

IT-Standards für die amtliche Statistik

Data Markup Language (DatML)

DatML/MAP 1.2

INI-basiertes Mapping

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

Status: Spezifikation

Stand: 14.06.2005

© Statistisches Bundesamt Wiesbaden, Deutschland

INI-basiertes Mapping von DatML/RAW

Spezifikation

Beschreibung

Für das generische Abbilden (Mapping) von DatML/RAW-Dokumenten auf „flache“ Datensätze ist es notwendig, einerseits die Datensätze und andererseits deren Beziehung zu Merkmalen und Metadaten des DatML/RAW-Dokuments zu beschreiben. Diese Spezifikation beschreibt ein für diese Zwecke geeignetes Format, welches auf dem INI-Format basiert und daher mit gewöhnlichen Texteditoren bearbeitbar ist. Es erlaubt, Ausgabedateien mit ein oder mehr Satzarten zu beschreiben und aus den Datensätzen von DatML/RAW-Rohdatennachrichten zu befüllen. Für den oben beschriebenen Zweck wird dieses Vorgehen als *INI-basiertes Mapping* bezeichnet.

Diese Spezifikation enthält außerdem eine Beschreibung von Erweiterungen in STATSPEZ-basierten Datensatzbeschreibungen und Werkzeugen, die das INI-basierte Mapping unterstützen.

HINWEIS: Dieses Format stellt eine Zwischenlösung dar, die mittelfristig durch eine XML-basierte Lösung (DatML/MAP) ersetzt werden soll (*XML-basiertes Mapping*).

Hinweise zur Version 1.2

Neben redaktionellen Änderungen enthält Version 1.2 folgende funktionale Erweiterungen:

- Erweiterte Templates für externe Dateinamen (3.12),
- Abfrage von Spannen in Berichtszeiträumen (3.2.5.1),
- Neue Funktionen `collapse()` und `normalize()` (3.9.1),
- Neue Metadaten `berichtszeitraum.bzr`, `berichtszeitraum.spanne` und `berichtszeitraum.string` (6.3.3), sowie entsprechende Schlüssel in Abschnitten vom Typ `kontext.kontext` (3.2.5).

Dokumentstatus

Dieses Dokument ist die verbindliche Spezifikation des INI-basierten Mappings von DatML/RAW-Dokumenten.

Teile dieses Dokumentes enthalten über die eigentliche Spezifikation hinausgehende Beschreibungen wie z.B. mögliche Implementierungsmodelle und sind daher nicht verbindlich; sie sind in der jeweiligen Überschrift mit der Zeichenfolge *[nicht-normativ]* markiert.

Inhaltsverzeichnis

1. Einführung	6
1.1. Glossar	6
1.2. Funktionen und Eigenschaften	7
1.3. Unterstützte Versionen von DatML/RAW	8
1.4. Metasyntax	9
1.4.1. Bezeichner und Literale	9
1.4.2. Metazeichen	9
2. Ein- und Ausgaben	10
2.1. Logisches Dateikonzept	10
2.2. INI-Dateien	11
2.3. Datendateien	11
2.3.1. Statistischer Kontext	11
2.3.2. Zuordnung Einzelmeldung - Dateideifinition	11
2.3.3. Zuordnung Meldungsdatensatz - Resultatdatensatz	12
2.3.4. Erzeugung von Datensätzen	13
2.3.4.1. Datensatz je Merkmalsgruppeninstanz erzeugen	13
2.3.4.2. Datensatz je Merkmal erzeugen	13
2.3.4.3. Feldgruppe je Merkmalsgruppeninstanz erzeugen	14
2.3.5. Speicherformat, Satzformat	14
2.4. Metadatendatei	14
2.5. Überblick (Beispiel)	15
3. Aufbau der INI-Dateien	16
3.1. INI-Meta	17
3.1.1. Abschnitt <code>meta.init</code>	17
3.1.2. Abschnitt <code>meta.kopf</code>	18
3.1.3. Abschnitt <code>meta.satz.n</code>	19
3.2. INI-Daten	21
3.2.1. Abschnitt <code>init</code>	21
3.2.2. Abschnitt <code>datei.init</code>	23
3.2.3. Abschnitt <code>datei.kopf</code>	25
3.2.4. Abschnitt <code>datei.satz.n</code>	26
3.2.5. Abschnitt <code>kontext.kontext</code>	27
3.2.5.1. Spannen in Berichtszeiträumen	29
3.3. Einzelfelder	30
3.3.1. Einzelfelddefinition	30
3.3.1.1. Operand <code>feldname</code>	30
3.3.1.2. Operand <code>splv-typ</code>	30
3.3.1.3. Operand <code>quelle</code>	31
3.3.1.4. Operand <code>default</code>	31
3.3.2. Beispiele	32
3.4. Strukturen	33
3.4.1. Strukturanfangsdefinition	33
3.4.1.1. Operand <code>strukturname</code>	33
3.4.1.2. Operand <code>strukturtyp</code>	33
3.4.1.3. Operand <code>mmgr_pfad</code>	34
3.4.2. Nachgeordnete Felddefinitionen	35
3.4.3. Strukturendefinition	35
3.4.4. Beispiele	35

