. Es stehen mehrere Elementtypen zur Definition von Zellen zur Verfügung, die jeweils einen anderen Basisdatentyp repräsentieren, wie z.B. einen Vorspaltext oder einen statistischen Wert. Der Textinhalt einer Zelle kann durch zusätzliches Markup weiter strukturiert werden; möglich sind z.B. mehrzeilige Texte, also die Definition eines Zeilenumbruchs in einer Zelle, oder die Markierung eines Textes als eine Datumsangabe oder Seitennummer.

Eine aus dem Anwendungssystem [SPLV](#) herstammende Besonderheit ist die Möglichkeit, eine Seite in mehrere Bahnen zu unterteilen, die in TabML als „Slice“ bezeichnet werden; im Druck zwei- oder mehrseitige Tabellen können so durchgängig in einer Seite zusammengefaßt werden. Ein gleichermaßen von [SPLV](#) und [HTML 4.0](#) übernommenes Merkmal ist die Aufteilung der Tabellen in vertikale Kolonnen. Das Layoutformat hält Elemente bereit, welche die Kolonnen- und Bahnstruktur einer Tabelle beschreiben können.

3.3. Matrixformat

Das Matrixformat ist nicht Bestandteil von TabML 1.0. Die im folgenden beschriebene Struktur ist eine *mögliche* Form der Implementierung.



Die Tabellenwerte sind in den Zellen zweidimensionaler Matrizen gespeichert. Im Extremfall besteht eine Matrix aus nur einer Zelle. Eine Referenzmatrix verweist für jede Zelle mit einem als Deskriptor bezeichneten Element auf Beschreibungselemente, die in verschiedenen Gruppen hierarchisch angeordnet sind: Kopfspalten(K), Vorspalten(V), Aktionen(A) und weitere, den Matrizen übergeordnete Dimensionen(D). „(..)“ gibt eine Hierarchiestufe an. Die Hierarchiestufe einer Aktion bezieht sich auf die entsprechende Hierarchiestufe einer Tabellenstaffel. Weil in gestaffelten Tabellen alle übergeordneten Staffeln i.d.R. Summenstaffeln, also Aggregate der untergeordneten Staffeln sind, reicht es meist aus, nur die Aktionen der untersten Hierarchiestufe zu beschreiben. Die Matrizen müssen aber in jedem Fall eine Hierarchiestufenangabe oder einen Verweis auf den Deskriptor der entsprechenden Dimension enthalten.

Die eigentlichen Beschreibungselemente könnten optional (und redundanzfrei) außerhalb der Referenzmatrix gespeichert werden; die Referenzmatrix würde dann lediglich die Beziehungen zwischen den Beschreibungselementen definieren.

Über die Zelldescriptoren können bei Bedarf weitere, zellenbezogene Metadaten verfügbar gemacht werden. In Betracht kommen beispielsweise Regeln für die Geheimhaltung oder (Verweise auf) Fußnoten, Klassifikationen etc.

Ebenfalls in der Entwicklung des Matrixformates vorgesehen ist ein Mechanismus, der die Ausprägungen variabler Bestandteile (i.d.R. Drucktexte zu den Kopf- und Vorspalten oder Ausprägungen übergeordneter Dimensionen) in den Matrizen verfügbar macht.

Im Matrixformat sollen Beschreibungselemente und ihre Ausprägungen flexibel gespeichert werden können, d.h.grundsätzlich auch außerhalb eines Tabellendokuments. Ausprägungen (z.B. von Ordnungsfeldern) sollen wahlweise mit dem Beschreibungselement, in den Matrizen oder am Ende des Dokuments abgelegt werden können.

4. TabML Root

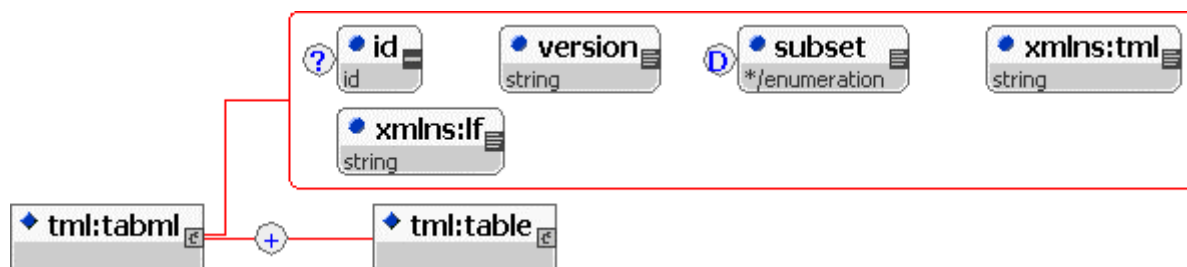
4.1. Beschreibung

Das Element `<tml:tabml>` ist das Wurzelement aller TabML-konformen Dokumente.

4.2. Elemente

4.2.1. Wurzelement `<tml:tabml>`

Element	<code><tml:tabml></code>		
Datentyp:	<i>element-content</i>		
Namespace:	<code>http://www.destatis.de/schema/tabml/1.0</code>	DTD:	<code>tml</code>
Inhaltsmodell:	<code>(tml:table+)</code>		



Attribute:	S	Datentyp, <u>Default</u> , Werte
<code>id</code>	?	<code>id</code> ;
<code>version</code>	M	<code>enumeration:1.0</code> ; Versionsbezeichnung
<code>subset</code>	?	<code>enumeration:none</code> ; Verwendetes Subset
<code>xmlns:tml</code>	M	<code>enumeration:</code> <code>http://www.destatis.de/schema/tabml/1.0</code>
<code>xmlns:lf</code>	M	<code>enumeration:</code> <code>http://www.destatis.de/schema/tabml-layout/1.0</code>

Details

Jedes Kindelement vom Typ `<tml:table>` enthält eine Tabelle im Layoutformat.

Das Attribut `version` gibt an, mit welcher TabML-Version ein Dokument erstellt wurde und dient Anwendungen daher vor allem zur Kompatibilitätsprüfung. Es dürfen nur freigegebene Versionsbezeichnungen verwendet werden. Die Verarbeitung eines TabML-Dokuments *muß* abgebrochen werden, wenn keine oder eine unzulässige Versionsangabe vorhanden ist.

Das Attribut `subset` ist erst in einer zukünftigen TabML-Version zur Verwendung vorgesehen. Es beschreibt eine definierte Untermenge einer TabML-Version und soll die Teilimplementierung von TabML erleichtern. Der Grundgedanke ist, daß schrei-

bende Anwendungen die tatsächlich verwendete Untermenge mit diesem Attribut bekanntmachen und lesenden Anwendungen, die nur eine Untermenge von TabML implementieren, eine einfache Kompatibilitätsprüfung ermöglichen.

Die Namensraumattribute `xmlns:*` sind erforderlich und müssen genau den angegebenen Wert haben.

5. TabML Tabelle

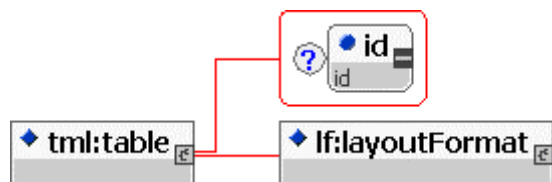
5.1. Beschreibung

Das Element `<tml:table>` enthält genau eine Tabelle im Layoutformat.

5.2. Elemente

5.2.1. Element `<tml:table>`

Element	<code><tml:table></code>	
Datentyp:	<i>element-content</i>	
Namespace:	http://www.destatis.de/schema/tabml/1.0	DTD: tml
Inhaltsmodell:	(lf:layoutFormat))	



Attribute: **S** **Datentyp, Default, Werte**

id ? id;

Details

`<tml:table>` enthält genau eine Tabelle im Layoutformat. In einer zukünftigen Version wird es zulässig sein, dieselbe Tabelle wahlweise bzw. zusätzlich im Matrixformat in diesem Element abzulegen.

6. TabML Layoutformat

6.1. Beschreibung

Das Layoutformat beschreibt statistische Präsentationstabellen. Es ermöglicht die generische Erzeugung anderer Präsentationsformate wie HTML, PDF, RTF usw.

6.2. Layoutinformation

Im Sinne des Layoutformates unterscheidet sich eine präsentationsreife Tabelle nicht nur in der Darstellung, sondern auch in der Struktur wesentlich von eher abstrakten, auf die Inhalte einer Tabelle hin orientierten Tabellenmodellen wie dem Matrixformat. Zur Layoutinformation zählen daher nicht nur Formatierungs-, sondern vor allem die Strukturinformationen, z.B. die Aufteilung der Tabelle in einzelne Seiten (oder Abschnitte, die als solche betrachtet werden können) und die Positionierung von Fußnoten.

6.3. Funktionen des Layoutformates

Das TabML-Layoutformat ermöglicht es, Tabellendaten zusammen mit Layoutinformationen in einem TabML-Dokument zu speichern.

- Das Layoutformat ist ein Ausgangsformat für die Konvertierung von Tabellen in unterschiedliche Präsentationsformate, wie z.B. HTML, RTF oder PDF.
- Es macht tabellenspezifische Layoutinformationen durch die Ablage im Dokument einer generischen Layoutgenerierung verfügbar. Die Layoutinformationen stammen direkt aus der Anwendung, die die Tabelle erzeugt, und/oder aus einer Nachbearbeitung. Die Notwendigkeit, für jedes spezielle Layout ein eigenes Stylesheet zu entwickeln, soll damit auf ein Minimum reduziert werden.
- Es bietet einen Container für das Ergebnis der Layoutgenerierung (generisch oder per Tabelleneditor) aus einer Tabelle im Matrixformat. Die Layoutgenerierung aus dem Matrixformat in beliebige Zielformate erfolgt auf diese Weise in zwei Schritten, wobei der erste Schritt stets die Erzeugung eines Dokumentes im Layoutformat ist, und für den zweiten Schritt die Werkzeuge verwendet werden können, die auf dem Layoutformat aufsetzen.
- Mit dem Layoutformat wird nicht das Ziel verfolgt, Tabellen in einer Art und Weise beschreiben zu können, die die Auswertung oder Manipulation der Tabellendaten, etwa für eine statistische Geheimhaltung, ermöglicht. Komplexe Funktionen dieser Art werden durch das Matrixformat unterstützt.

Da es konstruktive Überschneidungen zwischen Layoutformat und Matrixformat geben kann, sind die Elemente dieser Formate an unterschiedliche Namensräume gebunden (s.2.6)

6.4. Grundzüge des Layoutformats

Da Layoutformat eng an das Tabellenmodell von HTML 4.0 angelehnt, mit einer Reihe von Erweiterungen und Einschränkungen, die es an die entsprechenden speziellen Anforderungen anpassen. Analog zu HTML besteht eine Tabelle aus einer

Folge von Zeilen, die in größeren logischen Einheiten wie Tabellenkopf und -fuß zusammengefaßt und in Zellen unterteilt sind. Eine Zelle kann mehrere Zeilen und Spalten überspannen. Grundsätzlich sollten zusammenhängende Daten, z.B. ein Kopfspaltentext oder eine Zwischenüberschrift, in einer Zelle gespeichert werden.

Das Layoutformat speichert die meisten Daten in der Folge, in der sie üblicherweise in der Tabellenpräsentation erscheinen und in ein Präsentationsformat ausgegeben werden. Ausgenommen hiervon sind der Tabellenfuß (`<lf:foot>`), welcher in Analogie zu HTML direkt nach dem Tabellenkopf abgelegt wird, und die Fußnoten (siehe 6.4.2.8). Dies, und die Möglichkeit, für die Seitengenerierung zwei unterschiedliche Strukturknoten zu verwenden (siehe 6.4.2.6), erfordern u.U. die Zwischenspeicherung von Überschriften, Tabellenkopf- und fuß, sowie Fußnoten.

6.4.1. Layoutformat vs. HTML 4.0

Gegenüber dem in HTML 4.0 implementierten Tabellenmodell weist das Layoutformat einige Erweiterungen auf, die vor allem eine detailliertere und genauere Beschreibung der einzelnen Tabellenkomponenten erlauben. Einschränkungen bestehen dort, wo das HTML-Tabellenmodell in seiner Allgemeinheit über die Anforderungen an statistische Tabellen hinausgeht oder nachrangige Anforderungen in zukünftigen Versionen befriedigt werden können.

Wesentliche Erweiterungen vs. HTML:

- Eine Tabelle kann horizontal und vertikal feiner segmentiert werden, um beispielsweise Seiten- und Druckbahneinteilungen darzustellen.
- Überschriften, Kopfspalten, Tabellenfuß und Kolonneneinteilung können abschnittsweise variieren.
- Es steht eine größere Auswahl von Elementtypen für die Definition von Zellen zur Verfügung.

Wesentliche Einschränkungen vs. HTML:

- Geschachtelte Tabellen werden nicht unterstützt (aber Untertabellen).
- Graphische Elemente werden nicht explizit unterstützt.
- Formatierungsangaben sind auf die gebräuchlichsten Attribute beschränkt.

6.4.2. Strukturierung von Tabellen

6.4.2.1. Überblick

Im Folgenden werden die Strukturelemente des Layoutformates oberhalb der Zeilenebene beschrieben.

6.4.2.2. Elementklassen

Die Elementtypen des Layoutformates bilden eine Reihe von Elementklassen. *Metadaten* enthalten beispielsweise Strukturinformationen wie Kolonnenbeschreibungen (`<lf:columnStructure>`) oder Informationen für die Darstellung wie Stilattributdefinitionen (`<lf:styleDefinition>`). *Blockelemente* erlauben die horizontale Segmentierung einer Tabelle, d.h. die Darstellung von Tabellenzeilen und deren Gruppierung für die Abbildung logischer Strukturen wie Tabellenkopf und -körper. *Inline-Elemente* wie `<lf:h>` strukturieren eine Tabellenzeile durch ihre Unterteilung in Zellen verschiedenen Typs, die Kolonnen zugeordnet werden. *Datenelemente* wie `<lf:pTag>` erlauben es, den Inhalt einer Zelle genauer zu beschreiben.

6.4.2.3. Segmentierung

Das Element `<lf:frame>` definiert einen sogenannten Frame und dient zur Segmentierung von Tabellen. Jeder Frame verwendet eine Kolonnenbeschreibung und hat eine Überschrift (`<lf:title>`), einen Tabellenkopf (`<lf:head>`), einen Tabellenfuß (`<lf:foot>`) und einen oder mehrere Tabellenkörper (`<lf:body>`), wobei nur der Tabellenkörper vorhanden sein muß. Ein Frame eignet sich daher zur Beschreibung einer einzelnen oder einer Folge von Tabellenseiten, die auf Basis des Frames oder der nachgeordneten Tabellenkörper (`<lf:body>`) erzeugt werden können. Das Attribut `page-rendering` des Elements `<lf:layoutFormat>` dokumentiert die gewünschte Art der Seitengenerierung.

6.4.2.4. Horizontale Segmentierung, Bahnen

Kolonnenbeschreibungen (`<lf:columnStructure>`) erlauben die horizontale Segmentierung einer Tabelle und werden von `<lf:frame>` implizit oder explizit referenziert. Jeder `<lf:frame>` kann daher unterschiedlich segmentiert sein. In einer Kolonnenbeschreibung können Einzelkolonnen (`<lf:column>`) eine Kolonnengruppe (`<lf:columnGroup>`) bilden oder durch diese implizit definiert werden. Einzelkolonnen und Kolonnengruppen können wiederum eine sogenannte Bahn bilden (`<lf:slice>`). Eine Bahn ist ein Hilfsmittel, Kolonnen zu einer Einheit zusammenzufassen und einen Frame horizontal zu segmentieren. Ein häufiges Kriterium für die Bildung solcher Segmente ist der Wunsch, sie separat auszugeben, etwa wenn sich eine Tabelle horizontal über mehrere Papierseiten erstreckt. Eine Tabellenzelle muß vollständig innerhalb einer Bahn liegen, doch können Tabellenfelder aus mehreren Zellen bestehen, die in benachbarten Bahnen liegen.

6.4.2.5. Tabellenfelder über Bahngrenzen

Wenn ein Tabellenfeld eine Bahngrenze überschreitet, ist es notwendig, den Inhalt des Tabellenfeldes auf mehrere Zellen zu verteilen, da Zellen komplett innerhalb einer Bahn liegen müssen, und außerdem sichergestellt sein muß, an welchen Stellen der Inhalt des Tabellenfeldes durch Bahngrenzen geteilt wird. Umgekehrt soll es möglich sein, das Tabellenfeld inhaltlich und formal wiederherzustellen. Der Elementtyp `<lf:fr>` übernimmt hier die Funktion, Zellen semantisch zu klammern, die aus der Aufteilung eines Tabellenfeldes an Bahngrenzen entstanden und daher Fragmente eines Tabellenfeldes sind. Er kann alle Zellelementtypen enthalten, aber es müssen mindestens zwei Zellen vorhanden sein, und alle müssen denselben Typ haben (da sie aus einem Tabellenfeld entstanden sind).

Bei der Zerlegung eines Tabellenfeldes sollte wie folgt vorgegangen werden:

- Der Inhalt des Tabellenfeldes muß, ggf. nach rechnerischer Ausrichtung, an den Bahngrenzen zerlegt und auf soviel Zellen verteilt werden, wie Bahnen betroffen sind.
- Alle Zellen haben denselben Elementtyp. Sie unterscheiden sich außer durch den Inhalt nur durch die Ausrichtung und die Anzahl der Kolonnen, die sie belegen, vom ursprünglichen Tabellenfeld.
- Die Ausrichtung der Zellen wird nach folgendem Schema manipuliert:

Alte Ausrichtung	Linke Zelle	Mittlere Zelle(n)	Rechte Zelle
left	left	left	left
center	right	center	left
right	right	right	right

- Der ursprüngliche Inhalt des Tabellenfeldes wird durch Verketteten der auszugebenden Zellen rekonstruiert.
- Die ursprüngliche Ausrichtung für das *vollständige* Tabellenfeld wird aus den Ausrichtungen der äußeren Zellen abgeleitet; andere Werte sind unzulässig:

Linke Zelle	Rechte Zelle	Gesamtausrichtung
left	left	left
right	left	center
right	right	right

- Die Rekonstruktionsregel für die Ausrichtung kann auch so formuliert werden: Wenn die äußeren Zellen eine verschiedene Ausrichtung haben, ist die Resultatausrichtung `center`, ansonsten die gegebene.

6.4.2.6. Seitengenerierung und –nummerierung

Sowohl zur Ablage einer vorhandenen Seitennummerierung oder –kennung als auch zu deren Generierung stehen die Elemente `<lf:pTag>` und `<lf:pNum>` zur Verfü-

gung. `<lf:pTag>` kann eine beliebige Seitenkennung enthalten, mit einem optionalen Kindknoten von Typ `<lf:pNum>`, der die eigentliche Seitennummer enthält.

Die Ausgabe einer *vorhandenen* Seitenkennung soll benutzeroptional unterdrückt bzw. erzwungen werden können. Anwendungen, die ein seitenbasiertes Präsentationsformat erzeugen, sollen sowohl die Steuerung der Seitengenerierung unabhängig von der dokumentinternen Einstellung als auch die Steuerung der Generierung bzw. Unterdrückung einer Seitenkennung erlauben.

Die Unterdrückung einer vorhandenen Seitenkennung bzw. die Erzeugung einer neuen mit TabML-konformen Anwendungen ist nur bei Vorhandensein des Elementes `<lf:pTag>` (und ggf. `<lf:pNum>`) möglich. Ohne dieses Element kann bzw. muß die Generierung einer Seitenkennung mit den Mitteln anderer Anwendungen erfolgen.