3.5. Namen	38
3.6. Merkmale und Merkmalsgruppen	39
3.6.1. Merkmale	39
3.6.2. Merkmalsgruppen	39
3.6.3. Adressierung	39
3.7. Metadaten	43
3.8. Konstanten	44
3.9. Funktionen	45
3.9.1. Stringverarbeitung	45
3.9.1.1. collapse()	45
3.9.1.2. normalize()	45
3.9.1.3. split()	46
3.9.1.4. substring()	47
3.10. Reguläre Ausdrücke	48
3.10.1. Syntax	48
3.10.1.1. Atom	48
3.10.1.1.1. Literal	49
3.10.1.1.2. Escapesequenz	49
3.10.1.1.3. Zeichenklasse	50
3.10.1.1.4. Geklammerter regulärer Ausdruck	50
3.10.1.2. Quantor	50
3.10.1.3. Anker	51
3.10.1.4. Zweig	51
3.10.2. Beispiele	51
3.11. Schlüssel und Operanden	53
3.11.1. Schlüssel (Alphabetisch)	53
3.11.2. Operanden (Alphabetisch)	55
3.12. Templates für externe Dateinamen	57
4. Verarbeitung <i>[nicht-normativ]</i>	59
5. Einfaches Beispiel <i>[nicht-normativ]</i>	60
6. Metadaten nach Bereichen	61
6.1. Metadaten auf Dokumentebene	61
6.1.1. Metadatenkontext <code>absender.*</code>	61
6.1.2. Metadatenkontext <code>empfaenger.*</code>	62
6.1.3. Metadatenkontext <code>protokoll.*</code>	63
6.2. Metadaten auf Nachrichtenebene	64
6.2.1. Metadatenkontext <code>nachricht.*</code>	64
6.3. Metadaten auf Meldungsebene	65
6.3.1. Metadatenkontext <code>berichtsempfaenger.*</code>	65
6.3.2. Metadatenkontext <code>berichtspflichtiger.*</code>	65
6.3.3. Metadatenkontext <code>berichtszeitraum.*</code>	65
6.3.4. Metadatenkontext <code>erhebung.*</code>	66
6.3.5. Metadatenkontext <code>material.*</code>	66
6.3.6. Metadatenkontext <code>meldung.*</code>	66
6.4. Metadaten auf Satzebene	66
6.4.1. Metadatenkontext <code>satz.*</code>	66
6.5. Metadaten auf Merkmalsebene	67
6.5.1. Metadatenkontext <code>nichtantwort.*</code>	67
6.6. Metadaten auf Anwendungsebene	67
6.6.1. Allgemeine Metadaten	67

6.6.2. Metadatenkontext <code>dokument.*</code>	67
6.7. Metadatum <code>zeilen</code>	68
6.8. Aggregierte Metadaten	68
7. STATSPEZ-Unterstützung [<i>nicht-normativ</i>].....	70
7.1. Voraussetzungen.....	70
7.2. Vorgehensweise	70
7.3. Anweisungen	71
7.3.1. Syntax.....	71
7.3.1.1. Operanden.....	71
7.3.2. Gültigkeitsbereich	71
7.3.3. Anweisung <code>init</code>	72
7.3.4. Anweisung <code>kontext</code>	72
7.3.5. Anweisung <code>kopf</code>	72
7.3.6. Anweisung <code>satz</code>	73
7.3.7. Anweisung <code>xml</code>	73
7.4. Aufruf von Dsb2Ini	74
7.5. Organisatorisches.....	74
8. Änderungen seit Version 1.1	75
8.1. Kompatible Änderungen	75
9. Einschränkungen der Version 1.2.....	76
10. Anhang [<i>nicht-normativ</i>].....	77
10.1. Beispiele	77
10.1.1. INI-Daten	77
10.2. Mapping der Metadaten auf DatML/RAW-Elemente.....	80
10.2.1. Mapping für DatML/RAW 1.0.x	80
10.2.1.1. Metadatenkontext <code>absender.*</code>	80
10.2.1.2. Metadatenkontext <code>berichtspflichtiger.*</code>	81
10.2.1.3. Metadatenkontext <code>berichtsempfaenger.*</code>	81
10.2.1.4. Metadatenkontext <code>berichtszeitraum.*</code>	81
10.2.1.5. Metadatenkontext <code>empfaenger.*</code>	82
10.2.1.6. Metadatenkontext <code>erhebung.*</code>	82
10.2.1.7. Metadatenkontext <code>material.*</code>	82
10.2.1.8. Metadatenkontext <code>protokoll.*</code>	83
10.2.1.9. Metadatenkontext <code>satz.*</code>	83
10.2.2. Mapping für DatML/RAW 2.0.....	83
10.2.2.1. Metadatenkontext <code>absender.*</code>	83
10.2.2.2. Metadatenkontext <code>berichtspflichtiger.*</code>	85
10.2.2.3. Metadatenkontext <code>berichtsempfaenger.*</code>	85
10.2.2.4. Metadatenkontext <code>berichtszeitraum.*</code>	85
10.2.2.5. Metadatenkontext <code>empfaenger.*</code>	85
10.2.2.6. Metadatenkontext <code>erhebung.*</code>	86
10.2.2.7. Metadatenkontext <code>material.*</code>	86
10.2.2.8. Metadatenkontext <code>meldung.*</code>	86
10.2.2.9. Metadatenkontext <code>nachricht.*</code>	86
10.2.2.10. Metadatenkontext <code>nichtantwort.*</code>	86
10.2.2.11. Metadatenkontext <code>protokoll.*</code>	86
10.2.2.12. Metadatenkontext <code>satz.*</code>	87

1. Einführung

INI-basiertes Mapping erlaubt die Übertragung von Daten und Metadaten aus einem DatML/RAW-Dokument in sequentielle Ausgabedateien. Alle notwendigen Steuerinformationen, der Aufbau der Ausgabedatensätze und die Abbildung der Daten und Metadaten auf deren Felder sind in einer INI-Datei zu beschreiben (s. 3). Für den Zweck der einheitlichen Ausgabe von Metadaten über Erhebungs- und Verfahrensgrenzen hinweg kann für die Definition einer Metadatendatei eine zweite, separate INI-Datei verwendet werden.

1.1. Glossar

Die Kenntnis folgender Begriffe dient dem leichteren Einstieg in diese Spezifikation:

Begriff	Erläuterung
Anwendung	Eine geeignete softwaretechnische Lösung, die diese Spezifikation implementiert.
Dateidefinition	Die Definition <i>genau einer (logischen) Daten- bzw. Metadatendatei</i> in der INI-Daten oder der INI-Meta. Eine Dateidefinition enthält die Verarbeitungsparameter und alle Datensatzdefinitionen für eine Datei. Eine Dateidefinition besteht aus mehreren Abschnitten, die alle mit dem selben Namen beginnen.
Datensatzdefinition	Die Definition <i>genau eines Datensatzes</i> in einer Dateidefinition.
Datendatei	Eine logische Datei; sie entspricht einem aus Daten und/oder Metadaten bestehenden Resultatdatenstrom, der aus einer Dateidefinition in eine physikalische Datensenkfließt.
Datensatzbeschreibung	Die Beschreibung der <i>Satzstrukturen einer Datei</i> wie sie z.B. mit STATSPEZ oder dem Dienstprogramm DSB erzeugt und verwaltet werden.
Einzelfelddefinition	Die Definition <i>eines atomaren, also nicht zusammengesetzten Feldes eines Datensatzes</i> in der INI-Daten oder der INI-Meta.
Felddefinition	Die Struktur eines Datensatzes wird in einer Datensatzdefinition durch eine Folge von Felddefinition beschrieben. Dabei ist jede Felddefinition entweder eine Einzelfeld- oder eine Strukturdefinition.
Felddeklaration	Die Deklaration <i>eines Einzelfeldes oder einer Feldgruppe</i> in einer Datensatzbeschreibung.
Hilfsmerkmal	Im Sinne dieser Spezifikation ein DatML/RAW-Element vom Typ <code><hmm></code> , das ein meldungsglobales Hilfsmerkmal repräsentiert.
INI-Daten	Eine INI-Datei; sie enthält ein oder mehrere Dateidefinitionen von Datendateien.
INI-Meta	Eine INI-Datei; sie enthält genau eine Dateidefinitionen einer Metadatendatei.
Kontextdefinition	Die Definition eines Kontextes aus ausgewählten Elementen der Erhebungsbeschreibung und des Berichtszeitrau-

	mes zum Zweck der Filterung und Zuordnung von Meldungen zu Dateidefinitionen.
Metadatendatei	Eine logische Datei; sie entspricht einem <i>ausschließlich</i> aus Metadaten bestehenden Resultatdatenstrom der aus einer Dateidefinition in eine physikalische Datensenke fließt.
Meldungsdatenstrom	Der Datenstrom, der aus einer Einzelmeldung zu einer Dateidefinition fließt.
Merkmalsgruppe	Eine durch den DatML/RAW-Elementtyp <code><mmgr></code> zusammengefaßte, benannte Gruppe von ein oder mehr Wertmerkmalen und/oder weiteren Merkmalsgruppen.
Ordnungsmerkmal	Im Sinne dieser Spezifikation ein DatML/RAW-Element vom Typ <code><omm></code> , das ein den Meldungsdatensätzen übergeordnetes Merkmal als Ordnungskriterium repräsentiert.
Resultatdatenstrom	Der Datenstrom, der aus einer Dateidefinition in eine physikalische Datensenke fließt.
Resultatdatensatz	Ein abgeschlossener Datensatz innerhalb des Resultatdatenstromes.
Statistischer Kontext	Summe der Metadaten, die eine Einzelmeldung in einen für die statistische Verarbeitung relevanten Kontext stellt, insbesondere Erhebung und Berichtszeitraum. Eine Dateidefinition kann einen Kontext ausgewählter Metadaten verwenden und wird dann allen Meldungen zugeordnet, die dessen Kriterien erfüllen.
Strukturdefinition	Die Definition <i>einer Struktur, also eines nicht-atomaren, zusammengesetzten Feldes eines Datensatzes</i> in einer Datensatzdefinition.
Wertmerkmal	Im Sinne dieser Spezifikation ein DatML/RAW-Element vom Typ <code><mm></code> , das ein einzelnes statistisches Merkmal repräsentiert.