Die Generierung einer Seitenkennung kann erforderlich sein, weil sie vom Anwender verlangt wird oder Seiten auf Basis von `<lf:body>` generiert werden; in diesem Fall werden Überschrift (`<lf:title>`), Tabellenkopf (`<lf:head>`) und Tabellenfuß (`<lf:foot>`) aus dem `<lf:frame>` in jede erzeugte Seite übernommen (s. 6.4.2.7). Jede in diesen Komponenten mit dem Element `<lf:pTag>` markierte Seitenkennung muß mit dem entsprechenden Inhalt versorgt werden. Die Generierung der Seitenkennung besteht im wesentlichen aus dem Errechnen einer neuen Seitennummer und deren Ausgabe – ggf. anstelle einer vorhandenen Seitennummer oder Seitenkennung – und ist abhängig vom Vorhandensein des Kindknotens `<lf:pNum>`:

- Ist der Kindknoten `<lf:pNum>` vorhanden, wird dessen Inhalt ignoriert und an seiner Stelle die errechnete neue Seitennummer ausgegeben; der übrige Inhalt des Elements `<lf:pTag>` bleibt unverändert.
- Fehlt der Kindknoten `<lf:pNum>`, wird die errechnete Seitennummer benutzeroptional
 - anstelle des Inhaltes von `<lf:pTag>` ausgegeben (Default) oder
 - dem Inhalt mit vorausgehendem Leerzeichen angehängt oder
 - dem Inhalt mit nachfolgendem Leerzeichen vorangestellt.

Startzahl und Schrittweite der errechneten Seitennummer sollen vom Anwender beeinflußt werden können; der jeweilige Defaultwert ist '1'.

6.4.2.7. Überschrift, Tabellenkopf und Tabellenfuß

Überschrift, Tabellenkopf und Tabellenfuß sind stets an einen `<lf:frame>` gebunden. Eine Tabelle mit durchgängig gleichen Überschriften, Tabellenköpfen und -füßen kann jede Seite in einem `<lf:body>` abbilden und benötigt so nur einen `<lf:frame>`. Überschrift, Tabellenkopf und -fuß werden für jede generierte Seite aus dem `<lf:frame>` entnommen. Eine weitere Voraussetzung ist, daß die Kolonneneinteilung für die ganze Tabelle einheitlich ist (s. 6.4.2.9). Tabellen mit wechseln-

den Überschriften usw. oder Kolonneneinteilungen können für jede Tabellenseite einen eigenen `<lf:frame>` verwenden. Auch eine Kombination ist möglich, etwa wenn eine Überschrift erst nach mehreren Seiten wechselt.

6.4.2.8. Fußnoten

Fußnoten werden in Tabellenzellen in Form einer sogenannten Fußnotenreferenz abgelegt, also dort, wo Bezug auf eine Fußnote genommen wird. Eine Fußnotenreferenz (`<lf:fRef>`) enthält eine Fußnotenkennung (`<lf:fId>`) und einen optionalen Fußnotentext (`<lf:fText>`). Aus der Fußnotenkennung kann ein (textueller) Verweis auf den Fußnotentext generiert werden. Fehlt der Fußnotentext, bezieht sich die Fußnotenkennung auf den letzten vorhergehenden Fußnotentext, der mit der gleichen Fußnotenkennung abgelegt wurde. Ist ein Fußnotentext vorhanden, ersetzt dieser einen mit der gleichen Fußnotenkennung abgelegten vorhandenen Fußnotentext.

Es ist der Anwendung überlassen, an welcher Stelle der Tabelle(nseite) sie den Fußnotentext ausgibt – beispielsweise am Ende des Tabellenkörpers oder im Tabellenfuß –, und sie kann die Steuerung dem Anwender überlassen.

6.4.2.9. Definition von Untertabellen

`<lf:axis>` gestattet es, innerhalb eines `<lf:body>` eine beliebige Anzahl von Untertabellen zu definieren. Es ist ein rekursives Element, das die hierarchische Anordnung von Kontextinformationen über die Tabellenwerte mit `<lf:subtitle>` erlaubt. I.d.R. wird es sich hierbei um eine Zwischenüberschrift handeln. Die Dimensionalität einer Tabelle wird hier also nicht (nur) durch (eine Folge von) Zwischenüberschriften, sondern durch die hierarchische Struktur der Elemente wiedergegeben. Kontextinformationen und Tabellenwerte können aber auch sequentiell abgelegt werden.

6.4.2.10. Werte in Tabellen

`<lf:data>` enthält Tabellenwerte und die zu ihnen gehörenden Vorspalten und ist entweder einer Untertabelle oder direkt `<lf:body>` zugeordnet, wenn keine Untertabellen unterscheiden werden müssen. Zellen, die Werte enthalten sollen vom Typ `<lf:v>` sein.

6.4.2.11. Repräsentation von Sonderzeichen

Der Elementtyp `<lf:sc>` gestattet es, ausgewählte Zeichen im Inhalt einer Zelle durch ihre Beschreibung zu ersetzen und damit häufig auftretende Probleme bei der Darstellung, Konvertierung usw. solcher Zeichen zu vermeiden.

Z.Z. Ist dies für folgende Zeichen möglich:

- Euro-Symbol.

Das Zeichen wird durch seinen Namen, der dem in HTML verwendeten Zeichennamen (character entities) entspricht, ohne deren Sonderzeichen „&“ und „;“, die dort zur Erkennung und Begrenzung der Zeichennamen dienen. Beispiele

```
<lf:sc character-name="euro"/>
```

Wie bei `<lf:bc>` gibt das optionale Attribut `repeat-content` an, wie oft ein Zeichen erzeugt wird.

6.4.2.12. Verwendung von Kolonnenbeschreibungen

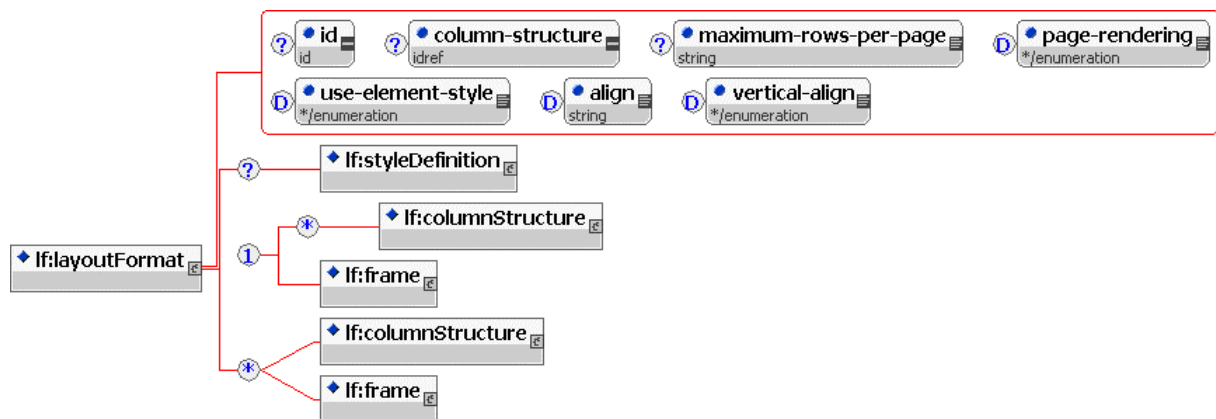
Das Layoutformat läßt beliebig viele Kolonnenbeschreibungen zu, die mittels ihres `name`-Attributes unterscheiden werden. Eine Kolonnenbeschreibung bezieht sich immer auf einen oder mehrere `<lf:frame>`, d.h., umgekehrt beruhen alle Zeilen eines `<lf:frame>` immer auf der gleichen Kolonnenstruktur.

Es darf nur eine Kolonnenbeschreibung *ohne* `name`-Attribut geben. Jeder `<lf:frame>` *ohne* expliziten Verweis auf eine Kolonnenbeschreibung mittels des Attributes `column-structure` referenziert implizit diese „unbenannte“ Kolonnenbeschreibung. Eine Anwendung muß eine Warnung ausgeben, wenn in einem TabML-Dokument für eine Tabelle mehr als eine Kolonnenbeschreibung ohne `name`-Attribut definiert wurde, und darf nur die erste dieser Kolonnenbeschreibungen verwenden.

6.5. Layoutformat: Root

6.5.1. Element <lf:layoutFormat>

Element	<lf:layoutFormat>	
Datentyp:	<i>element-content</i>	
Namespace:	http://www.destatis.de/schema/tabml-layout/1.0	DTD: lf
Inhaltsmodell:	(lf:styleDefinition?, (lf:columnStructure*, lf:frame), (lf:columnStructure lf:frame)*)	



Attribute:

id

maximum-rows-per-page

page-rendering

use-element-style

column-structure

Ausrichtung

S Datentyp, Default, Werte

? *id*;D *int_0..n*; Maximale Anzahl der pro Seite zu erwartenden Zeilen; kein Default.D *enumeration*; Seitengenerierung steuern"default" keine Vorgabe"frame" <frame> als Seite generieren."body" <body> als Seite generieren"none" Keine SeitengenerierungD *enumeration*; yes|no; siehe 6.11.5D *idref*; Verweist auf eine benannte oder (falls vorhanden) unbenannte Kolonnenbeschreibung:" leer: unbenannte Kolonnenbeschreibung*idref* Verweis auf benannte Kolonnenbeschreibung

Ausrichtung von Daten

6.6. Kolonnenbeschreibung

Eine Kolonnenbeschreibung für eine Tabelle besteht aus bis zu vier Ebenen. Die erste, oberste bildet `<lf:columnStructure>` als erforderliches Wurzelement der Kolonnenbeschreibung. Die zweite, optionale Ebene besteht aus einer beliebigen Anzahl von `<lf:slice>`-Elementen, die Einzelkolonnen und Kolonnengruppen zu Bahnen zusammenfassen. In der dritten, wiederum erforderlichen Ebene existiert mindestens ein Element `<lf:columnStructure>` oder `<lf:column>`; diese Elemente definieren Kolonnen. Die vierte, optionale Ebene besteht aus `<lf:column>`-Elementen, die Kolonnengruppen der dritten Ebene nachgeordnet sind, und von diesen Attribute erben können.

Kolonnengruppen sind für die reine Definition der Kolonnen nicht notwendig, da auch ein `<lf:column>`-Element beliebig viele Kolonnen beschreiben kann. Ihr Zweck ist, Kolonnen z.B. aus logischen Gründen zusammenfassen zu können (etwa alle Kolonnen der Vorspalte) und/oder Attributwerte für nachgeordnete Kolonnen bereitzustellen.

Die Kolonnenbeschreibung erlaubt die Angabe eines Kolonnentyps, der durchgängig ist, d.h. er kann nicht für einzelne Abschnitte, Zeilen oder Zellen überschrieben werden. Die Unterscheidung von Vorspalten- (`header`) und Wertspalten (`data`) ist allerdings nur in den Datenblöcken (`<lf:data>`) eines Tabellenkörpers (`<lf:body>`) relevant.

6.6.1. Tabellen-, Bahn- und Kolonnenbreite

In einer Kolonnenbeschreibung kann die Breite der Tabelle, jeder Bahn und jeder Kolonne explizit angegeben werden. Für die Darstellung der Tabelle ist nur die Kolonnenbreite relevant. Die Breite der Tabelle und der Bahnen wird sinnvollerweise nur benötigt, um fehlende Bahn- bzw. Kolonnenbreiten zu berechnen (s. 6.6.1.2).

Besteht ein Widerspruch zwischen den Werten, erhält die Summe der Kolonnenbreiten den Vorrang vor der Bahnbreite, und diese vor der Breite der Tabelle, und zwar gleichgültig, ob diese Werte explizit oder errechnet sind. Anwendungen können eine Warnung ausgeben, wenn sie einen Konflikt zwischen den Werten feststellen, müssen ihn aber auflösen, indem sie die Werte entsprechend ihrer Priorität aktualisieren. Es wird daher empfohlen, möglichst für jede Kolonne eine Breite > 0 anzugeben und für die Bahnen und die Tabelle auf die Angabe der Breite zu verzichten, oder den Defaultwert "0" zu verwenden.

6.6.1.1. Relative Breitenangaben

Das Layoutformat verwendet in den Kolonnenbeschreibungen für die Spezifikation der Tabellen-, Bahnen- und Kolonnenbreiten ganzzahlige Werte ohne Maßeinheit. Die Methode der Wertermittlung und der Wertebereich sind anwendungsabhängig. Für eine Tabelle mit festem Font könnte eine Anwendung z.B. die Anzahl der Zeichen verwenden, die Kolonnen, Bahnen und Tabelle belegen (die u.a. Beispiele beruhen auf einer 132 oder 264 Byte breiten Tabelle).

Da die Maßeinheit fehlt, ermöglichen die Werte nur eine Aussage über den relativen Anteil einer Bahn oder Kolonne an der Gesamtbreite der Tabelle. Dieses Vorgehen entkoppelt die Tabellenbeschreibung von Zielformat. Eine Anwendung, die ein Präsentationsformat auf Basis einer bestimmten Maßeinheit erzeugt (z.B. RTF auf Basis von „twips“), muß die relativen Werte umrechnen; hierfür benötigt sie die für die Präsentation zur Verfügung stehende Gesamtbreite in der verwendeten Maßeinheit und die relative Gesamtbreite der Tabelle bzw. der ausgewählten Bahn(en).

Die Tabelle des ersten Beispiels mit der relativen Gesamtbreite von 132 soll für die Verarbeitung mit WORD nach RTF konvertiert werden, die Gesamtbreite beträgt 11906 „twips“, abzüglich 359 für den linken und 431 für den rechten Rand, also 11116 nutzbare Breite. Es ergibt sich ein Umrechnungsfaktor von $11116 / 132 = 84,21$. Für eine 8 Zeichen breite Kolonne errechnet sich ungerundet eine Breite in „twips“ von $8 * 84,21 = 673,68$.

Die zweibahnige Tabelle des dritten Beispiels ist mit einer relativen Gesamtbreite von 264 genau doppelt so breit, und der Umrechnungsfaktor halbiert sich (eine zusätzliche Abweichung durch Rundung ist möglich). Bei Auswahl einer Bahn mit der relativen Breite von 132 ergibt sich wiederum der gleiche Wert wie für das erste Beispiel.

Wie genau der Umrechnungsfaktor berechnet wird und welche zusätzlichen Parameter in die Berechnung eingehen ist Sache der Anwendung. Sie kann beispielsweise Anwendern die Möglichkeit geben, den errechneten Umrechnungsfaktor zusätzlich zu beeinflussen oder vorzugeben, oder die Breite der Bahnen normalisieren, wenn nicht auf jeder Seite gleich viele Bahnen erscheinen.

6.6.1.2. Breiten ermitteln und berechnen

Die Ermittlung und ggf. Berechnung der Tabellen-, Bahn- und Kolonnenbreiten wird nachfolgend beispielhaft beschrieben. Ziel dieses Verfahrens ist es, das Durchlesen der kompletten Tabelle für die Ermittlung dieser Werte zu vermeiden. Anwendungen sind frei, andere, auch mehrere Algorithmen zu implementieren und nach eigenen Regeln auszuführen und dem Anwender die Auswahl des Algorithmus zu ermöglichen.

- A Ermitteln der Gesamtbreite der Tabelle, also des Wertes des Attributes `total-width` in `<lf:columnStructure>` oder ein optional von der Anwendung oder dem Anwender bereitgestellter Defaultwert. Der erste in dieser Folge ermittelte Wert > 0 wird übernommen, aber ein Ergebnis $=0$ ist ebenfalls zulässig.
- B Berechnen der Summe und des Durchschnitts aller Bahnbreiten (in Vorbereitung auf Schritt C.1) anhand des Attributes `total-width` in `<lf:slice>`. Falls keine expliziten Bahnen definiert sind, wird der Wert für die Breite der Tabelle übernommen, da Tabelle und Bahn identisch sind.
- C Für jede ausgewählte Bahn:
 - C.1 Ermitteln der Bahnbreite; dies ist der Wert des Attributes `total-width` in `<lf:slice>` oder ein optional von der Anwendung oder dem Anwender bereitgestellter Defaultwert oder der in Schritt B berechnete Durchschnitt der Bahn-

- breiten. Der erste in dieser Folge ermittelte Wert > 0 wird übernommen, aber ein Ergebnis $=0$ ist ebenfalls zulässig.
- C.2 Generieren aller Einzelkolonnen, also Auswertung der `column-span`-Attribute und Verteilung der expliziten Attribute auf die Einzelkolonnen
 - C.3 Berechnen der Anzahl der Kolonnen mit fehlender Kolonnenbreite und der Anzahl der Kolonnen mit bekannter Kolonnenbreite und der Summe der Kolonnenbreiten, also des Attributes `column-width` in `<lf:columnGroup>` und `<lf:column>`. Ist die Anzahl der Kolonnen mit fehlender Kolonnenbreite $= 0$, wird mit Schritt C.5 fortgefahren, ist sie > 0 , werden die fehlenden Kolonnenbreiten in Schritt C.4 berechnet
 - C.4 Fehlende Kolonnenbreiten können auf verschiedene Weise – ggf. benutzeroptional – eingesetzt oder ermittelt werden. Die Schritte C.4.1 und C.4.2 können bereits zusammen mit dem Schritt C.2 ausgeführt werden, so daß während der Ausführung von Schritt C.3 alle Kolonnenbreiten bekannt sind.
 - C.4.1 Die Anwendung oder der Anwender gibt einen Defaultwert > 0 vor.
 - C.4.2 Der Wert der nächsten vorhergehenden oder – falls nicht vorhanden – nachfolgenden Kolonne der Bahn mit einer Breite > 0 wird übernommen.
 - C.4.3 Die Summe der Kolonnenbreiten aus Schritt C.3 wird von der in Schritt C.1 ermittelten Bahnbreite abgezogen. Ist das Ergebnis mindestens so groß wie die Anzahl der Kolonnen mit unbekannter Breite, wird es durch diese geteilt, um die Kolonnenbreite zu erhalten, oder es wird sofort geteilt, und das Ergebnis muß ≥ 1 sein.
 - C.4.4 Es wird der Durchschnittswert der bekannten Kolonnenbreiten der Bahn berechnet und als Kolonnenbreite eingesetzt.
 - C.5 Sind alle Kolonnenbreiten bekannt, wird deren Summe als (neuer) Wert der Bahnbreite eingesetzt und die nächste Bahn verarbeitet.
 - D Alternativ besteht die Möglichkeit, alle ausgewählten Bahnen logisch zusammenzufassen und die Maßnahmen nach Schritt C auf Basis aller betroffenen Kolonnen auszuführen. Dies hat aber keine Auswirkungen, wenn fehlende Kolonnenbreiten nach C.4.1 ersetzt werden.
 - E Sind alle Bahnbreiten bekannt, wird deren Summe als (neuer) Wert der Tabellenbreite eingesetzt.

Die Aktualisierung der Bahn- und Tabellenbreiten in den Schritten C.5 und E ist nicht unbedingt erforderlich, da zu diesem Zeitpunkt alle Kolonnenbreiten bekannt sind. Siehe auch 6.6.1.

Das Durchlesen einer Tabelle zur Ermittlung dieser Werte ist immer notwendig, wenn keine Kolonnenbeschreibung vorhanden ist oder fehlende Kolonnenbreiten nicht ermittelt werden können.

6.6.1.3. Die Anzahl der Kolonnen berechnen

Die Anzahl der Kolonnen erfolgt durch die Auswertung der `column-span`-Attribute der Kolonnen (`<lf:column>`) und Kolonnengruppen (`<lf:columnGroup>`). Das `column-span`-Attribut einer Kolonnengruppe darf nicht berücksichtigt werden, wenn die Kolonnengruppe Kindknoten vom Typ `<lf:column>` hat. In diesem Fall muß sich die Auswertung des Attributs `column-span` auf die Kindknoten beschränken.

Fehlt die Kolonnenbeschreibung, muß die Tabelle komplett gelesen werden, um die Anzahl der Kolonnen (und ihre Breite) zu ermitteln. Für jede Zeile wird dabei die Anzahl der Kolonnen gebildet aus der Summe der `column-span`-Attribute der Zellen und sich ergebender Differenzen aus der Verwendung des Attributs `column-start` sowie unter Berücksichtigung von Zellen vorhergehender Tabellenzeilen, die sich in die aktuelle Zeile erstrecken (Attribut `row-span`).

Es ist ein Fehler, wenn eine Zeile mehr Kolonnen belegt, als die Kolonnenbeschreibung zuläßt (dieser Fehler kann nur in der letzten oder einzigen Bahn auftreten). Eine Zeile darf weniger als die erlaubte Anzahl von Kolonnen belegen, und es ist auch erlaubt, daß eine Zeile überhaupt keine Kolonne belegt, also leer ist. Da sich eine Zeile immer über alle Bahnen erstreckt, ist es möglich, daß sie in einer oder mehreren Bahnen leer ist, in vorausgehenden Bahnen aber einen Inhalt hat.