1.2. Funktionen und Eigenschaften

Die wichtigsten Funktionen und Eigenschaften sind:

- Zugriff auf Daten und Metadaten des DatML/RAW-Dokuments sowie auf Metadaten der *Anwendung*.
- Ausgabe von Daten und Metadaten in separate Dateien.
- Verarbeitung beliebig vieler Einzelmeldungen und Ausgabe in beliebig viele Daten- und Metadatendateien.
- Auswahl von Meldungen.
- Unterstützung der Speicherformate von STATSPEZ.
- Unterstützung von Dateien mit mehreren Satzarten.
- Belegung von Feldern mit vordefinierten Werten (Konstanten).
- Automatische Generierung der INI-Dateien aus STATSPEZ-DSBs.

Eigenschaften und Funktionalitäten in der Übersicht:

Eigenschaft, Funktion	Beschreibung
Format der Mappingdefinition	INI-Dateiformat, textbasiert

Satzarten in Ausgabedatei	Möglich
Auswahl von Meldungen	Möglich per Meldungskontext
Feldtypen in Ausgabedatei	ALN, NOV, NMV, GEP
Dateiformat der Ausgabe	EBCDIC, ASCII, ASCII-CSV, ASCII-BINAER
Strukturen und wiederholte Feldgruppen	Ja
Meldung-Datei Relation	1:n
Meldungssatz-Datensatz Relation	1:n
Daten	<ul style="list-style-type: none"> • Merkmale • Konstanten • Metadaten
Manipulation von Daten	<ul style="list-style-type: none"> • Verkettung • Teilzeichenketten • Leerraumbehandlung
Datendatei	ja
Metadatendatei	ja
Konfigurierbare Dateinamen	ja
Sonstige	<ul style="list-style-type: none"> • Manipulation des Feldtrennzeichens • Manipulation des Dezimalzeichens

1.3. Unterstützte Versionen von DatML/RAW

Diese Spezifikation ist gültig für DatML/RAW bis einschließlich Version 2.0. Der Umfang der verfügbaren Metadaten ist abhängig von der verwendeten DatML/RAW-Version.

Das INI-basierte Mapping beschreibt den logischen, kontextbasierten Zugriff auf Daten und Metadaten des DatML/RAW Datenmodells. Auch wenn in einigen Fällen auf Elemente des XML-Dokumenttyps DatML/RAW Bezug genommen wird, und sich die Namen der Kontexte und Metadaten an dessen Elemente und Strukturen anlehnen, sind Zugriffsschicht und konkrete Dokumentstruktur prinzipiell entkoppelt, da der Bezugspunkt das logische Datenmodell und nicht eine konkrete Implementierung ist. Dies vermeidet Abhängigkeiten und Inkompatibilitäten in den Fällen, in denen sich lediglich die Implementierung, nicht aber das Datenmodell ändert.

Es ist Aufgabe der Anwendung, DatML/RAW-Dokumente versionsgerecht auf das DatML/RAW-Datenmodell und dieses auf die logische Zugriffsschicht des INI-basierten Mappings abzubilden.

1.4. Metasyntax

Im folgenden wird die in der formalen Beschreibung von Bezeichnern und Metazeichen verwendete Metasyntax erläutert.

1.4.1. Bezeichner und Literale

Beispiele	Erläuterung
<code>satzformat</code> # . + u.s.w.	Für konstante Bezeichner, die für sich selbst stehen (Literale), wird <i>Courier New</i> , <i>blau</i> verwendet, die Groß-/Kleinschreibung ist zu beachten. Hierunter fallen auch (Folgen von) Einzelzeichen, sofern es keine Metazeichen sind (s. 1.4.2).
<code>rotationsdauer</code>	Für variable Bezeichner, die für einen anderen Wert stehen, wird <i>Courier New</i> , <i>blau</i> , <i>kursiv</i> verwendet.
<code>datei.init</code>	Variable und konstante Bezeichner können zusammen in einem Wert vorkommen.

1.4.2. Metazeichen

Für Metazeichen wird wie für konstante Bezeichner durchgehend *Courier New*, *blau* verwendet.

Metazeichen	Erläuterung
[]	Eckige Klammern schließen optionale Angaben ein.
{ }	Geschweifte Klammern schließen obligatorische Angaben ein; sie werden aber i.d.R. nur verwendet, wenn dies für die Eindeutigkeit erforderlich ist, z.B. bei Oder-Listen.
	Der senkrechte Strich trennt Alternativen .
..	Zwei aufeinanderfolgende Punkte kennzeichnen (optionale) Wiederholungen .
''	Hochkommata schließen Metazeichen ein, wenn diese die Bedeutung eines Literales haben: <code>'{meta}'</code> wird z.B. ersetzt durch <code>{#berichtszeitraum.jahr}</code> .

In regulären Ausdrücken gelten besondere Metazeichen (s. Abschnitt 3.10).

2. Ein- und Ausgaben

Auf der Eingabeseite eines INI-basiertes Mappings gibt es ein oder zwei INI-Dateien für die Verarbeitungssteuerung und genau ein DatML/RAW-Dokument mit einer beliebigen Anzahl von Einzelmeldungen. Die INI-Dateien beschreiben mittels sogenannter *Dateidefinitionen* die auszugebende Daten- bzw. Metadatendateien; sie steuern u.a. auch die Auswahl der Einzelmeldungen mit Hilfe von *Kontextdefinitionen* (s. 2.2), legen das Speicherformat der Ausgabedateien fest u.s.w. Kontextdefinitionen fungieren als Meldungsfilter und stellen den Bezug zwischen Meldung und Dateidefinition her. Sie enthalten Vergleichswerte für ausgewählte Elemente der Erhebungsbeschreibung und des Berichtszeitraumes.

Auf der Ausgabeseite gibt es *Datendateien*, in deren sätzen die Werte der Erhebungsmerkmale gespeichert werden, und optionale *Metadatendateien*. Die Anzahl der Ausgabedateien hängt u.a. von der Anzahl der Meldungen ab, sowie davon, ob die Namen der Ausgabedateien variable Bestandteile enthalten, deren Werte meldungsspezifische Metadaten sind, z.B. eine Berichtszeitraumangabe.

Eine Metadatendatei wird parallel zu einer Datendatei ausgegeben, wenn es für diese verlangt wird. Eine Daten- und die zugehörige Metadatendatei haben einen gemeinsamen statistischen Kontext. Eine Metadatendatei enthält somit die Metadaten zu den Daten genau einer korrespondierenden Datendatei. Das Format der Metadatendatei ist wie das der Datendatei frei definierbar, aber innerhalb eines Umsetzungsprozesses kann es nur eine und damit einheitliche Dateidefinition der Metadatendatei geben.

2.1. Logisches Dateikonzept

Im Sinne dieser Spezifikation sind Daten- und Metadatendatei als *logische* Dateien, zu betrachten, unabhängig von einem physikalischen Speicherort und *entsprechend jeweils genau einer Dateidefinition*. Für die Metadatendatei gilt die Einschränkung, daß es nur eine Dateidefinition geben darf, die jedoch durch Bindung an die Dateidefinitionen der Datendateien quasi multipliziert wird.

Zum besseren Verständnis ist jede Daten- bzw. Metadatendatei als separater Datenstrom vorstellbar. Aus einer Einzelmeldung fließt ein aus Daten und Metadaten bestehender *Meldungsdatenstrom* zu einem statistischen Kontext, von dort zu einer Dateidefinition und von dieser als *Resultatdatenstrom* an einen Speicherort. Es ist prinzipiell möglich, jeden Resultatdatenstrom in eine separate oder mehrere Resultatdatenströme in eine gemeinsame Datei auszugeben. In den INI-Dateien läßt sich dies durch Verwendung identischer bzw. variabler Dateinamen (s. 3.2.2) darstellen.

Eine Anwendung muß sicherstellen, daß zwei oder mehr Dateidefinitionen den selben statistischen Kontext für die Auswahl der Einzelmeldungen verwenden können. In diesem Fall fließt der Meldungsdatenstrom von einem Kontext parallel zu jeder Dateidefinition, die diesem Kontext zugeordnet ist.

Die maximale Anzahl der Resultatdatenströme ist gleich der doppelten Anzahl von Dateidefinitionen für Datendateien, da für jede Dateidefinition einer Datendatei die Ausgabe einer Metadatendatei angefordert werden kann.

2.2. INI-Dateien

INI-Dateien steuern die Verarbeitung eines DatML/RAW-Dokuments und beschreiben den Aufbau der Datendateien und der Metadatendateien. Es können zwei INI-Dateien übergeben werden. Die erste INI-Datei – *INI-Daten* – enthält unter anderem die Dateidefinitionen der Datendateien, die Beschreibungen der statistischen Kontexte und die Zuordnung der Datendateien zu den statistischen Kontexten. Die zweite INI-Datei – *INI-Meta* – enthält nur die Dateidefinition der Metadatendatei. Diese Trennung erleichtert die Verwendung zentraler Definitionen von Metadatendateien.

2.3. Datendateien

Bei der Verarbeitung eines DatML/RAW-Dokumentes fließen in eine (logische) Datendatei Daten und/oder Metadaten aus ein oder mehr Einzelmeldungen mit einem gemeinsamen statistischen Kontext. Eine Datendatei entspricht genau einer Dateidefinition in der INI-Daten und kann mit anderen Datendateien in eine gemeinsame physikalische Datei ausgegeben werden (s. 2.1).

2.3.1. Statistischer Kontext

Der statistische Kontext ist das maßgebliche Kriterium für die Zuordnung einer Einzelmeldung zu einer Dateidefinition. Er kann unter anderem Angaben zu Erhebung und Berichtszeitraum enthalten, aber auch leer sein.