Es ist ebenfalls ein Fehler, wenn sich Zellen überlappen. Lücken hingegen sind erlaubt und müssen ggf. durch das Einfügen oder Simulieren einer leeren Zelle ausgefüllt werden.

6.6.1.4. Beispiele:

- Eine Tabelle mit einer impliziten Bahn, mit stets gleich breiten Kolonnen innerhalb einer Kolonnengruppe (nur Ebene 1 und 3), mit und ohne kolonnengruppen:

```
<lf:columnStructure total-width="132">
  <lf:columnGroup id="cg1"
    column-span="3"
    column-type="header"
    column-width="10"
  />
  <lf:columnGroup id="cg2"
    column-span="8"
    column-type="data"
    column-width="12"
  />
</lf:columnStructure>

<lf:columnStructure total-width="132">
  <lf:column column-span="3"
    column-type="header"
    column-width="10"
  />
  <lf:column column-span="8"
    column-type="data"
    column-width="12"
  />
</lf:columnStructure>
```

Eine zweibahnige, spiegelbildlich aufgebaute Tabelle; die Kolonnen der Vorspalten (insgesamt drei pro Bahn) sind zu einer Kolonnengruppe zusammengefaßt, aber unterschiedlich breit, weshalb nachgeordnete [<lf:column>](#)-Elemente erforderlich sind

(Ebene 4); die beiden ersten Kolonnen übernehmen alle Attribute der Kolonnengruppe, die dritte Kolonne überschreibt für sich die Kolonnenbreite mit dem Attribut column-width.

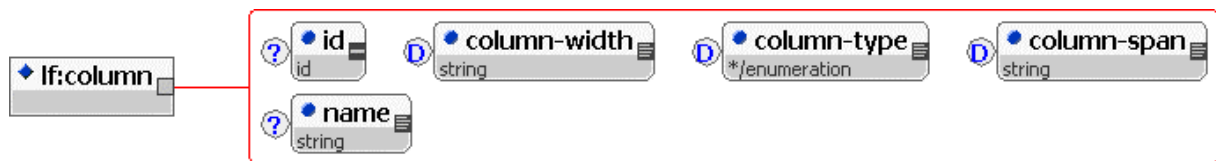
```
<lf:columnStructure total-width="264">
  <lf:slice total-width="132">
    <lf:columnGroup id="cg1"
      column-type="header"
      column-width="10"
    >
      <lf:column column-span="2"/>
      <lf:column column-width="16"/>
    </lf:columnGroup>
    <lf:columnGroup id="cg2" column-span="8"
      column-type="data"
      column-width="12"
    />
  </lf:slice>
  <lf:slice total-width="132">
    <lf:columnGroup id="cg3" column-span="8"
      column-type="data"
      column-width="12"
    />
    <lf:columnGroup id="cg4"
      column-type="header"
      column-width="10"
    >
      <lf:column column-span="2"/>
      <lf:column column-width="16"/>
    </lf:columnGroup>
  </lf:slice>
</lf:columnStructure>
```

6.6.2. Tabelle der Elementtypen

Element	Beschreibung
<lf:column>	Definiert Attribute einer Kolonne.
<lf:columnGroup>	Faßt 1-n Kolonnen zu einer Kolonnengruppe zusammen und definiert optionale Attribute für diese Kolonnen.
<lf:columnStructure>	Enthält eine Kolonnenbeschreibung
<lf:slice>	Definiert eine Bahn in der Kolonneneinteilung.

6.6.3. Element <lf:column>

Element	<lf:column>		
Datentyp:	<i>empty-element</i>		
Namespace:	http://www.destatis.de/schema/tabml-layout/1.0	DTD:	lf
Inhaltsmodell:	EMPTY		



Attribute:	S	Datentyp, <u>Default</u> , Werte
id	?	<i>id</i> ;
name	?	<i>char</i> ;
column-width	D	<i>int</i> ; Default:= "0"; Breite der Kolonne(n)
column-type	D	<i>enumeration</i> : <u>undefined</u> header separator data numbering other
column-span	D	<i>int</i> ; Default:= "1"; Anzahl der Kolonnen

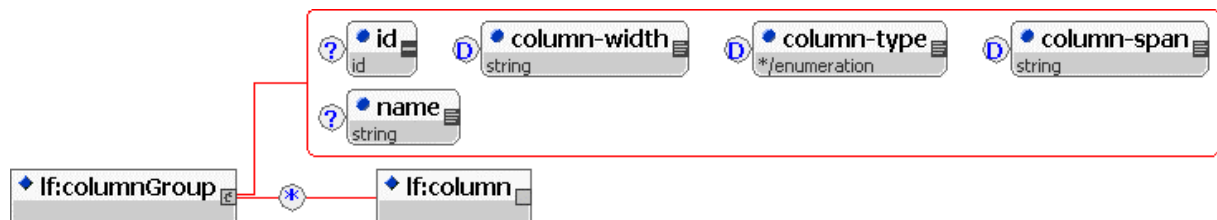
Details

<lf:column> beschreibt eine beliebige Anzahl Kolonnen, die identische Attribute haben. Kolonnenattribute können von übergeordneten <lf:columnGroup>-Elementen geerbt werden.

Das Attribut `column-span` gibt die Anzahl von Kolonnen an, die mit den angegebenen Attributen erzeugt werden. Im Unterschied zu <lf:columnGroup> werden die generierten Kolonnen nicht als zusammengehörend betrachtet.

6.6.4. Element `<lf:columnGroup>`

Element	<code><lf:columnGroup></code>		
Datentyp:	<i>element-content</i>		
Namespace:	<code>http://www.destatis.de/schema/tabml-layout/1.0</code>	DTD:	<code>lf</code>
Inhaltsmodell:	<code>(lf:column*)</code>		



Attribute:	S	Datentyp, <u>Default</u> , Werte
<code>id</code>	?	<code>id</code> ;
<code>column-width</code>	D	<code>int</code> ; Default:= "0"; Breite der Kolonne(n)
<code>column-type</code>	D	<i>enumeration</i> : <code>undefined</code> <code>header</code> <code>separator</code> <code>data</code> <code>numbering</code> <code>other</code>
<code>column-span</code>	D	<code>int</code> ; Default:= "1"; Anzahl der Kolonnen

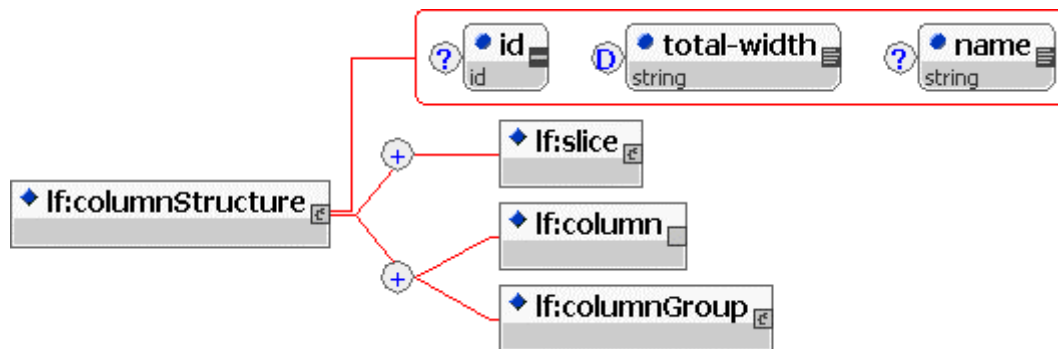
Details

`<lf:columnGroup>` faßt die nachgeordneten Kolonnen zu einer Kolonnengruppe zusammen und stellt ihnen Attributwerte zur Verfügung. Die Kolonnen werden entweder durch das Attribut `column-span` oder durch Elemente vom Typ `<lf:columnGroup>` erzeugt.

Das Attribut `column-span` gibt die Anzahl von Kolonnen an, die mit den angegebenen Attributen erzeugt werden. Das Attribut *muß* ignoriert werden, wenn mindestens ein `<lf:column>`-Kindknoten vorhanden ist.

6.6.5. Element `<lf:columnStructure>`

Element	<code><lf:columnStructure></code>		
Datentyp:	<i>element-content</i>		
Namespace:	<code>http://www.destatis.de/schema/tabml-layout/1.0</code>	DTD:	<code>lf</code>
Inhaltsmodell:	<code>(lf:slice+ (lf:column lf:columnGroup)+)</code>		



Attribute:	S	Datentyp, <u>Default</u> , Werte
<code>id</code>	?	<code>id</code> ;
<code>name</code>	?	<code>char</code> ;
<code>total-width</code>	D	<code>int</code> ; Default:= "0"; Breite der Tabelle

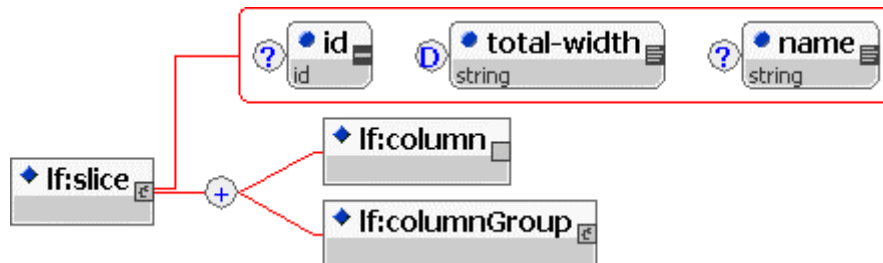
Details

`<lf:columnStructure>` enthält eine vollständige Kolonnen- und Bahnbeschreibung. Sie besteht aus einer Folge von 1-n `<lf:slice>`-Elementen mit 1-n nachgeordneten `<lf:column>` und/oder `<lf:columnGroup>`-Elementen oder einer direkten Folge von 1-n Elementen vom Typ `<lf:column>` und/oder `<lf:columnGroup>`, die implizit eine Bahn bilden.

Zur Verwendung des Attributes `name` s. 6.4.2.12.

6.6.6. Element <lf:slice>

Element	<lf:slice>		
Datentyp:	<i>element-content</i>		
Namespace:	http://www.destatis.de/schema/tabml-layout/1.0	DTD:	lf
Inhaltsmodell:	(lf:column lf:columnGroup) +		



Attribute: S Datentyp, Default, Werte

id	? <i>id</i> ;
name	? <i>char</i> ;
total-width	D <i>int</i> ; Default:= "0"; Breite der Druckbahn

Details

<slice> faßt eine Anzahl Kolonnen bzw. Kolonnengruppen zu einer Bahn zusammen.

6.7. Stilattribute

Die in TabML definierten Stilattribute sind eine Untermenge aus CSS1/CSS2, sowohl bezüglich der Attribute als auch der Werte einzelner Attribute. Bei der Erzeugung eines Dateiformates, das die Verwendung von CSS1/CSS2-Stilattributen erlaubt, können die Werte daher direkt übernommen werden. Es ist aber auch vorstellbar, die Stildefinitionen eines TabML-Dokuments (d.h. die der Dokument- und der Elementtyp-Ebene) zur Generierung von (externen) Stylesheets zu verwenden.

6.7.1. Warum Stilattribute?

Die Trennung von Struktur und Layout wird allgemein als wünschenswert und als Vorzug angesehen. Damit verbunden ist der Gedanke, daß das endgültige Layout bestimmt wird durch das gewählte Präsentationsformat und die für die Präsentation verwendete Soft- und Hardware. Einerseits kann dies dazu führen, daß entweder nur noch Standard-Layouts verwendet werden, oder jede gewünschte Abweichung bei jeder Transformation in ein anderes Format explizit definiert werden muß, ggf. durch das Erstellen von Stylesheets. Andererseits gibt es keine verlässliche und ausreichend große Schnittmenge von Stilattributen, die überall unterstützt werden.

TabML versucht hier, einen Mittelweg zu gehen, indem das Layoutformat zwar die Angabe von Stilattributen erlaubt, um eine durchgängige Präsentation zu erzielen, aber da die Beschränkungen und Möglichkeiten der verwendeten Formate und der Soft- und Hardware hierdurch nicht aufgehoben werden können, haben diese Stilattribute eher die Funktion einer „Absichtserklärung“. Mit anderen Worten, Stilattribute definieren ein gewünschtes Layout, stehen aber unter dem Vorbehalt, mit den gewählten Präsentationsmitteln nicht dargestellt werden zu können. Ein Anwender kann daher nicht erwarten, daß von ihm verwendete Stilattribute immer die gewünschte Formatierung bewirken.

6.7.2. Verarbeitung von Stilattributen

Stilattribute können im Layoutformat auf unterschiedlichen Ebenen definiert und vererbt werden. Die volle Unterstützung des Vererbungsmechanismus erfordert die Verwendung entsprechend leistungsfähiger Software, und es wird davon ausgegangen, daß dies z.Z. mit erträglichem Aufwand nur möglich ist, wenn hierfür auf ausreichend mächtigen Programmiersprachen basierende Anwendungen entwickelt werden. Insbesondere Stylesheets (z.B. CSS oder XSL) werden hier rasch an Grenzen stoßen.

Die Unterstützung von Stilattributen ist aus diesen Gründen kein Kriterium für die TabML-Konformität von Anwendungen, denen es daher überlassen ist, inwieweit sie Stilattribute unterstützen. Z.B. kann eine Anwendung lediglich Stilattribute der Elementinstanzebene (s.u.) unterstützen. Jede Anwendung muß aber sämtliche Elemente und Attribute akzeptieren, und der Grad der Unterstützung sollte in den Unterlagen dokumentiert sein.

6.7.3. Definition und Vererbung

Stilattribute existieren innerhalb des Layoutformats auf drei logischen, hierarchisch angeordneten Ebenen. Die erste, sogenannte *Dokument-Ebene* stellt Defaultwerte für die beiden anderen, ihr nachgeordneten Ebenen zur Verfügung, und wird durch Elemente vom Typ `<lf:style.defaults>` mit Werten versorgt. Die *Elementtyp-Ebene* definiert elementtypbezogene Attributwerte, z.B. für Überschriften, Fußnoten und Tabellenwerte, die aus den übrigen `<lf:style.elementtype>` Elementen stammen. Das Attribut `use-element-style` eines Elementes bestimmt, ob fehlende Attribute aus der Elementtyp-Ebene ersetzt (`yes`, was zugleich der Default ist) oder vom Elternknoten übernommen werden (`no`), und wird selbst vom Knoten `<lf:layoutFormat>` an alle Nachfahren vererbt. Die explizit in den Elementen spezifizierten Stilattribute bilden die dritte, die *Elementinstanz-Ebene*. Die Anwendung, die Attributwerte als Default bereitstellen oder explizit setzen, löschen und überschreiben kann, bildet eine vierte, hier nicht behandelte Ebene außerhalb eines TabML-Dokuments.

Die Dokument-Ebene wird durch den Elementtyp `<lf:style.defaults>` in `<lf:styleDocumentDefaults>` erzeugt. Die Elementtyp-Ebene wird durch den Elementtyp `<lf:style.elementtype>` in `<lf:styleElementDefaults>` erzeugt. Im Einzelnen:

- **Attribute der Elementinstanz-Ebene**
Auf der Elementinstanz-Ebene sind Stilattribute als Attribute eines Elementes angegeben. Ein nicht explizit angegebenes Attribut wird auf der Elementtyp-Ebene gesucht, wenn für den aktuellen Knoten `use-element-style="yes"` gilt. Ist es in der Elementtyp-Ebene nicht definiert, wird das Attribut vom Elternknoten des aktuellen Knotens übernommen.
- **Attribute der Elementtyp-Ebene**
Auf der Elementtyp-Ebene werden Stilattribute für bestimmte Elementtypen definiert. Nichtdefinierte Attribute (bzw. Attribute mit dem Wert `"default"`) erhalten ihre Werte aus der Dokument-Ebene. Attribute der Elementtyp-Ebene werden nur verwendet, wenn für den aktuellen Knoten `use-element-style="yes"` gilt.
- **Attribute der Dokument-Ebene**
Auf der Dokument-Ebene werden Stilattribute durch `<lf:style.defaults>` spezifiziert. Nichtdefinierte Attribute (bzw. Attribute mit dem Wert `"default"`) erhalten ihre Werte von der Anwendung.
- **Attribute der Anwendung**
Werte, die auch auf der Dokument-Ebene fehlen, müssen von der Anwendung, z.B. einem Stylesheet, bereitgestellt werden oder fehlen ganz.

6.7.4. Ausrichtung von Daten in Zellen

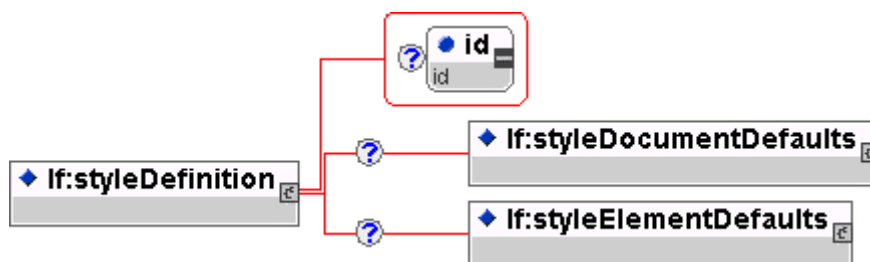
Die Ausrichtung wird nur auf der Ebene einer Zelle wirksam. Der Attributwert kann vom Elternknoten geerbt oder aus der Elementtyp-Ebene ersetzt werden. Er wirkt in gleicher Weise auf alle Datenfragmente, die zusammen den Inhalt der Zelle bilden. Kindknoten einer Zelle können keine eigene Ausrichtung besitzen.

6.7.5. Tabelle der Elementtypen

Element	Beschreibung
<code><lf:styleDefinition></code>	Stilattribute der Dokument- und Elementtyp-ebene
<code><lf:styleDocumentDefaults></code>	Stilattribute der Dokument-Ebene.
<code><lf:styleElementDefaults></code>	Stilattribute der Elementtyp-Ebene.
<code><lf:style.defaults></code>	Defaultwerte für Stilattribute.
<code><lf:style.elementtype></code>	Defaultwerte für Stilattribute eines Elementtyps.

6.7.6. Element `<lf:styleDefinition>`

Element	<code><lf:styleDefinition></code>	
Datentyp:	<i>element-content</i>	
Namespace:	<code>http://www.destatis.de/schema/tabml-layout/1.0</code>	DTD: lf
Inhaltsmodell:	(lf:styleDocumentDefaults?, lf:styleElementDefaults?)	



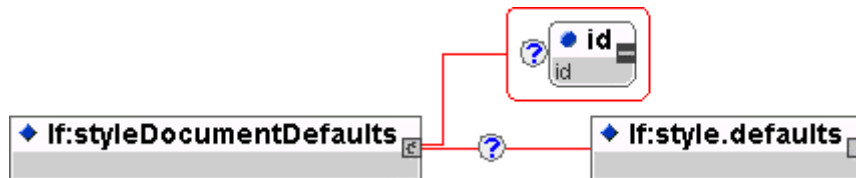
Attribute: S Datentyp, Default, Werte
 id ? *id*;

Details

`<lf:styleDefinition>` ist der Top-Level-Container für die Stilattribute der Dokument- und Elementtypenebene..

6.7.7. Element `<lf:styleDocumentDefaults>`

Element	<code><lf:styleDocumentDefaults></code>		
Datentyp:	<i>element-content</i>		
Namespace:	<code>http://www.destatis.de/schema/tabml-layout/1.0</code>	DTD:	<code>lf</code>
Inhaltsmodell:	<code>(lf:style.defaults?)</code>		



Attribute:	S	Datentyp, <u>Default</u>, Werte
<code>id</code>	?	<code>id;</code>

Details

Das Element `<lf:styleDocumentDefaults>` definiert die Dokumenttyp-Ebene der Stilattribute. Alle Stilattribute werden als Attribute des optionalen Kindknotens `<lf:style.defaults>` spezifiziert.

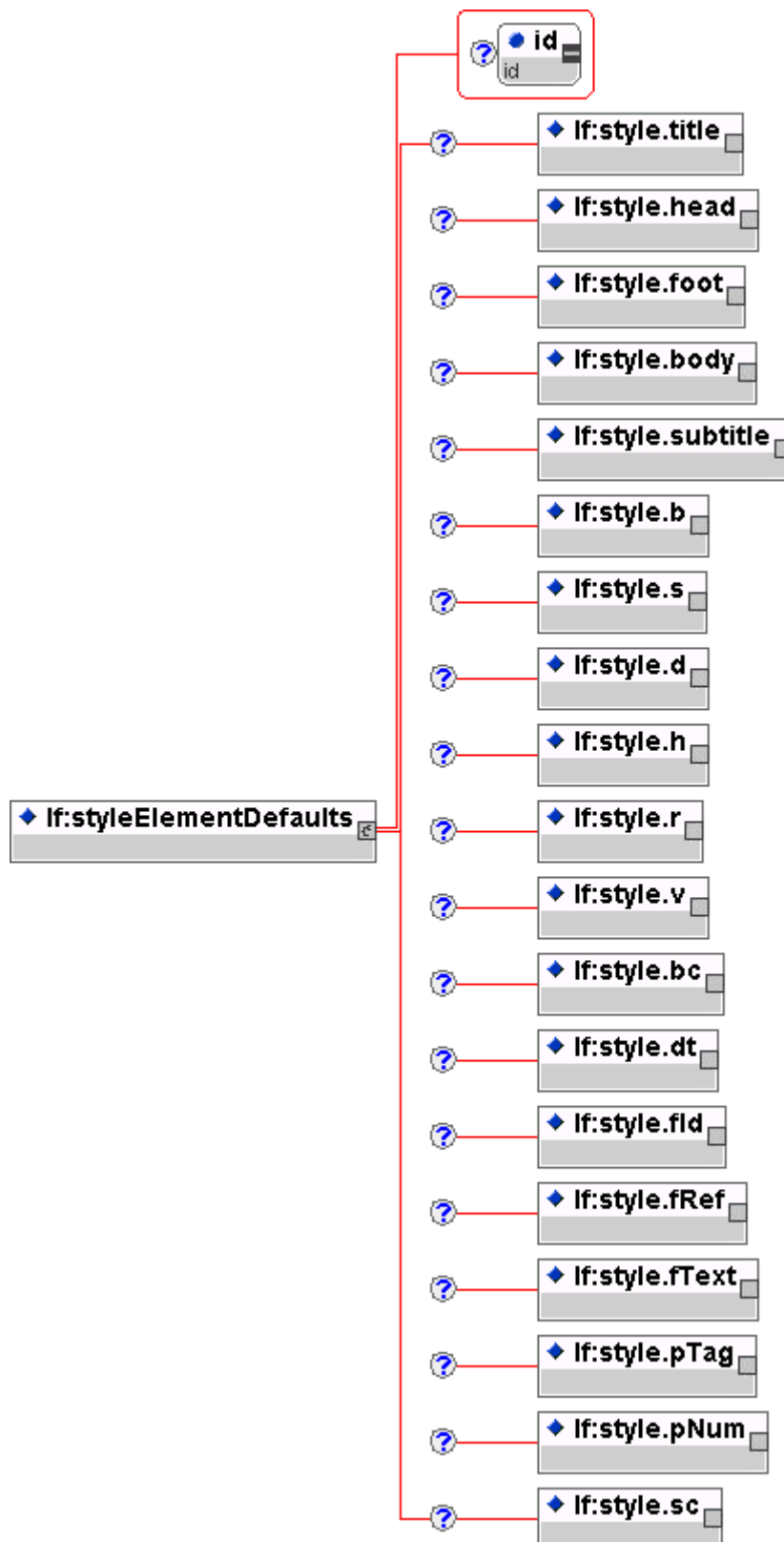
6.7.8. Element <lf:styleElementDefaults>

Element <lf:styleElementDefaults>

Datentyp: *element-content*

Namespace: <http://www.destatis.de/schema/tabml-layout/1.0>

DTD: lf



Inhaltsmodell:

```
( <!-- block element types -->
  <lf:style.title>?, <lf:style.head>?,
  <lf:style.foot>?, <lf:style.body>?,
  <lf:style.subtitle>?,

  <!-- cell element types -->
  <lf:style.b>?, <lf:style.d>?, <lf:style.h>?,
  <lf:style.r>?, <lf:style.s>?, <lf:style.v>?,

  <!-- data element types -->
  <lf:style.bc>?, <lf:style.dt>?, <lf:style.fId>?,
  <lf:style.fRef>?, <lf:style.fText>?,
  <lf:style.pTag>?, <lf:style.pNum>?
)
```

Attribute: **S** **Datentyp, Default, Werte**

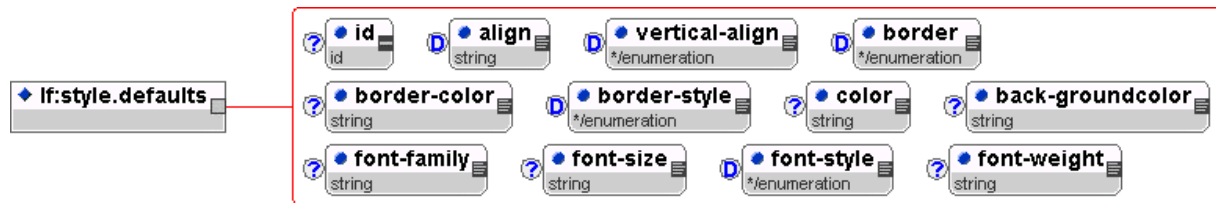
id ? *id*;

Details

<lf:styleDocumentDefaults> definiert die Elementtyp-Ebene der Stilattribute, m.a.W., es erlaubt die Festlegung elementtypspezifischer Stilattribute. Für jeden unten aufgelisteten Elementtyp *elementtype* ist die Angabe eines Kindknotens i.d.F. <lf:style.elementtype> möglich.

6.7.9. Element `<lf:style.defaults>`

Element	<code><lf:style.defaults></code>		
Datentyp:	<i>empty-element</i>		
Namespace:	<code>http://www.destatis.de/schema/tabml-layout/1.0</code>	DTD:	<code>lf</code>
Inhaltsmodell:	EMPTY		



Attribute:	S	Datentyp, <u>Default</u> , Werte
<code>id</code>	?	<code>id</code> ;
Stilattribute	?	Stilattribute
Ausrichtung	?	Ausrichtung von Daten

Details

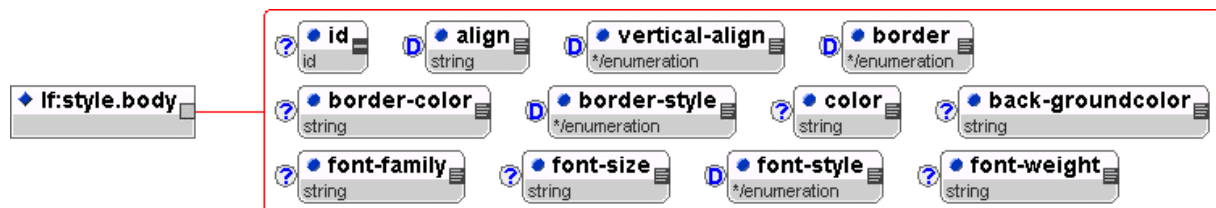
`<lf:style.defaults>` definiert Defaultwerte für Stilattribute der Dokument-Ebene.

6.7.10. Elemente `<lf:style.elementtype>`

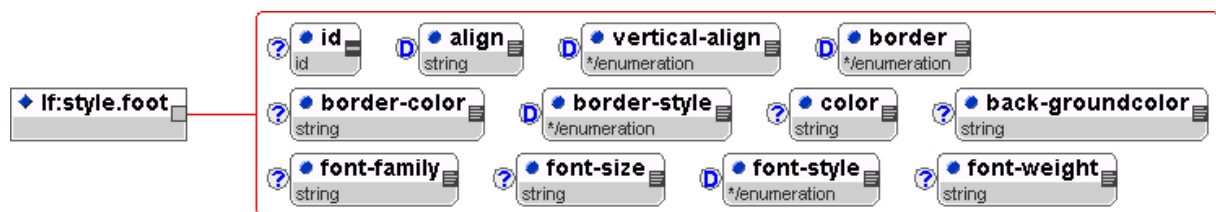
`<lf:style.elementtype>`-Elemente definieren Stilattribute für bestimmte Elementtypen der Block, In-Line- bzw. Zell- und Datenebene.

6.7.10.1. Blockebene

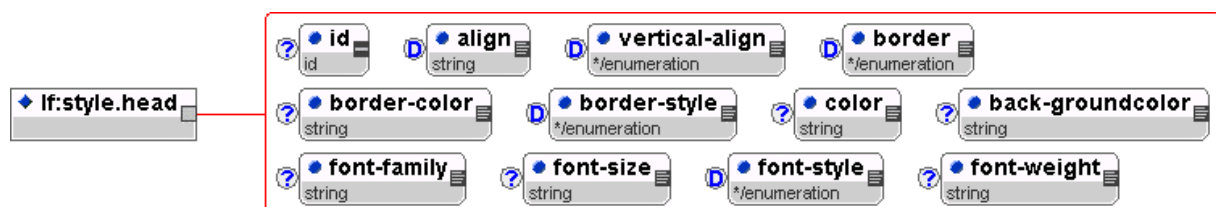
Elementtyp	Beschreibung
<code><lf:style.body></code>	Stilattribute für den Elementtyp <code><lf:body></code>



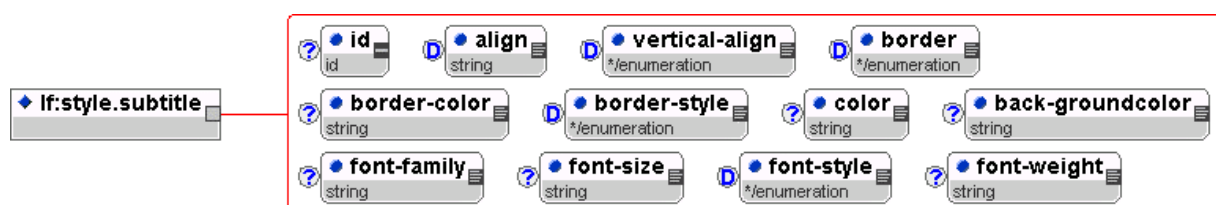
Elementtyp	Beschreibung
<code><lf:style.foot></code>	Stilattribute für den Elementtyp <code><lf:foot></code>



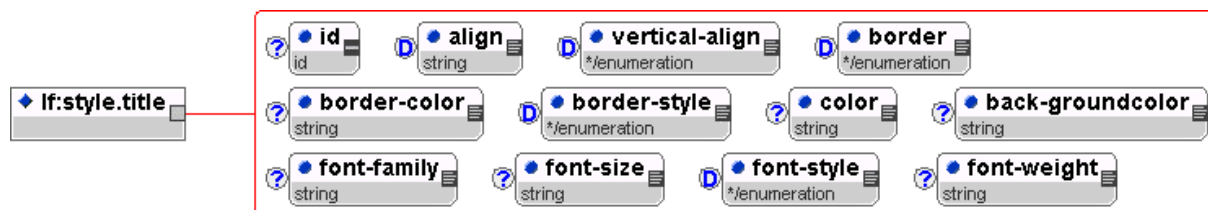
Elementtyp	Beschreibung
<code><lf:style.head></code>	Stilattribute für den Elementtyp <code><lf:head></code>



Elementtyp	Beschreibung
<code><lf:style.subtitle></code>	Stilattribute für den Elementtyp <code><lf:subtitle></code>

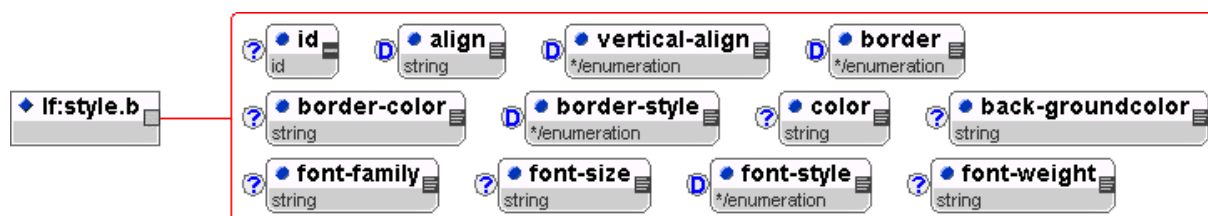


Elementtyp	Beschreibung
<lf:style.title>	Stilattribute für den Elementtyp <lf:title>

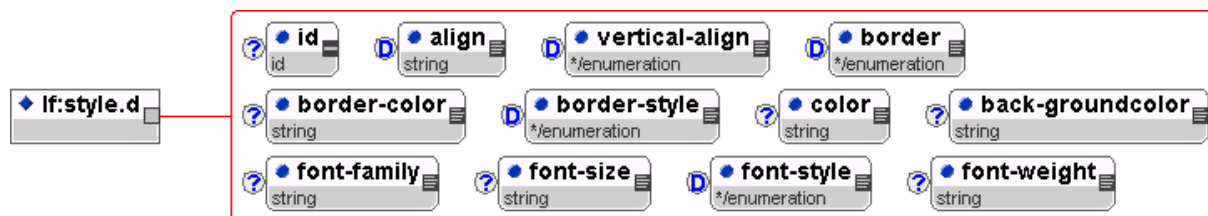


6.7.10.2. Zellebene

Elementtyp	Beschreibung
<lf:style.b>	Stilattribute für den Elementtyp <lf:b>



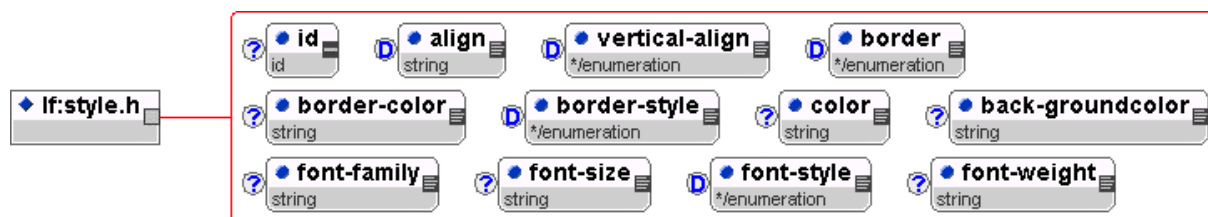
Elementtyp	Beschreibung
<lf:style.d>	Stilattribute für den Elementtyp <lf:d>



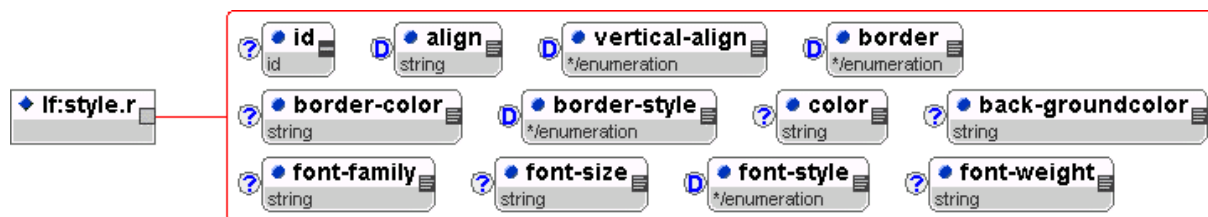
Elementtyp	Beschreibung
<lf:style.f>	Stilattribute für den Elementtyp <lf:f>

!!! Graphik !!!

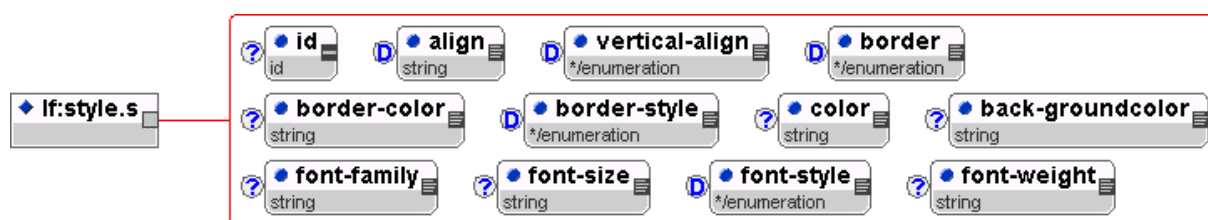
Elementtyp	Beschreibung
<lf:style.h>	Stilattribute für den Elementtyp <lf:h>



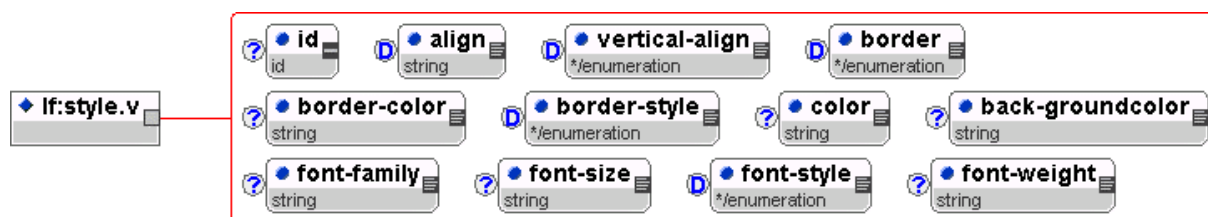
Elementtyp	Beschreibung
<lf:style.r>	Stilattribute für den Elementtyp <lf:r>



Elementtyp	Beschreibung
<lf:style.s>	Stilattribute für den Elementtyp <lf:s>

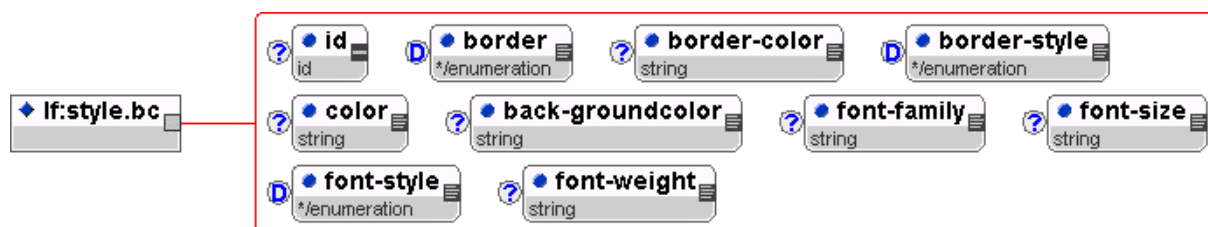


Elementtyp	Beschreibung
<lf:style.v>	Stilattribute für den Elementtyp <lf:v>



6.7.10.3. Datenebene

Elementtyp	Beschreibung
<lf:style.bc>	Stilattribute für den Elementtyp <lf:bc>

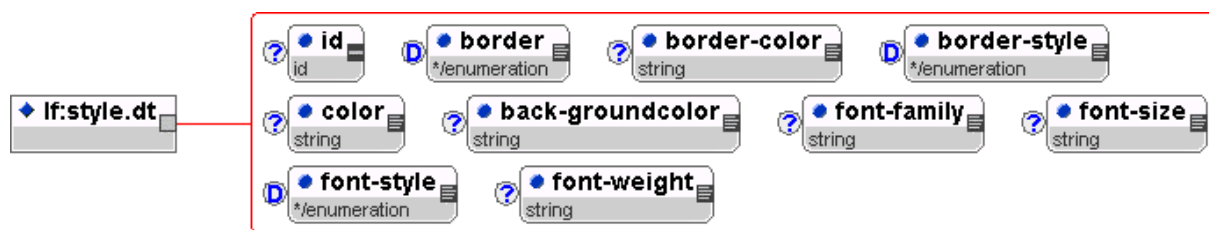


Elementtyp

<lf:style.dt>

Beschreibung

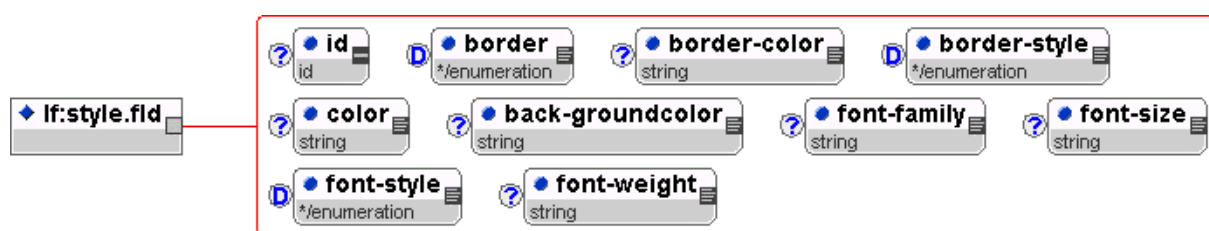
Stilattribute für den Elementtyp <lf:dt>