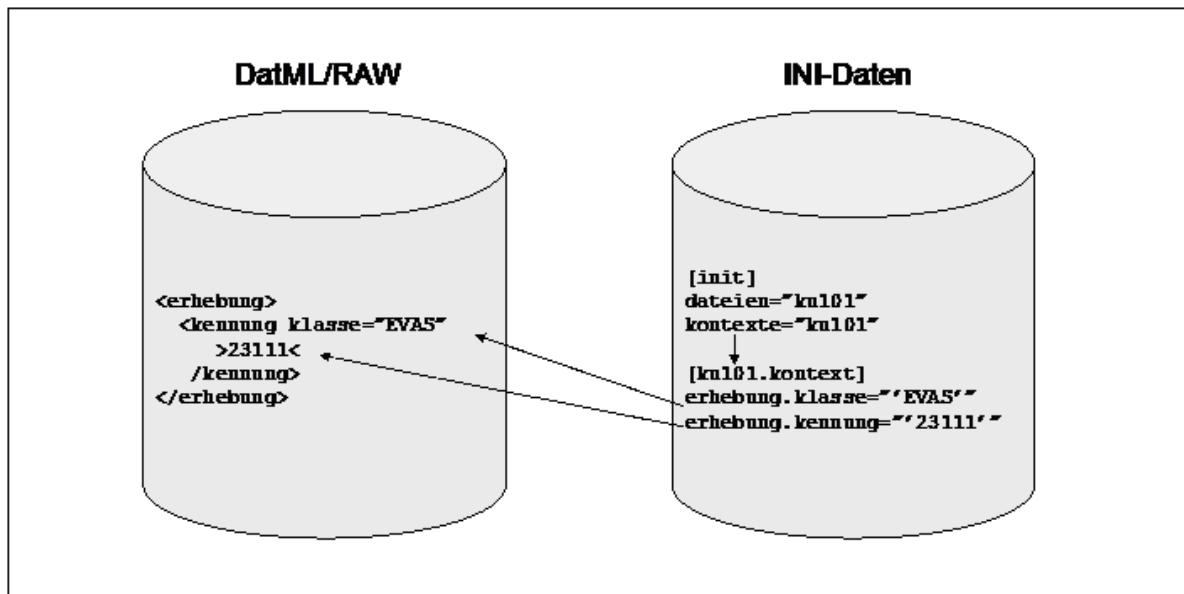
2.3.2. Zuordnung Einzelmeldung - Dateidefinition

Das Verbindungsglied zwischen Einzelmeldungen und Dateidefinitionen sind die in der INI-Daten definierten statistischen Kontexte. Für jeden definierten Kontext werden dessen Werte mit den entsprechenden Metadaten der Einzelmeldung verglichen. Stimmen alle Werte überein, wird die Einzelmeldung dem Kontext zugeordnet.

Enthält ein *definierter* Kontext keine Werte, werden im *alle* Meldungen zugeordnet.

Die Dateidefinition einer Datendatei ist über den Schlüssel `kontext` im Abschnitt `datei.ini` mit genau einem statistischen Kontext verbunden. Eine Dateidefinition ohne Bezug zu einem statistischen Kontext wird nicht ausgegeben. Ein statistischer Kontext kann von mehr als einer Dateidefinition referenziert werden.

Im Ergebnis ist eine Einzelmeldung i.d.R. mit ein- oder mehreren Kontexten und diese jeweils mit ein oder mehreren Dateidefinitionen von Datendateien verknüpft. Findet sich kein passender statistischer Kontext oder gibt es für diesen keine Dateizuordnungen, wird die Einzelmeldung nicht verarbeitet.