**Elementtyp**

<lf:style.fId>

Beschreibung

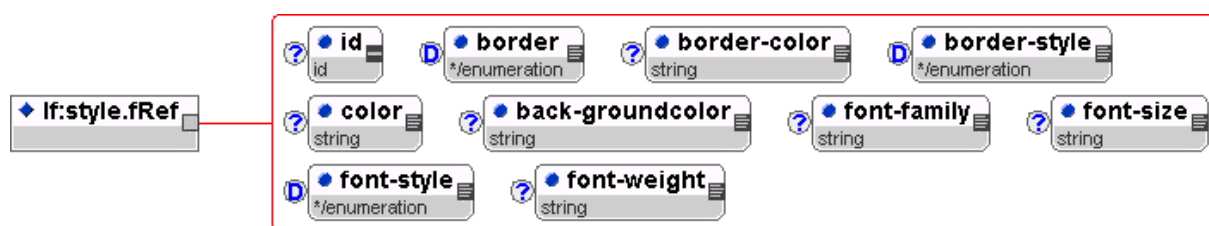
Stilattribute für den Elementtyp <lf:fId>

**Elementtyp**

<lf:style.fRef>

Beschreibung

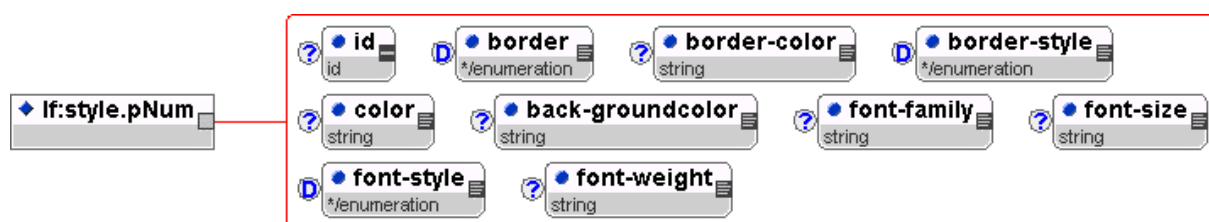
Stilattribute für den Elementtyp <lf:fRef>

**Elementtyp**

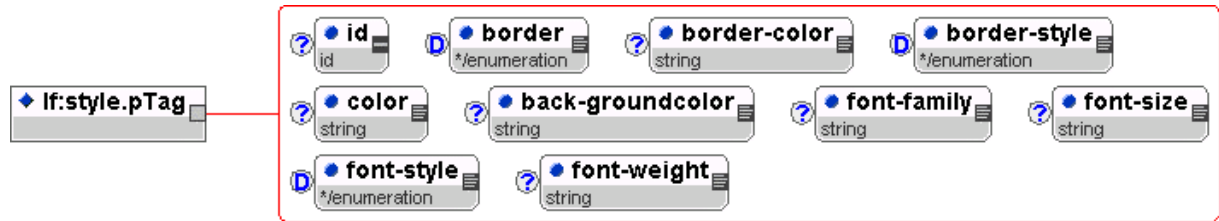
<lf:style.pNum>

Beschreibung

Stilattribute für den Elementtyp <lf:pNum>



Elementtyp	Beschreibung
<lf:style.pTag>	Stilattribute für den Elementtyp <lf:pTag>



6.7.10.4. Attribute

<lf:style.elementtype>-Elemente haben folgende Attribute:

Attribute:	S	Datentyp, <u>Default</u> , Werte
id	?	id;
Stilattribute	?	Stilattribute
Ausrichtung	?	Ausrichtung von Daten

6.8. Blockelemente: vertikale Segmentierung

6.8.1. Beschreibung

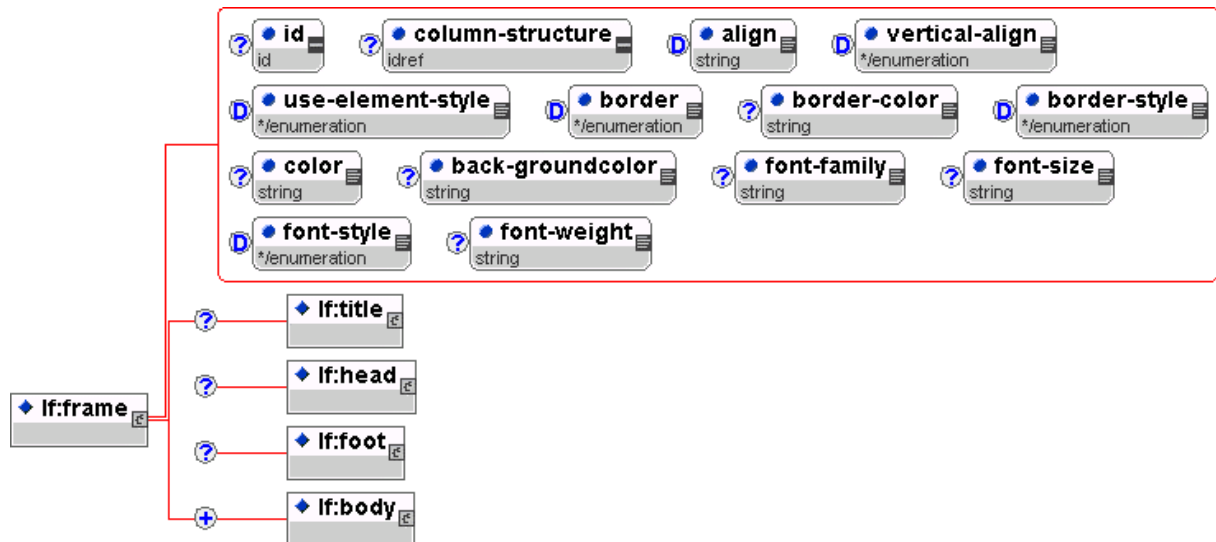
Elemente der Blockebene definieren die Tabellenstruktur bis herab auf einzelne Tabellenzeilen.

6.8.2. Tabelle der Elementtypen

Element	Beschreibung
<code><lf:frame></code>	Ein Top-Level-Element des Layoutformats, das einen Rahmen für die Anordnung der wichtigsten Komponenten der Tabellenstruktur darstellt; Überschrift (<code><lf:title></code>), Tabellenkopf (<code><lf:head></code>), Tabellenkörper (<code><lf:body></code>) und Tabellenfuß (<code><lf:foot></code>) sind immer einem Rahmen nachgeordnet.
<code><lf:title></code>	Eine Tabellenüberschrift
<code><lf:head></code>	Ein Tabellenkopf
<code><lf:foot></code>	Ein Tabellenfuß
<code><lf:body></code>	Ein Tabellenkörper
<code><lf:axis></code>	Ermöglicht es, Daten beliebig tief in hierarchisch angeordnete Kontexten zu kapseln und damit ihre Dimensionalität in der Dokumentstruktur widerzuspiegeln, sowie Untertabellen zu beschreiben.
<code><lf:subtitle></code>	Ermöglicht die Ablage von einer Zwischenüberschrift – als Kontextbeschreibung – in <code><lf:axis></code>
<code><lf:data></code>	Enthält Tabellenwerte und ggf. dazugehörige Vorspalten
<code><lf:row></code>	Enthält eine Tabellenzeile.

6.8.3. Element <lf:frame>

Element	<lf:frame>		
Datentyp:	<i>element-content</i>		
Namespace:	http://www.destatis.de/schema/tabml-layout/1.0	DTD:	lf
Inhaltsmodell:	(lf:title?, lf:head?, lf:foot?, lf:body+)		



Attribute	S	Datentyp, <u>Default</u> , Werte
id	?	id;
column-structure	?	idref; Verweist auf eine benannte oder (falls vorhanden) unbenannte Kolonnenbeschreibung: "" leer: unbenannte Kolonnenbeschreibung idref Verweis auf benannte Kolonnenbeschreibung
use-element-style	D	enumeration; <u>yes</u> no; siehe 6.11.5
Stilattribute	?	Stilattribute
Ausrichtung	?	Ausrichtung von Daten

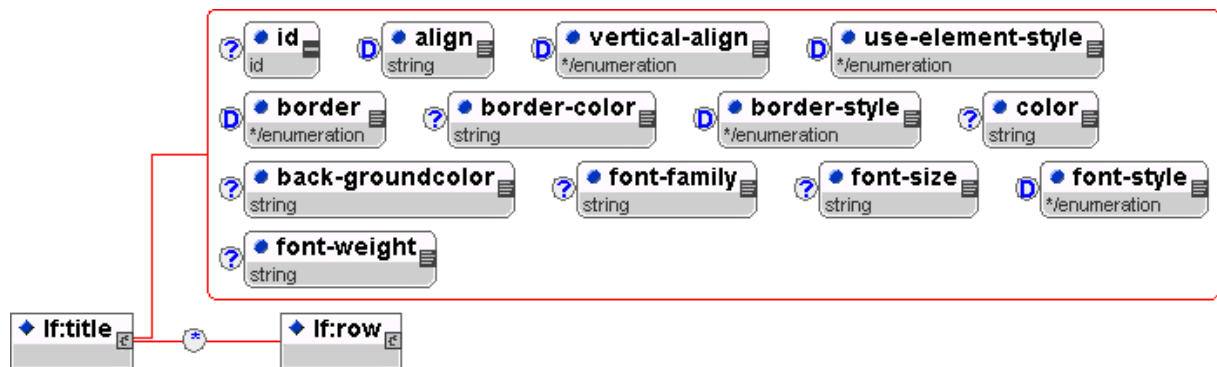
Details

<lf:frame> ist ein Rahmenelement, an welches Überschrift, Tabellenkopf, Tabellenkörper, Tabellenfuß und eine Kolonneneinteilung gebunden sind. Eine Tabelle kann aus einer beliebigen Anzahl von <lf:frame>-Elementen bestehen, und jedes kann eine Kolonnenbeschreibung referenzieren. <lf:frame>s erlauben es dadurch, Tabellen mit seiten- oder abschnittsweise variierenden Überschriften, Kolonnenstrukturen usw. zu beschreiben. Folgen von Seiten mit gleichen Überschriften, Kolonnenstrukturen usw. können als separate Tabellenkörper (<lf:body>) *innerhalb* <lf:frame> erscheinen. Die Seitengenerierung ist wahlweise auf der Basis von <lf:frame> und <lf:body> möglich.

Das Attribut `column-structure` referenziert ein `<lf:columnStructure>`-Element mit Kolonnendefinitionen. Es wird vom Elternknoten `<lf:layoutFormat>` geerbt.

6.8.4. Element `<lf:title>`

Element	<code><lf:title></code>		
Datentyp:	<i>element-content</i>		
Namespace:	<code>http://www.destatis.de/schema/tabml-layout/1.0</code>	DTD:	<code>lf</code>
Inhaltsmodell:	<code>(lf:row*)</code>		



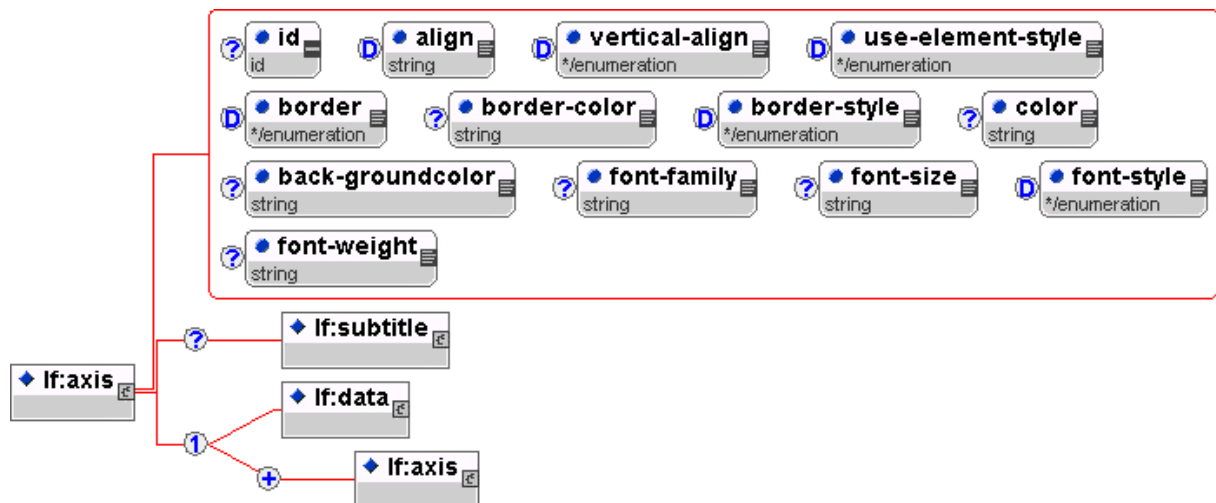
Attribute	S	Datentyp, <u>Default</u> , Werte
<code>id</code>	?	<i>id</i> ;
<code>use-element-style</code>	?	<i>enumeration</i> ; <u>yes</u> no; siehe 6.11.5
Stilattribute	?	Stilattribute
Ausrichtung	?	Ausrichtung von Daten

Details

`<lf:title>` enthält eine oder mehrere Zeilen einer Tabellenüberschrift.

6.8.5. Element `<lf:axis>`

Element	<code><lf:axis></code>		
Datentyp:	<i>element-content</i>		
Namespace:	<code>http://www.destatis.de/schema/tabml-layout/1.0</code>	DTD:	<code>lf</code>
Inhaltsmodell:	<code>(lf:subtitle?, (lf:data lf:axis+))</code>		



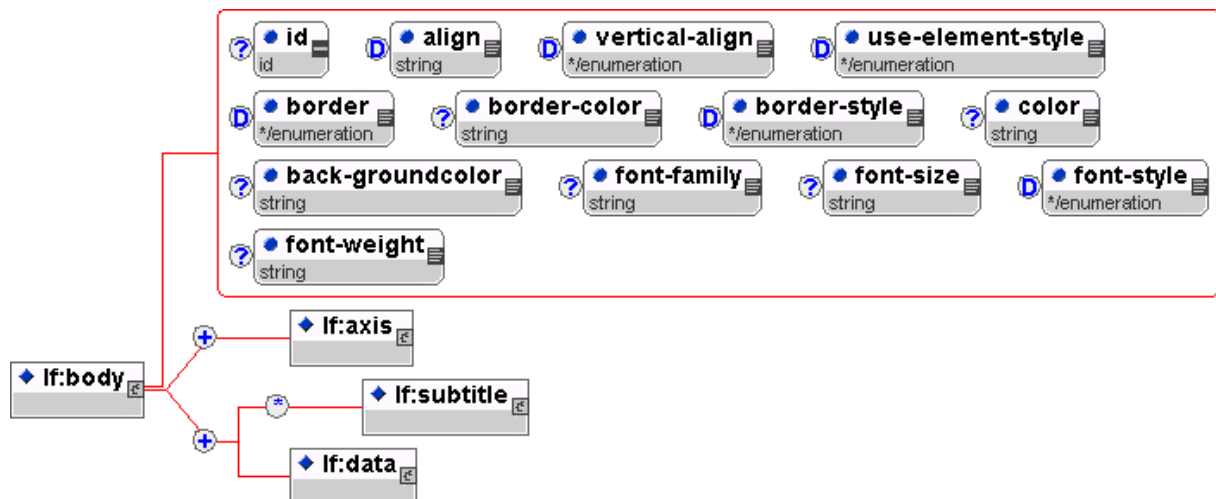
Attribute	S	Datentyp, <u>Default</u> , Werte
<code>id</code>	?	<code>id</code> ;
<code>use-element-style</code>	?	<i>enumeration</i> : <code>yes</code> <code>no</code> ; siehe 6.11.5
Stilattribute	?	Stilattribute
Ausrichtung	?	Ausrichtung von Daten

Details

Der Elementtyp `<lf:axis>` ermöglicht es, Tabellenwerte beliebig tief in eine hierarchische Struktur einzubetten. Auf jeder Ebene der Struktur kann eine Kontextinformation, i.d.R. eine Zwischenüberschrift, eingefügt werden. `<lf:data>` beendet die Rekursion, d.h., Tabellenwerte können nur auf der untersten Hierarchiestufe abgelegt werden. Im Ggs. zu einer sequentiellen Anordnung von Kontextinformation und Werten stellt `<lf:axis>` einen eindeutigen Bezug zwischen hierarchisch übergeordneten Kontextinformationen in der Dokumentstruktur her.

6.8.6. Element <lf:body>

Element	<lf:body>		
Datentyp:	<i>element-content</i>		
Namespace:	http://www.destatis.de/schema/tabml-layout/1.0	DTD:	lf
Inhaltsmodell:	(lf:axis+ (lf:subtitle*, lf:data)+)		



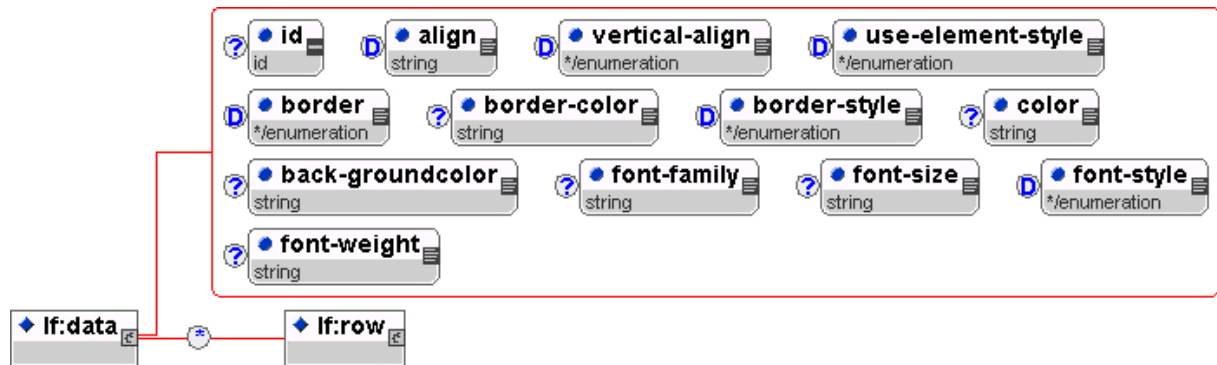
Attribute	S	Datentyp, <u>Default</u> , Werte
id	?	<i>id</i> ;
use-element-style	?	<i>enumeration</i> : <u>yes</u> no; siehe 6.11.5
Stilattribute	?	Stilattribute
Ausrichtung	?	Ausrichtung von Daten

Details

<lf:body> enthält einen Tabellenkörper und erlaubt die Ablage der Tabellenwerte mit ihren Vorspalten (<lf:data>) sowie von Kontextinformationen zu diesen Werten (<lf:subtitle>, i.d.R. Zwischenüberschriften). <lf:subtitle> und <lf:data> können wahlweise sequentiell oder hierarchisch in geschachtelten Elementen vom Typ <lf:axis> abgelegt werden. Für den Zweck der Layoutgenerierung dürfte die sequentielle Variante in den meisten Fällen ausreichen; die hierarchische Anordnung gibt hingegen die Beziehungen zwischen Werten und Kontextinformationen auch strukturell wieder und eignet sich deshalb besser zur Beschreibung von Untertabellen.