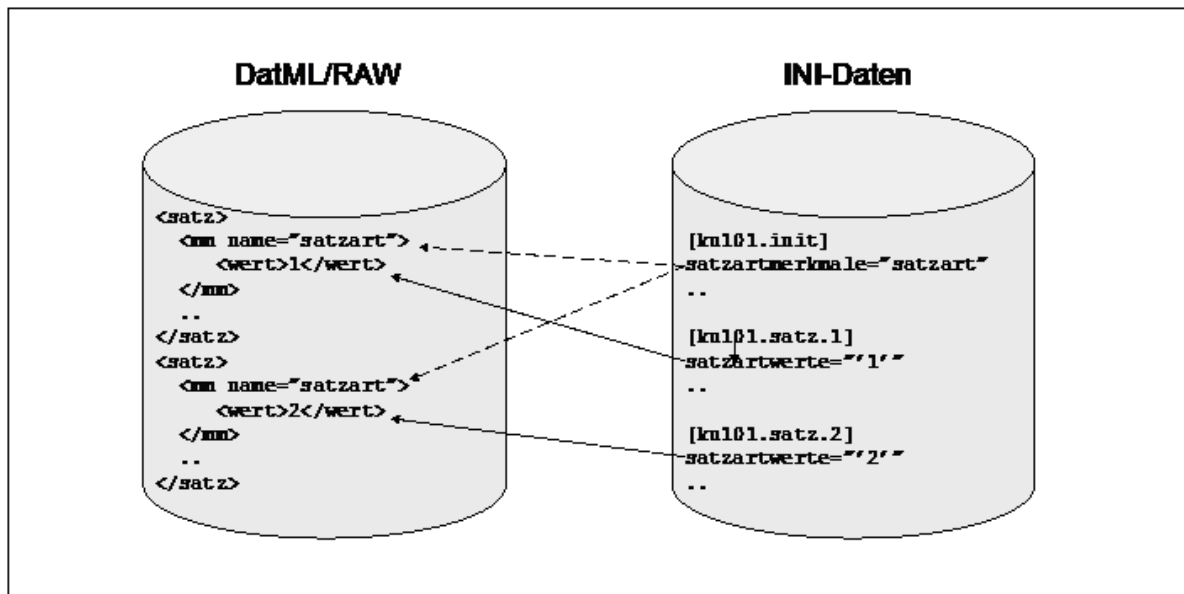


2.3.3. Zuordnung Meldungsdatensatz - Resultatdatensatz

Die Datensätze einer Datendatei (*Resultatdatensätze*) werden aus den Meldungsdatensätzen (`<satz>`) eines DatML/Raw-Dokumentes erzeugt. Ein Meldungsdatensatz kann Merkmale für beliebig viele Datensatzdefinitionen beliebig vieler Dateien liefern (s. 2.3.2). Per Default wird ein Resultatdatensatz einmal aus einem zugeordneten Meldungsdatensatz befüllt und ausgegeben. Es ist außerdem möglich, die Ausgabe eines Resultatdatensatzes an das Vorkommen von Merkmalsgruppen zu binden (s. 2.3.4.1).

Das Fehlen bzw. Vorhandensein von *Satzartangaben* steuert die Zuordnung der Meldungsdatensätze zu den Datensatzdefinitionen (`datei.satz.n`) der Datei:

- Ist für eine *Dateidefinition* kein *Satzartmerkmal* definiert, wird jeder der Datendatei zugeordnete Meldungsdatensatz jeder Datensatzdefinition dieser Dateidefinition zugeordnet.
- Ist für eine *Datensatzdefinition* kein *Satzartwert* definiert, wird jeder der Dateidefinition zugeordnete Meldungsdatensatz dieser Datensatzdefinition zugeordnet.
- Sind für eine Dateidefinition mindestens ein Satzartmerkmal und für eine Datensatzdefinition eine entsprechende Anzahl von Satzartwerten definiert, wird jeder der Dateidefinition zugeordnete Meldungsdatensatz, der der Satzartauswahl genügt, der Datensatzdefinition zugeordnet.
- Ein Meldungsdatensatz genügt einer Satzartauswahl, wenn er alle Satzartmerkmale enthält, die für eine Dateidefinition definiert sind, und die Werte dieser Merkmale gleich den Satzartwerten mindestens einer Datensatzdefinition dieser Dateidefinition sind. Für einen Meldungsdatensatz kann es mehrere passende Datensatzdefinitionen innerhalb einer Dateidefinition geben.
- Ein Meldungsdatensatz, der keiner Satzartauswahl genügt, wird allen Datensatzdefinitionen einer Dateidefinition zugeordnet, die nicht aufgrund einer Satzartauswahl ausgewählt werden, also allen Datensatzdefinitionen ohne Satzartwert.



2.3.4. Erzeugung von Datensätzen

Mit der Zuordnung eines Meldungsdatensatzes zu einer Datensatzdefinition liegt abschließend die Menge der Daten und Metadaten fest, die in den Resultatdatensatz übertragen werden kann. Standardmäßig werden für jeden zugeordneten Meldungsdatensatz genau eine Instanz des Resultatdatensatzes und für jede Einzelfelddefinition genau eine Instanz des Einzelfeldes erzeugt. Im Fall einer Strukturdefinition ist die Anzahl der erzeugten Strukturinstanzen abhängig vom Strukturtyp, der die maximale Häufigkeit festlegt und der Anzahl der Laufzeitinstanzen der zugeordneten Merkmalsgruppe (s. 3.4). Schließlich ist es möglich, aus einem Meldungsdatensatz für alle Instanzen einer Merkmalsgruppe je eine Instanz eines Resultatdatensatzes zu erzeugen.

2.3.4.1. Datensatz je Merkmalsgruppeninstanz erzeugen

Um für jede oder eine bestimmte Instanz einer Merkmalsgruppe einen separaten Resultatdatensatz auszugeben, muß diese in der Datensatzdefinition des Resultatdatensatzes mit dem Schlüssel `ausgabe` angegeben werden. Die Ausgabe unterbleibt, wenn es im aktuellen Meldungsdatensatz keine Merkmalsgruppe mit dem angegebenen Namen gibt. Wenn die Angabe der Merkmalsgruppe einen Index – direkt oder indirekt – enthält, wird der Datensatz entweder nur einmal für die Merkmalsgruppe mit dem angegebenen Index erzeugt, oder überhaupt nicht, falls diese nicht existiert. Ansonsten wird je Instanz der Merkmalsgruppe eine Instanz des Resultatdatensatzes erzeugt.