6.8.7. Element `<lf:data>`

Element	<code><lf:data></code>		
Datentyp:	<i>element-content</i>		
Namespace:	<code>http://www.destatis.de/schema/tabml-layout/1.0</code>	DTD:	<code>lf</code>
Inhaltsmodell:	<code>(lf:row*)</code>		



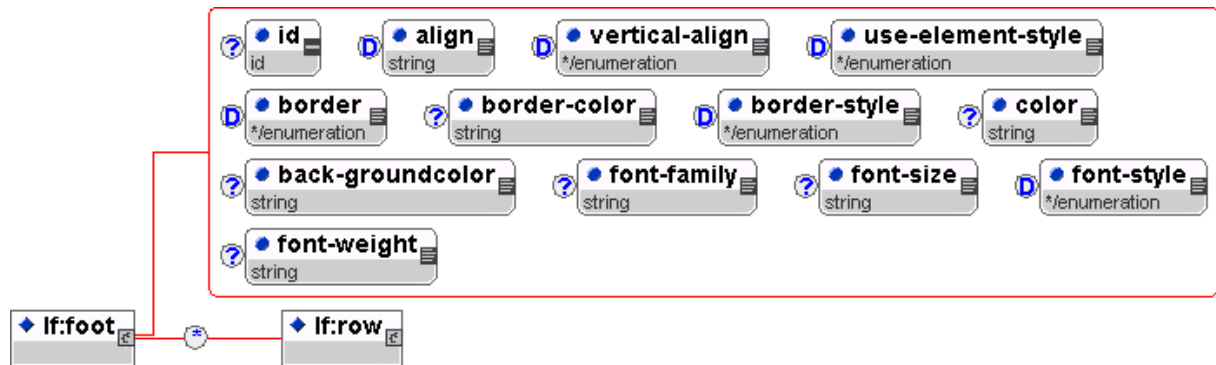
Attribute	S	Datentyp, <u>Default</u> , Werte
<code>id</code>	?	<i>id</i> ;
<code>use-element-style</code>	?	<i>enumeration</i> : <u>yes</u> no; siehe 6.11.5
Stilattribute	?	Stilattribute
Ausrichtung	?	Ausrichtung von Daten

Details

Das Element `<lf:data>` ist ein Container für Daten aus dem Tabellenkörper. In ihm stehen die Tabellenwerte und – falls vorhanden – die dazugehörenden Vorspalten

6.8.8. Element `<lf:foot>`

Element	<code><lf:foot></code>		
Datentyp:	<i>element-content</i>		
Namespace:	<code>http://www.destatis.de/schema/tabml-layout/1.0</code>	DTD:	<code>lf</code>
Inhaltsmodell:	<code>(lf:row*)</code>		



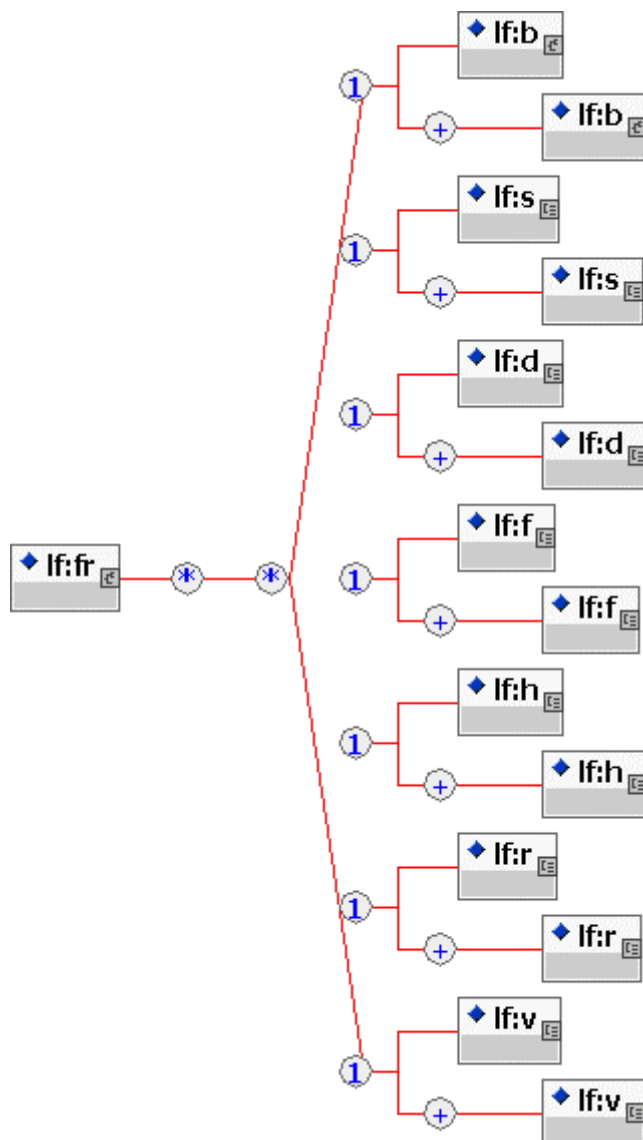
Attribute	S	Datentyp, <u>Default</u> , Werte
<code>id</code>	?	<i>id</i> ;
<code>use-element-style</code>	?	<i>enumeration</i> : <u>yes</u> no; siehe 6.11.5
Stilattribute	?	Stilattribute
Ausrichtung	?	Ausrichtung von Daten

Details

`<lf:foot>` enthält einen Tabellenfuß.

6.8.9. Element <lf:fr>

Element	<lf:fr>	
Datentyp:	<i>element-content</i>	
Namespace:	http://www.destatis.de/schema/tabml-layout/1.0	DTD: lf
Inhaltsmodell:	((lf:b,lf:b+) (lf:s,lf:s+) (lf:d,lf:d+) (lf:f,lf:f+) (lf:h,lf:h+) (lf:r,lf:r+) (lf:v,lf:v+))	



Details

Wenn ein Tabellenfeld eine Bahngrenze überschreitet, ist es notwendig, den Inhalt des Tabellenfeldes auf mehrere Zellen zu verteilen, da Zellen komplett innerhalb einer Bahn liegen müssen, und außerdem sichergestellt sein muß, an welchen Stellen der Inhalt des Tabellenfeldes durch Bahngrenzen geteilt wird. Umgekehrt soll es möglich sein, das Tabellenfeld inhaltlich und formal wiederherzustellen. Der Ele-

menttyp `<lf:fr>` übernimmt hier die Funktion, Zellen semantisch zu klammern, die aus der Aufteilung eines Tabellenfeldes an Bahngrenzen entstanden und daher Fragmente eines Tabellenfeldes sind. Er kann alle Zellelementtypen enthalten, aber es müssen mindestens zwei Zellen vorhanden sein, und alle müssen denselben Typ haben (da sie aus einem Tabellenfeld entstanden sind).

6.8.10. Element `<lf:head>`

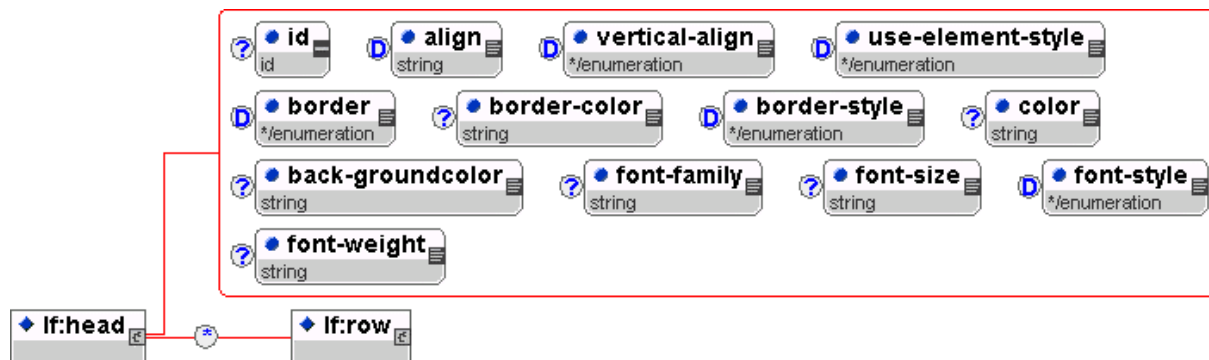
Element `<lf:head>`

Datentyp: *element-content*

Namespace: `http://www.destatis.de/schema/tabml-layout/1.0`

DTD: `lf`

Inhaltsmodell: `(lf:row*)`



Attribute

`id`

`use-element-style`

Stilattribute

Ausrichtung

S Datentyp, Default, Werte

? *id*;

? *enumeration*; yes|no; siehe 6.11.5

? Stilattribute

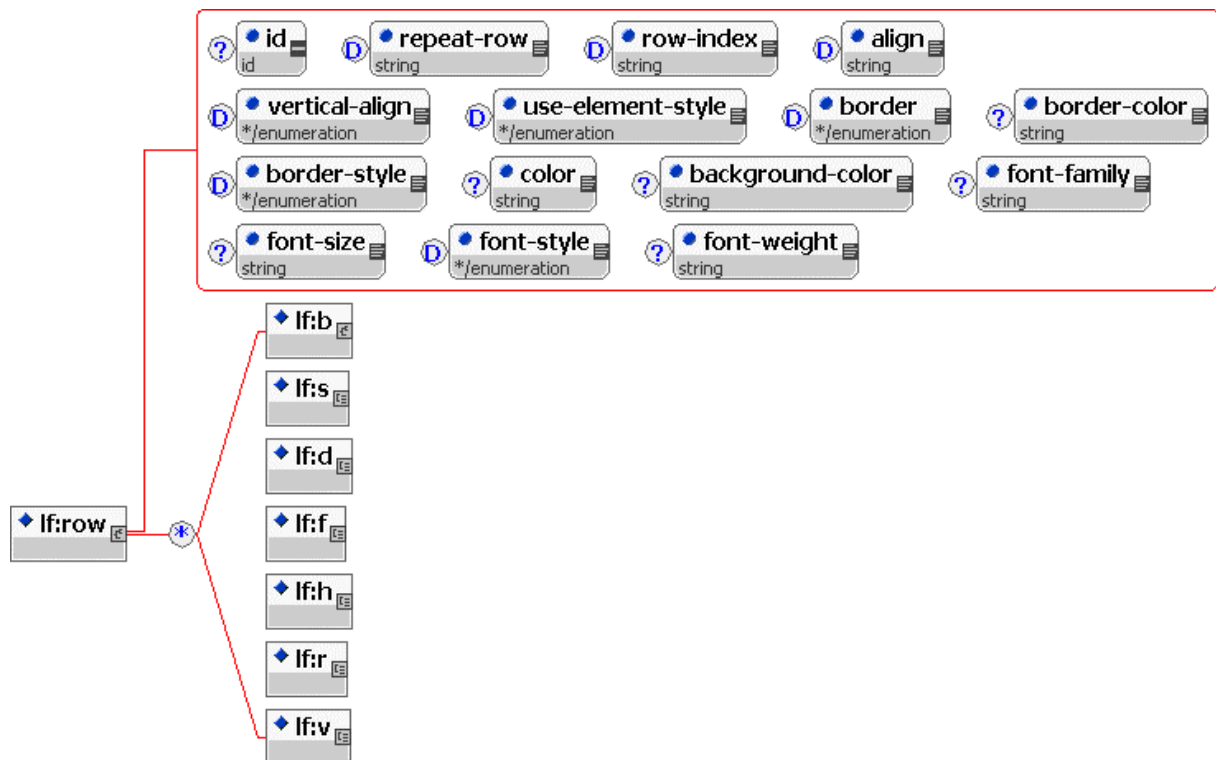
? Ausrichtung von Daten

Details

`<lf:head>` enthält den Kopfspaltenabschnitt einer Tabelle oder Tabellenseite.

6.8.11. Element `<lf:row>`

Element	<code><lf:row></code>		
Datentyp:	<i>element-content</i>		
Namespace:	<code>http://www.destatis.de/schema/tabml-layout/1.0</code>	DTD:	<code>lf</code>
Inhaltsmodell:	<code>(lf:b lf:s lf:d lf:f lf:h lf:r lf:v)*</code>		



Attribute

id
use-element-style
repeat-row

row-index

Stilattribute

Ausrichtung

S Datentyp, Default, Werte

? *id*;
 ? *enumeration*: yes | no; siehe 6.11.5
 ? *int_0..n*; Default:="1"; gibt an, wie oft die Zeile ausgegeben werden soll; "0" bewirkt Zeilenunterdrückung.
 ? *int_1..n*; Default:="" ; Index der Zeilennummer in der ursprünglichen Tabellenmatrix.

? Stilattribute

? Ausrichtung von Daten

Details

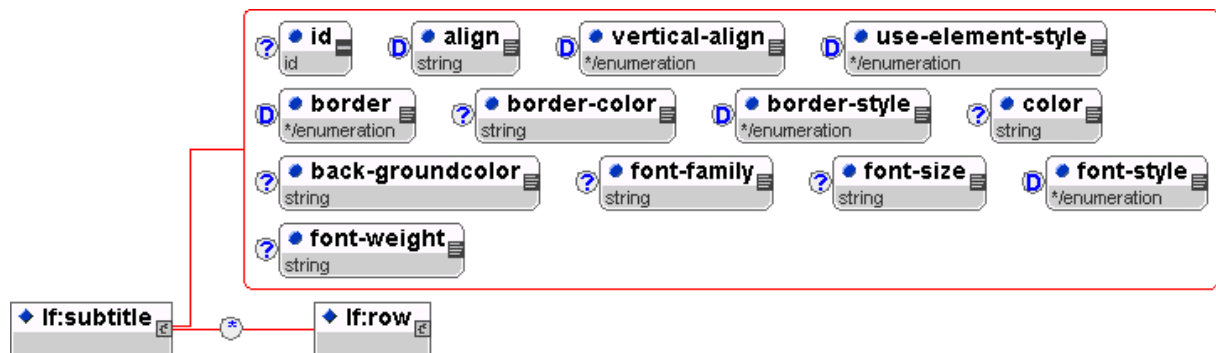
`<lf:row>` enthält die Zellen einer Tabellenzeile. Da Zellen mehrere Zeilen überspannen können, ist eine Tabellenzeile in diesem Sinne nicht unbedingt mit den Zeilen eines druckfähigen Formats identisch. Leerzeilen können durch ein leeres Element dargestellt werden. Zellen müssen in aufsteigender Reihenfolge ihrer (berechneten oder durch das Attribut `column-start` explizit angegebenen) Startkolonne in einer Tabellenzeile abgelegt werden.

Das Attribut `repeat-row` ermöglicht es, eine Zeile zu unterdrücken oder mehrfach auszugeben. Die Zeile darf keine Zellen mit einem `row-span` ungleich 1 haben, und es dürfen keine Zellen aus vorhergehenden Zeilen in die aktuelle Zeile reichen. Außer als Mittel der Zeilenunterdrückung sollte das Attribut typischerweise nur für die Komprimierung von Leerzeilen verwendet werden.

Das Attribut `row-index` enthält den Index der Tabellenzeile in der ursprünglichen Tabellenmatrix; dieser weicht i.d.R. von der Tabellenzeile des Layoutformates z.B. durch eingefügte Überschriftenzeilen und unterdrückte Wertezeilen ab.

6.8.12. Element `<lf:subtitle>`

Element	<code><lf:subtitle></code>		
Datentyp:	<i>element-content</i>		
Namespace:	<code>http://www.destatis.de/schema/tabml-layout/1.0</code>	DTD:	<code>lf</code>
Inhaltsmodell:	<code>(lf:row*)</code>		



Attribute	S	Datentyp, <u>Default</u> , Werte
<code>id</code>	?	<code>id</code> ;
<code>use-element-style</code>	?	<i>enumeration</i> : <code>yes</code> <code>no</code> ; siehe 6.11.5
Stilattribute	?	Stilattribute
Ausrichtung	?	Ausrichtung von Daten

Details

`<lf:subtitle>` enthält eine Zwischenüberschrift (i.d.R. eine Kontextinformation zu Tabellenwerten) und ist ein optionaler Kindknoten von `<lf:axis>` und `<lf:body>`. Es wird empfohlen, die Textinformation der Zwischenüberschrift in `<lf:s>`-Elementen abzulegen. Da `<lf:axis>` ein rekursives Element ist, ermöglicht es die hierarchische Anordnung von Zwischenüberschriften und den zugehörigen Werten.

6.9. Inline-Elemente: Tabellenzellen

6.9.1. Beschreibung

Eine Tabellenzeile besteht aus einer Folge von Tabellenzellen. Eine Tabellenzelle kann sich in vorausgehende und folgende Tabellenzeilen und über eine beliebige Anzahl von Kolonnen erstrecken. Für die Definition von Tabellenzellen stehen mehrere Elementtypen zur Verfügung, die jeweils einen eigenen Basisdatentyp für die Tabellenzelle repräsentieren.

Da alle Zellen Kindknoten von `<lf:row>` sind, kann das Vorkommen eines Zelltyps nicht über die Dokumentstruktur kontrolliert, z.B. auf einen bestimmten Block-Level-Typ wie `<lf:title>` usw. beschränkt, werden. Die Behandlung von Zellen, deren Datentyp mit dem Typ des nächsten übergeordneten Blockelementes inkonsistent ist, ist Sache der Anwendung.

6.9.2. Tabelle der Elementtypen

Element	Beschreibung
<code><lf:b></code>	Markiert eine Zelle, die <i>ausschließlich</i> Rahmenzeichen (<i>border characters</i>) oder Trennzeichen enthält.
<code><lf:d></code>	Markiert eine Zelle mit beliebigen Daten..
<code><lf:f></code>	Markiert einen Fußnotentext im Tabellenfuß.
<code><lf:h></code>	Markiert eine Zelle, die den Text einer Kopfspalte (<i>column header</i>) enthält.
<code><lf:r></code>	Markiert eine Zelle, die den Text einer Vorspalte (<i>row header</i>) enthält.
<code><lf:s></code>	Markiert eine Zelle, die den Text einer Zwischenüberschrift (<i>subtitle</i>) enthält.
<code><lf:v></code>	Markiert eine Zelle mit einem Tabellenwert (<i>value</i>)

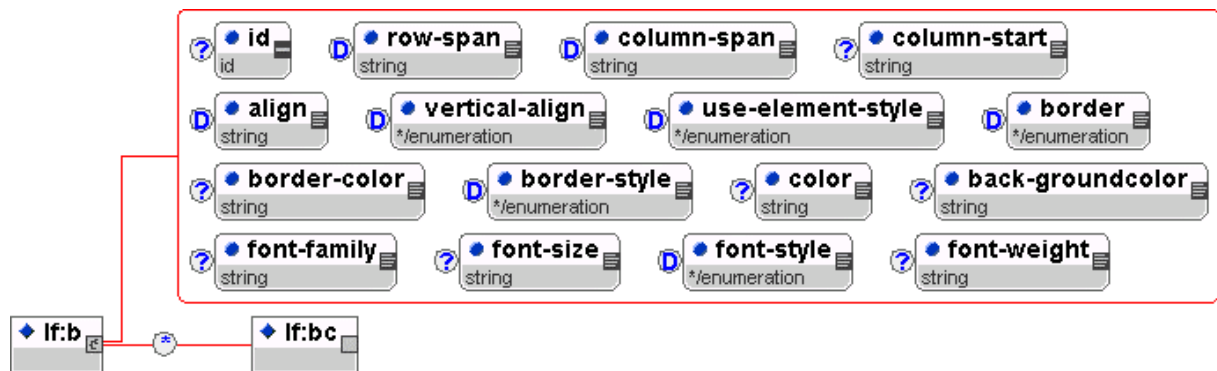
6.9.3. Zuordnung von Zell-Elementtypen zu Block-Level-Elementtypen

Das Vorkommen von Zell-Elementtypen *soll* anhand folgender Tabelle in Abhängigkeit vom nächsten übergeordneten Block-Level-Element beschränkt werden (alle Block-Level-Elemente, die unmittelbar Tabellenzeilen, also `<lf:row>`, enthalten).

Block-Level-Elementtyp	Zell-Elementtypen
<code><lf:data></code>	<code><lf:b></code> , <code><lf:d></code> , <code><lf:r></code> , <code><lf:v></code>
<code><lf:foot></code>	<code><lf:b></code> , <code><lf:d></code> , <code><lf:f></code>
<code><lf:head></code>	<code><lf:b></code> , <code><lf:d></code> , <code><lf:h></code>
<code><lf:subtitle></code>	<code><lf:b></code> , <code><lf:d></code> , <code><lf:s></code>
<code><lf:title></code>	<code><lf:b></code> , <code><lf:d></code> , <code><lf:h></code>

6.9.4. Element `<lf:b>`

Element	<code><lf:b></code>		
Datentyp:	<code>element-content</code>		
Namespace:	<code>http://www.destatis.de/schema/tabml-layout/1.0</code>	DTD:	<code>lf</code>
Inhaltsmodell:	<code>(lf:bc*)</code>		



Attribute

`id`

Zellenattribute

S Datentyp, Default, Werte

? `id`;

? Attribute der Elementtypen auf Zellebene

Details

`<lf:b>` markiert eine Tabellenzelle, deren Inhalt *ausschließlich* Rahmenzeichen oder Trennzeichen sind. Rahmen- und Trennzeichen können darüber hinaus – als „freie“ Daten oder durch das Element `<lf:bc>` explizit markiert – auch in Zellen anderer Typen vorkommen.

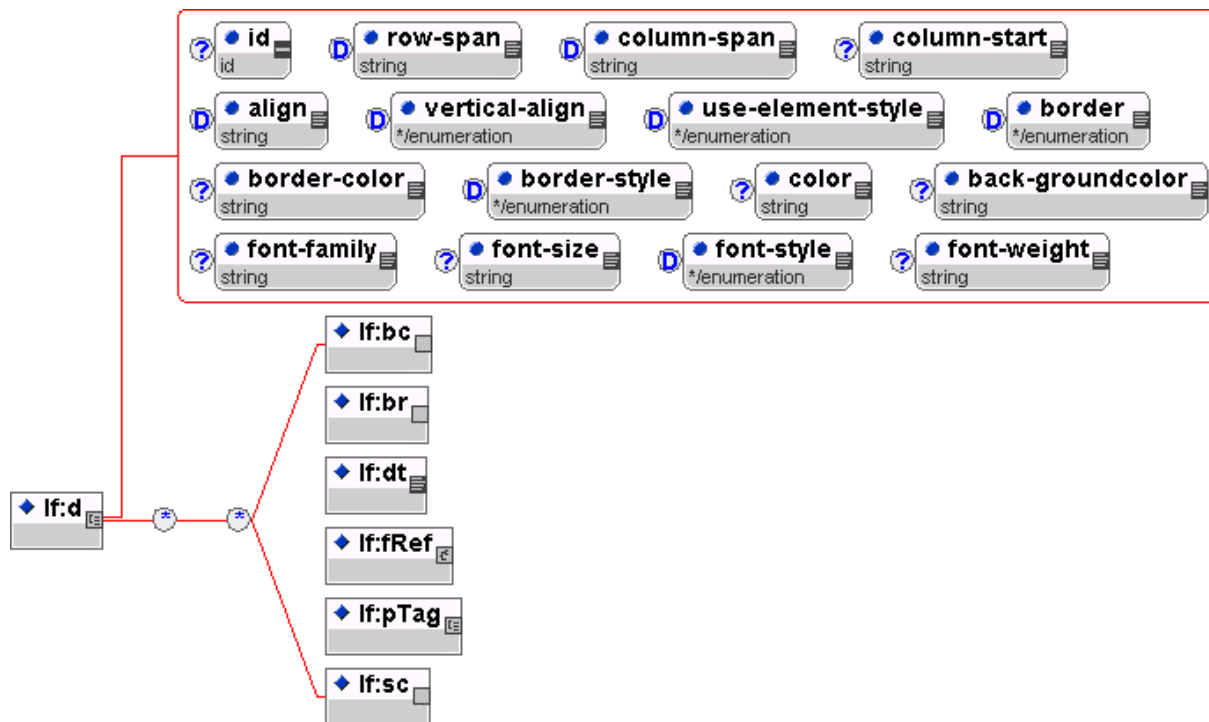
6.9.5. Element <lf:d>

Element <lf:d>

Datentyp: *mixed-content*

Namespace: <http://www.destatis.de/schema/tabml-layout/1.0> DTD: lf

Inhaltsmodell: (#PCDATA | lf:bc | lf:br | lf:dt | lf:fRef | lf:pTag | lf:sc) *



Attribute

`id`

Zellenattribute

S Datentyp, Default, Werte

? `id`;

? Attribute der Elementtypen auf Zellebene

Details

`<lf:d>` markiert eine Tabellenzelle, deren Inhalt beliebige Daten sind.

6.9.6. Element <lf:f>

!!! Graphik !!!

Element	<lf:f>		
Datentyp:	<i>mixed-content</i>		
Namespace:	http://www.destatis.de/schema/tabml-layout/1.0	DTD:	lf
Inhaltsmodell:	(#PCDATA lf:bc lf:br lf:dt lf:fRef lf:pTag lf:sc)*		

Attribute	S	Datentyp, <u>Default</u>, Werte
id	?	id;
Zellenattribute	?	Attribute der Elementtypen auf Zellebene

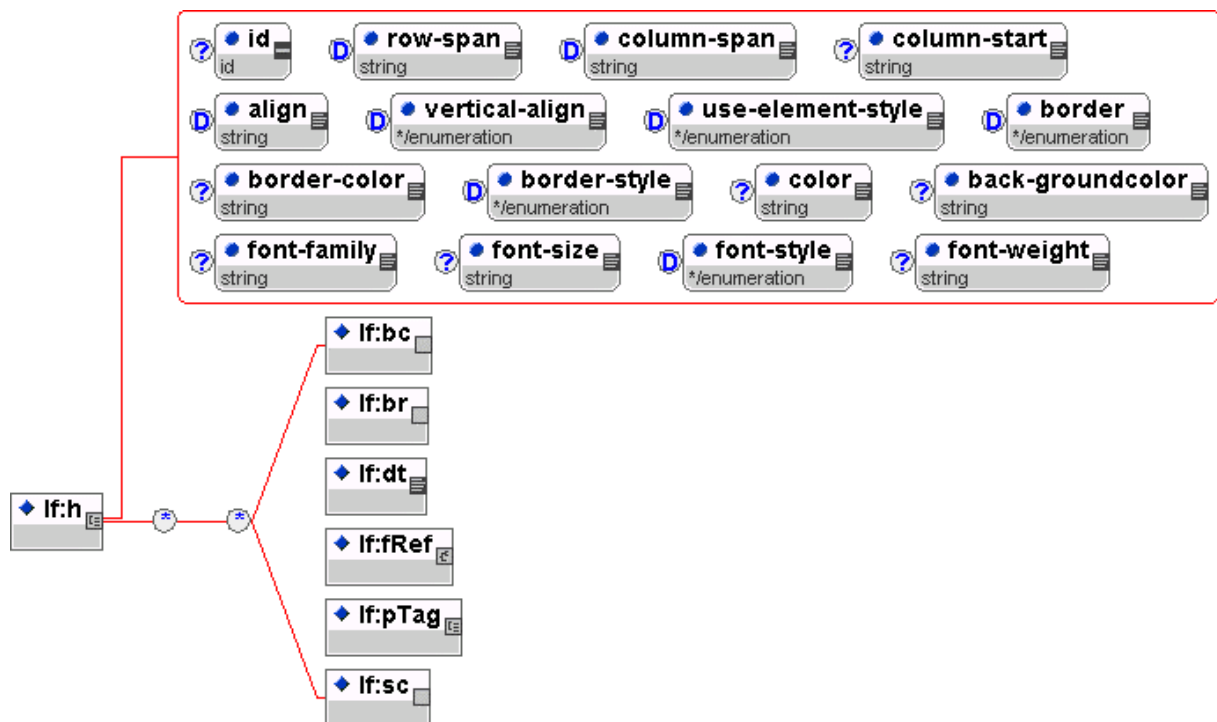
Details

<lf:f> markiert eine Tabellenzelle im Tabellenfuß (lf:foot), die einen Fußnoten Text enthält.

HINWEIS: <lf:f> bietet eine einfache Möglichkeit, Fußnotentexte im Tabellenfuß zu markieren. Wenn möglich, sollte der Verwendung von <lf:fRef> der Vorzug gegeben werden.

6.9.7. Element <lf:h>

Element	<lf:h>		
Datentyp:	<i>mixed-content</i>		
Namespace:	http://www.destatis.de/schema/tabml-layout/1.0	DTD:	lf
Inhaltsmodell:	(#PCDATA lf:bc lf:br lf:dt lf:fRef lf:pTag lf:sc) *		
Verwendung:	Eine Tabellenzelle, die einen Kopfspaltentext enthält.		



Attribute

`id`

Zellenattribute

S Datentyp, Default, Werte

? `id`;

? Attribute der Elementtypen auf Zellebene

Details

`<lf:h>` markiert eine Tabellenzelle, deren Inhalt ein Kopfspaltentext (Header) ist.

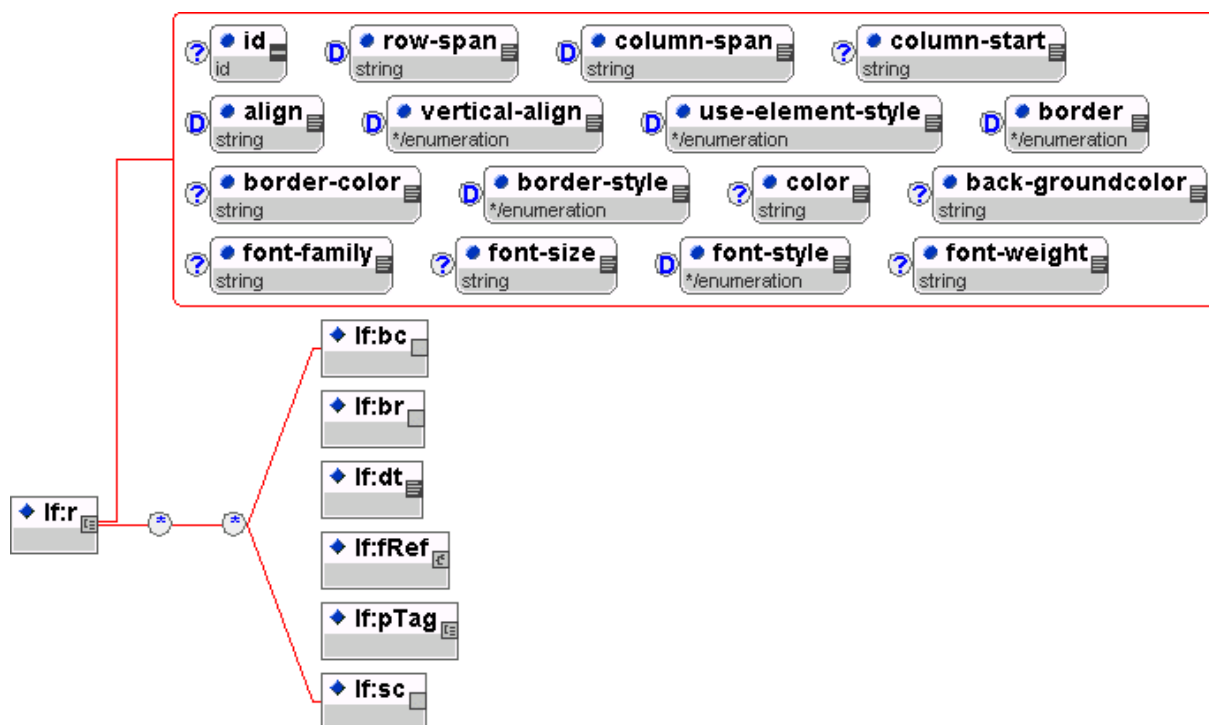
6.9.8. Element `<lf:r>`

Element `<lf:r>`

Datentyp: *mixed-content*

Namespace: `http://www.destatis.de/schema/tabml-layout/1.0` DTD: `lf`

Inhaltsmodell: `(#PCDATA | lf:bc | lf:br | lf:dt | lf:fRef | lf:pTag | lf:sc) *`



Attribute

`id`

Zellenattribute

S Datentyp, Default, Werte

? `id`;

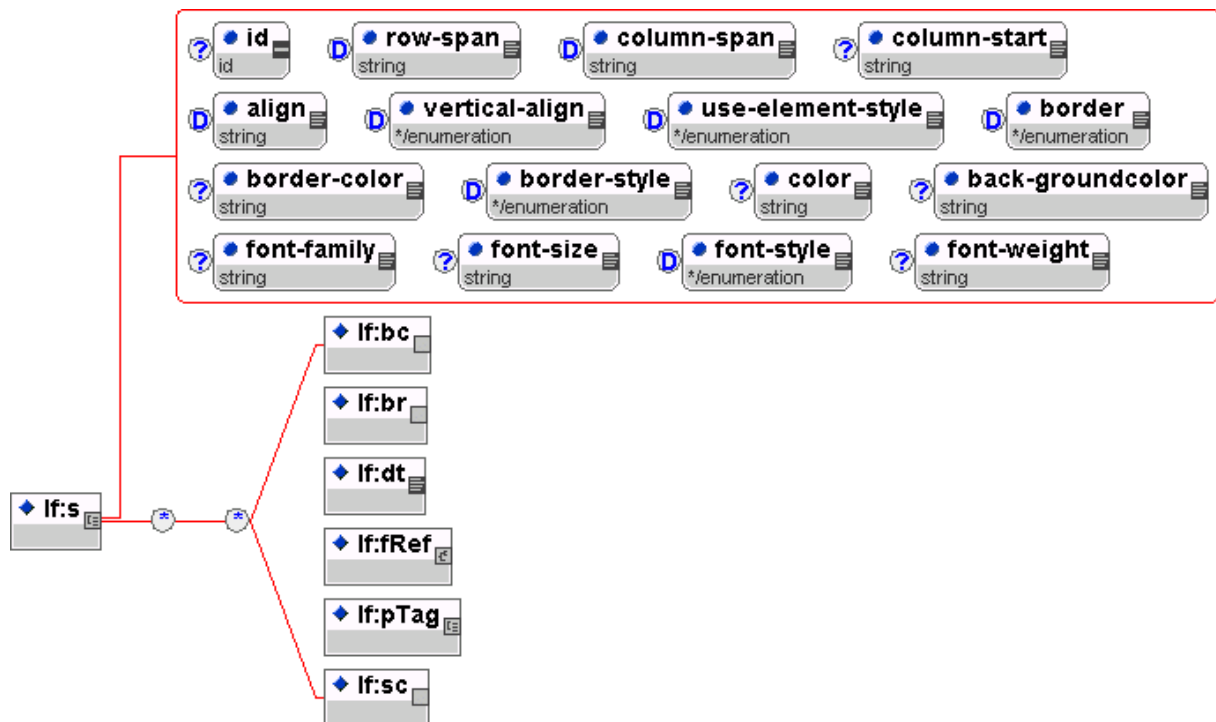
? Attribute der Elementtypen auf Zellebene

Details

`<lf:r>` markiert eine Tabellenzelle, deren Inhalt ein Vorspaltentext ist.

6.9.9. Element <lf:s>

Element	<lf:s>		
Datentyp:	<i>mixed-content</i>		
Namespace:	http://www.destatis.de/schema/tabml-layout/1.0	DTD:	lf
Inhaltsmodell:	(#PCDATA lf:bc lf:br lf:dt lf:fRef lf:pTag lf:sc) *		



Attribute

id

Zellenattribute

S Datentyp, Default, Werte

? id;

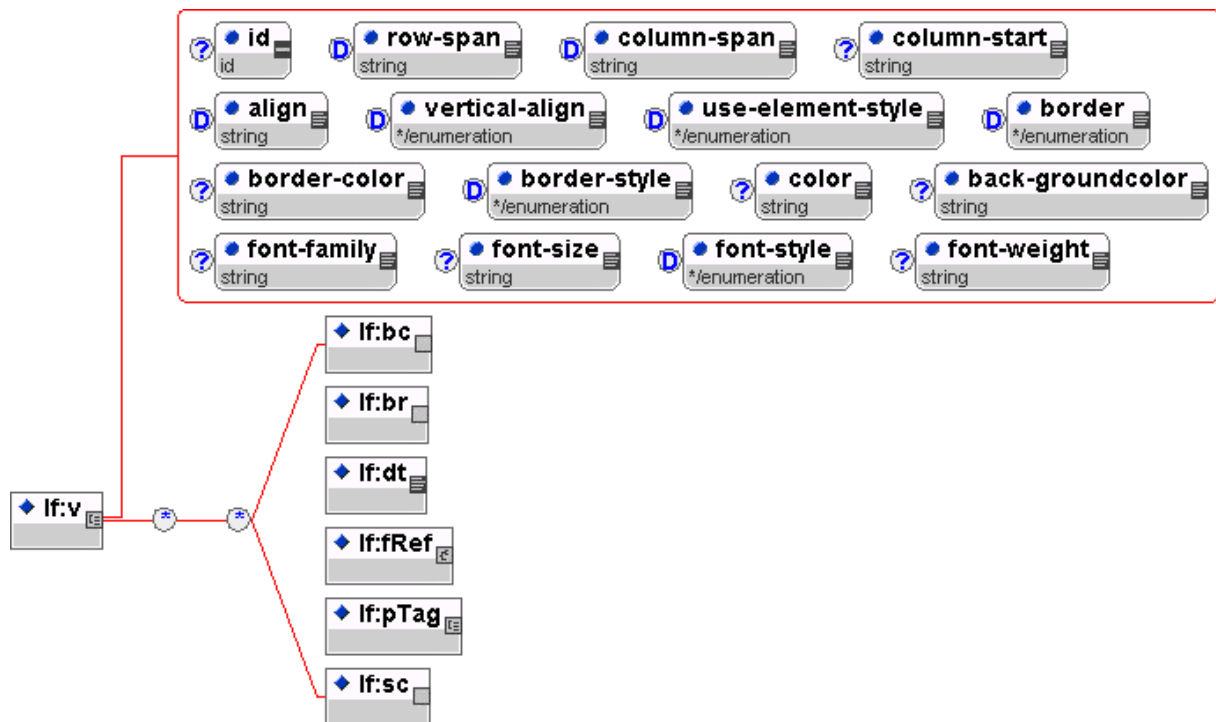
? Attribute der Elementtypen auf Zellebene

Details

<lf:s> markiert eine Tabellenzelle, die den Text oder einen Textbestandteil einer Zwischenüberschrift (subtitle) enthält.

6.9.10. Element `<lf:v>`

Element	<code><lf:v></code>		
Datentyp:	<i>mixed-content</i>		
Namespace:	<code>http://www.destatis.de/schema/tabml-layout/1.0</code>	DTD:	<code>lf</code>
Inhaltsmodell:	<code>(#PCDATA lf:bc lf:br lf:dt lf:fRef lf:pTag lf:sc) *</code>		



Attribute	S	Datentyp, <u>Default</u> , Werte
<code>id</code>	?	<code>id</code> ;
Zellenattribute	?	Attribute der Elementtypen auf Zellebene

Details

`<lf:v>` markiert eine Tabellenzelle, deren Inhalt ein Tabellenwert ist.

6.10. Datenelemente: Strukturierung von Zelleninhalt

6.10.1. Beschreibung

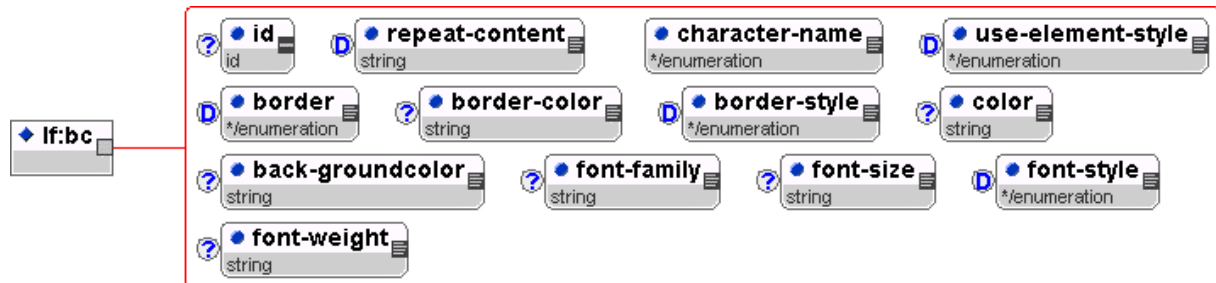
Der Inhalt einer Zelle kann ausschließlich Zeichendaten enthalten, aber auch durch eingebettetes Markup weiter strukturiert werden (sogenannter „mixed content“). Es ist außerdem möglich, eine Folge von Textzeilen in einer Zelle abzulegen.

6.10.2. Tabelle der Elementtypen

Element	Beschreibung
<lf:bc>	Enthält Rahmenzeichen.
<lf:br>	Signalisiert einen Zeilenumbruch im Textfluß einer Zelle.
<lf:dt>	Enthält eine Datums- und/oder Uhrzeitangabe.
<lf:fId>	Markiert ein Fußnotenkenzeichen.
<lf:fRef>	Markiert eine Fußnotenreferenz.
<lf:fText>	Enthält einen Fußnotentext.
<lf:pNum>	Enthält eine Seitennummer.
<lf:pTag>	Enthält eine Seitenkennung, optional mit <lf:pNum>

6.10.3. Element `<lf:bc>`

Element	<code><lf:bc></code>		
Datentyp:	<i>empty-element</i>		
Namespace:	<code>http://www.destatis.de/schema/tabml-layout/1.0</code>	DTD:	<code>lf</code>
Inhaltsmodell:	EMPTY		



Attribute	S	Datentyp, <u>Default</u> , Werte
id	?	<i>id</i> ;
repeat-content	?	<i>int_0..n</i> ; Default="1"; "0" unterdrückt die Ausgabe
character-name		<i>bc-def</i> ;
use-element-style		<i>enumeration</i> ; <u>yes</u> no; siehe
Stilattribute	?	Stilattribute

Details

`<lf:bc>` ist ein Platzhalterelement für eine Folge von 1-n Rahmenzeichen (border character). Rahmenzeichen treten auf, wenn ein Rahmen aus im Dokument vorhandenen Zeichen erzeugt wird, im Ggs. zu Rahmen, die nicht als Daten in einem Dokument vorhanden sind, sondern erst bei der Ausgabe, z.B. durch einen Browser, erzeugt werden.

Das Attribut `character-name` spezifiziert einen der Namen, die als Wert des Datentyps *bc-def* erlaubt sind und den Typ des Rahmenzeichens bestimmen. Das Attribut `character-name` legt damit fest, welches Rahmenzeichen durch das Element `<lf:bc>` repräsentiert wird. Bei der Darstellung kann jede Anwendung eine geeignete Kodierung bzw. Darstellung wählen.

Das Attribut `repeat-content` gibt an, wie oft das mit `character-name` spezifizierte Zeichen erzeugt werden sollen.

Beispiel

```
<lf:b><lf:bc character-name="rc"/></lf:b>
```

6.10.4. Element `<lf:br>`

Element	<code><lf:br></code>		
Datentyp:	<i>empty-element</i>		
Namespace:	<code>http://www.destatis.de/schema/tabml-layout/1.0</code>	DTD:	<code>lf</code>
Inhaltsmodell:	EMPTY		



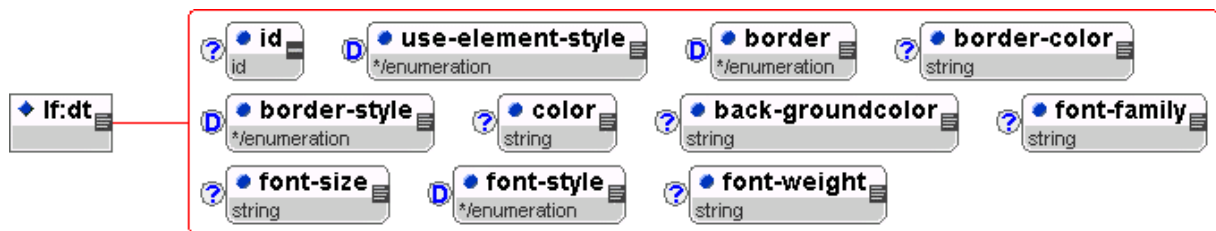
Attribute	S	Datentyp, <u>Default</u> , Werte
<code>id</code>	?	<code>id</code> ;

Details

Das Element `<lf:br>` (break) signalisiert einen Zeilenumbruch im Textfluß einer Zelle.

6.10.5. Element `<lf:dt>`

Element	<code><lf:dt></code>		
Datentyp:	<i>char</i>		
Namespace:	<code>http://www.destatis.de/schema/tabml-layout/1.0</code>	DTD:	<code>lf</code>
Inhaltsmodell:	(#PCDATA)		



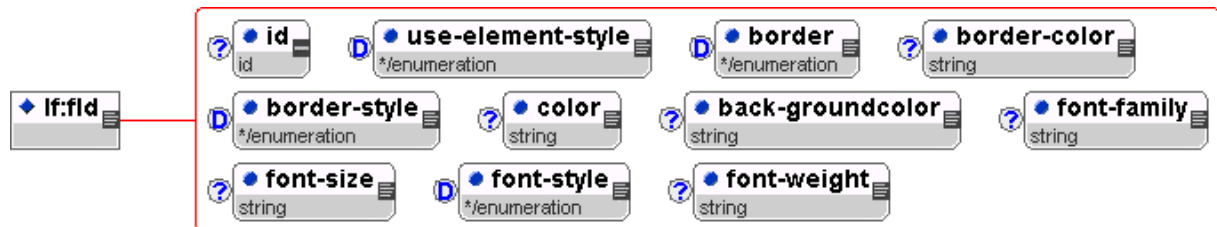
Attribute	S	Datentyp, <u>Default</u> , Werte
<code>id</code>	?	<code>id</code> ;
<code>use-element-style</code>		<i>enumeration</i> ; <u>yes</u> no; siehe
Stilattribute	?	Stilattribute

Details

`<lf:dt>` markiert eine Datums- und/oder Uhrzeitangabe

6.10.6. Element <lf:fId>

Element	<lf:fId>		
Datentyp:	<i>char</i>		
Namespace:	http://www.destatis.de/schema/tabml-layout/1.0	DTD:	lf
Inhaltsmodell:	(#PCDATA)		



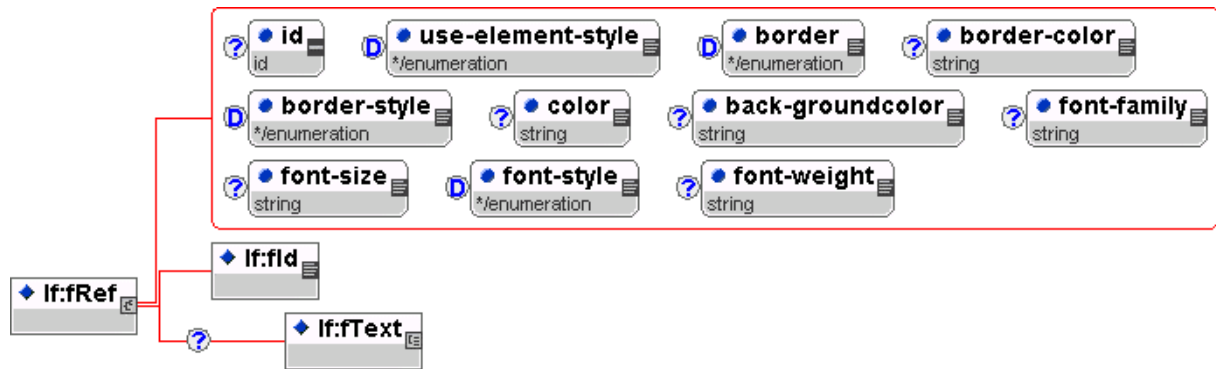
Attribute	S	Datentyp, <u>Default</u> , Werte
id	?	<i>id</i> ;
use-element-style		<i>enumeration</i> ; <u>yes</u> no; siehe 6.11.5
Stilattribute	?	Stilattribute

Details

<lf:fId> markiert ein Fußnotenkennzeichen.

6.10.7. Element <lf:fRef>

Element	<lf:fRef>		
Datentyp:	<i>element-content</i>		
Namespace:	http://www.destatis.de/schema/tabml-layout/1.0	DTD:	lf
Inhaltsmodell:	(lf:fId, lf:fText?)		



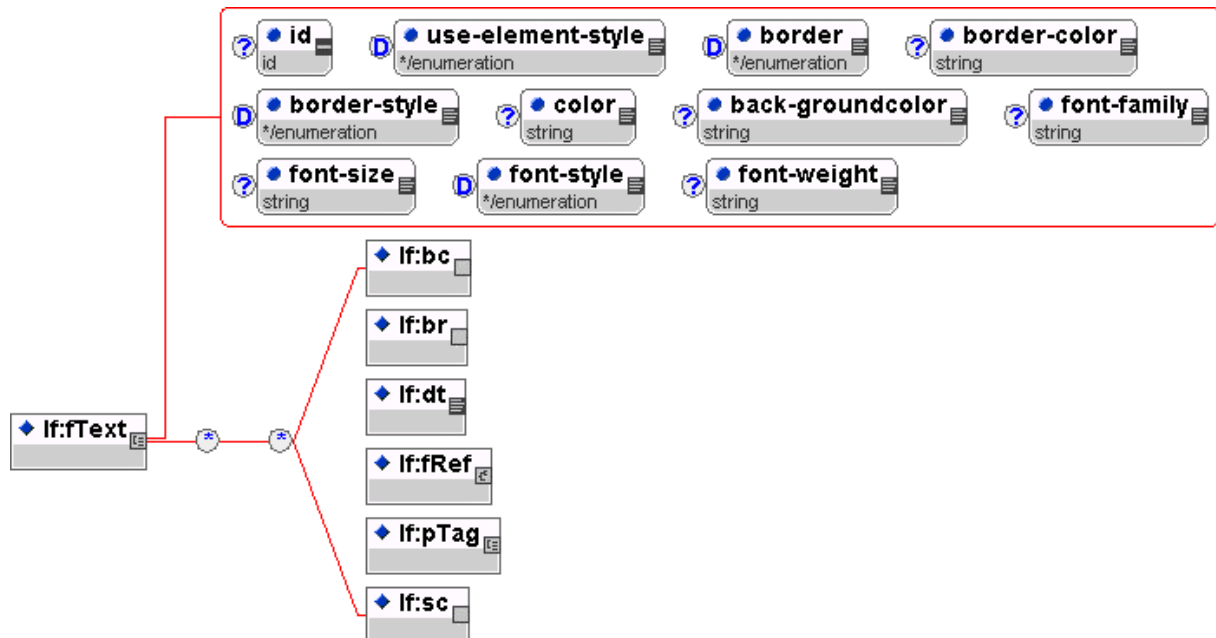
Attribute	S	Datentyp, <u>Default</u> , Werte
<code>id</code>	?	<code>id</code> ;
<code>use-element-style</code>		<i>enumeration</i> ; <u>yes</u> no; siehe 6.11.5
Stilattribute	?	Stilattribute

Details

<lf:fRef> markiert eine Referenz auf eine Fußnote.

6.10.8. Element `<lf:fText>`

Element	<code><lf:fText></code>		
Datentyp:	<i>char</i>		
Namespace:	<code>http://www.destatis.de/schema/tabml-layout/1.0</code>	DTD:	<code>lf</code>
Inhaltsmodell:	<code>(#PCDATA lf:bc lf:br lf:dt lf:fRef lf:pTag lf:sc) *</code>		



Attribute

`id`

`use-element-style`

Stilattribute

S Datentyp, Default, Werte

? `id`;

`enumeration`; `_yes` | `no`; siehe 6.11.5

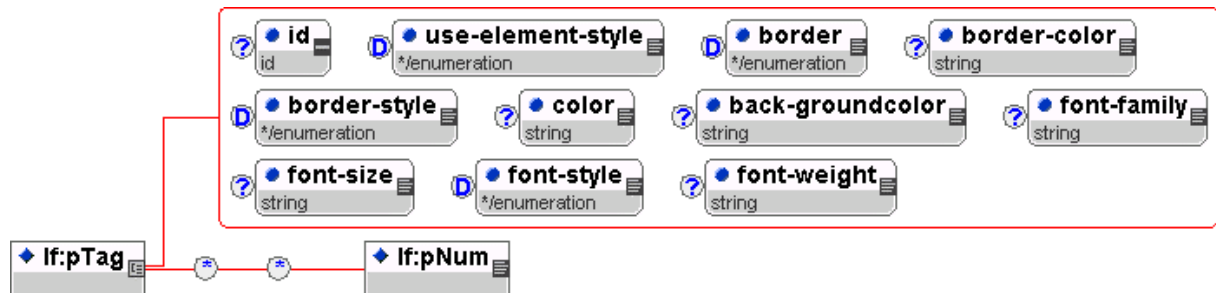
? Stilattribute

Details

`<lf:fText>` enthält den Text einer Fußnote

6.10.9. Element <lf:pTag>

Element	<lf:pTag>		
Datentyp:	<i>mixed-content</i>		
Namespace:	http://www.destatis.de/schema/tabml-layout/1.0	DTD:	lf
Inhaltsmodell:	(#PCDATA lf:pNum) *		



Attribute	S	Datentyp, <u>Default</u> , Werte
id	?	id;
use-element-style	D	enumeration; <u>yes</u> no; siehe 6.11.5
Stilattribute	?	Stilattribute

Details

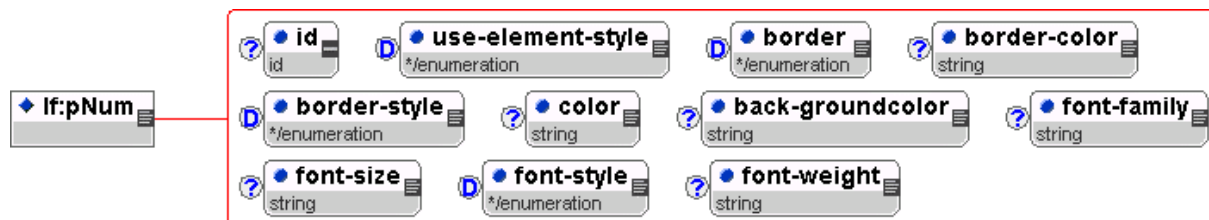
<lf:pTag> ist ein Container für eine Seitenkennung, also eine Angabe, die eine Seite identifiziert. Eine Seitennummer ist zwar nur optionaler Bestandteil einer solchen Seitenkennung, trotzdem dient <lf:pTag> vor allem dazu, eine Seitennummer und dazugehörige Texte in einen Zusammenhang zu stellen und als Einheit betrachten und bearbeiten zu können.

Beispiel:

```
<lf:pTag>Blatt Nr. <lf:pNum>1</lf:pNum></lf:pTag>
```

6.10.10. Element <lf:pNum>

Element	<lf:pNum>		
Datentyp:	<i>char</i>		
Namespace:	http://www.destatis.de/schema/tabml-layout/1.0	DTD:	lf
Inhaltsmodell:	(#PCDATA)		



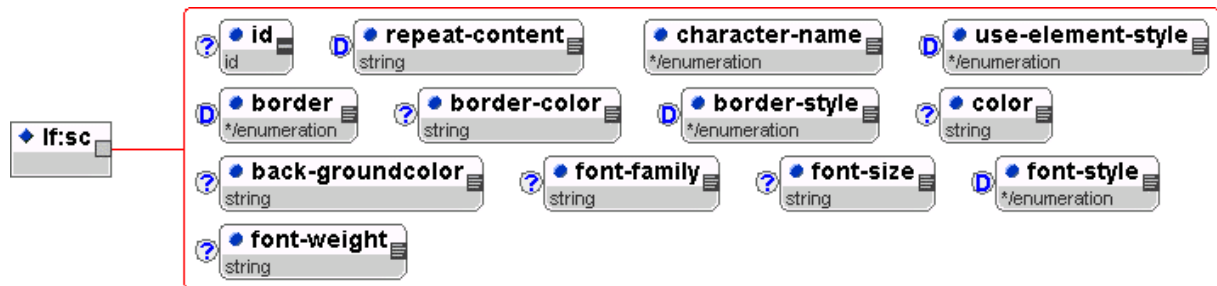
Attribute	S	Datentyp, <u>Default</u> , Werte
id	?	id;
use-element-style		enumeration; <u>yes</u> no; siehe 6.11.5
Stilattribute	?	Stilattribute

Details

<lf:pNum> markiert eine Seitennummer. Eine vorhandene Seitennummer kann entfernt bzw. ignoriert und eine fehlende erzeugt werden, je nach den Anforderungen der Layoutgenerierung.

6.10.11. Element <lf:sc>

Element	<lf:sc>		
Datentyp:	<i>empty-element</i>		
Namespace:	http://www.destatis.de/schema/tabml-layout/1.0	DTD:	lf
Inhaltsmodell:	EMPTY		

**Attribute**

id
repeat-content
character-name

S Datentyp, Default, Werte

? id;
? *int_0..n*; Default="1"; "0" unterdrückt die Ausgabe
? *enumeration*;

undefined

Nicht definiert

euro

Euro-Zeichen

use-element-style

enumeration; yes|no; siehe 6.11.5

Stilattribute

? Stilattribute

Details

<lf:sc> ist ein Platzhalterelement, das ein Sonderzeichen (special character) markiert. Z.Z. kann damit nur das Euro-Symbol repräsentiert werden.

6.11. Attributklassen

6.11.1. Beschreibung

TabML organisiert Gruppen von Attributen in Attributklassen, wenn diese typischerweise als Gruppe verwendet werden, z.B. Stilattribute, oder in einem funktionalen Zusammenhang stehen.

6.11.2. Attribute der Kolonnenbeschreibung

Attribute	S	Datentyp, <u>Default</u> , Werte
total-width	?	<i>int</i> ; Default:= "0"; Breite der Tabelle oder Bahn
column-width	?	<i>int</i> ; Default:= "0"; Breite der Kolonne(n)
column-type	?	<i>enumeration</i> : <u>undefined</u> header separator numbering data other
column-span	?	<i>int</i> ; Default:= "1"; Anzahl der Kolonnen

6.11.3. Allgemeine Attribute von Blockelementen

Attribute	S	Datentyp, <u>Default</u> , Werte
use-element-style	?	Verarbeitung von Elementtyp-Stilattributen
Stilattribute	?	Stillattribute
Ausrichtung	?	Ausrichtung von Daten

6.11.4. Allgemeine Attribute von Tabellenzellen

Attribute	S	Datentyp, <u>Default</u> , Werte
row-span	?	<i>int</i> ; Default:= "1"; Anzahl der Zeilen
column-span	?	<i>int</i> ; Default:= "1"; Anzahl der Kolonnen
column-start	?	<i>int</i> ; Default:= ""; Index der Startkolonne
use-element-style	?	Verarbeitung von Elementtyp-Stilattributen
Stilattribute	?	Stillattribute
Ausrichtung	?	Ausrichtung von Daten

6.11.5. Verarbeitung von Stilattributen steuern

Attribute	S	Datentyp, <u>Default</u> , Werte
use-element-style	?	<i>enumeration</i> : <u>yes</u> no Übernahme der Defaultwerte für Stilattribute aus der Elementtyp-Ebene (<style.elementtype>) steuern. yes Aus der Elementtyp-Ebene no Vom Elternknoten.

6.11.6. Stilattribute

Die in TabML definierten Stilattribute sind eine Untermenge aus CSS1/CSS2, sowohl bezüglich der Attributnamen als auch der Attributwerte.

6.11.6.1. Rahmen

Attribute	S	Datentyp, <u>Default</u> , Werte
border	?	<i>enumeration</i> : <u>none</u> top bottom horizontal vertical left right box <ul style="list-style-type: none"> ▪ <u>none</u> : keine Linie ▪ box : alle Seiten ▪ top : nur oben ▪ bottom : nur unten ▪ left : nur links ▪ right : nur rechts ▪ horizontal : oben und unten ▪ vertical : links und rechts
border-color	?	<i>color</i> ; Farbe des Rahmens
border-style	?	<i>enumeration</i> : <u>none</u> solid dashed dotted double groove inset outset ridge Das Attribut bestimmt, ob und wie Zellen durch Linien abgegrenzt werden: <ul style="list-style-type: none"> ▪ none : kein Rahmen ▪ solid : einfache Linie ▪ dashed : gestrichelt ▪ dotted : gepunktet ▪ double : doppelt ▪ groove : Pyramideneffekt nach außen ▪ inset : eingerückt ▪ outset : hervorgehoben ▪ ridge : Pyramideneffekt nach innen

6.11.6.2. Farbe und Hintergrund

Attribute	S	Datentyp, <u>Default</u> , Werte
color	?	<i>color</i> ; Schriftfarbe
background-color	?	<i>color</i> ; Hintergrundfarbe

6.11.6.3. Textgestaltung

Attribute	S	Datentyp, <u>Default</u> , Werte
font-family	?	<i>font-family</i> ; Schriftart(en)- und/oder Familie(n)
font-size	?	<i>font-size</i> ; Schriftgröße
font-style	?	<i>font-style</i> ; Schriftstil
font-weight	?	<i>font-weight</i> ; Schriftgewicht

6.11.7. Ausrichtung von Text in einer Zelle

Attribute	S	Datentyp, Default, Werte
align	?	<i>enumeration</i> : <u>center</u> left right <i>char_1..1</i> Es kann ein Zeichen angegeben werden, an dem der Text ausgerichtet wird.
padding-left	?	<i>int</i> ; Default:= "0"; Relativer Abstand des Zellinhaltes zum linken Kolonnenrand.
padding-right	?	<i>int</i> ; Default:= "0"; Relativer Abstand des Zellinhaltes zum rechten Kolonnenrand.
vertical-align	?	<i>enumeration</i> : <u>baseline</u> bottom middle top