2.3.4.2. Datensatz je Merkmal erzeugen

Analog zu der Ausgabe von Datensätzen für Merkmalsgruppen ist es möglich, einen Datensatz zu erzeugen, wenn ein bestimmtes Merkmal vorhanden ist bzw. einen definierten Wert hat. Mit diesem Mechanismus lassen sich auch Satzarten erzeugen. Das Merkmal wird wie eine Merkmalsgruppe in der Datensatzdefinition mit dem Schlüssel `ausgabe` angegeben, optional mit einem Vergleichswert, der eine Konstante oder ein regulärer Ausdruck sein kann. Die Ausgabe unterbleibt, wenn es im aktu-

ellen Meldungsdatensatz das betreffende Merkmal nicht gibt oder es einen anderen als den angegebenen Wert hat. Es sind alle Merkmalstypen zulässig.

2.3.4.3. Feldgruppe je Merkmalsgruppeninstanz erzeugen

Die Instanzen von Strukturen des Typs wiederholte Feldgruppe werden auf analoge Weise wie die von Datensätzen gebildet, nämlich durch Zuordnung einer Merkmalsgruppe. Die Anzahl der Instanzen hängt in gleicher Weise von dem Vorkommen der Merkmalsgruppe und der Art des Index ab. Nicht variable wiederholte Feldgruppen werden ggf. bis zur Maximalanzahl um leere – mit Anfangswerten initialisierte – Instanzen ergänzt.

2.3.5. Speicherformat, Satzformat

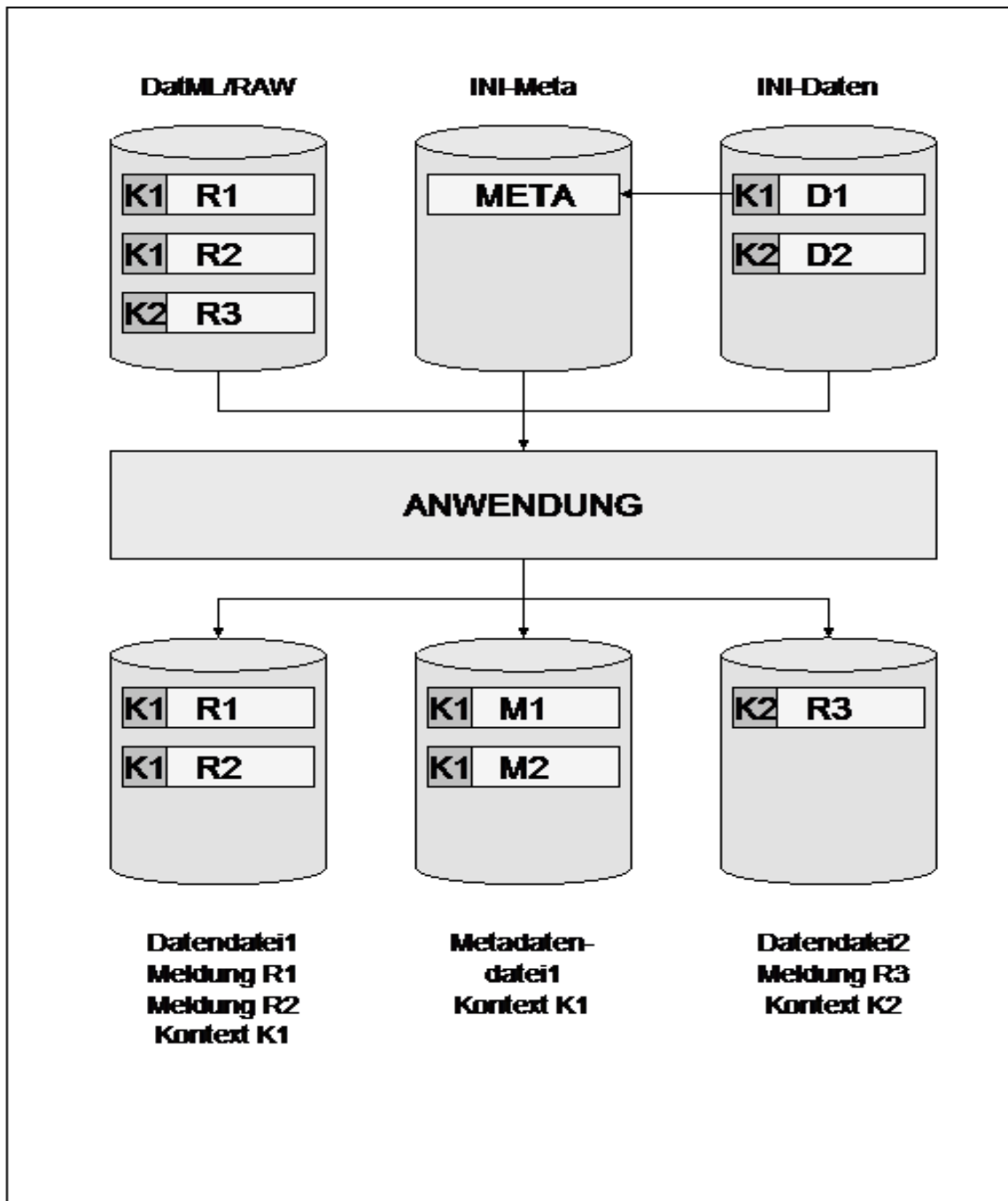
Speicher- und Satzformat einer Datendatei werden über entsprechende Schlüssel des Abschnitts `datei.init` gesteuert (s. 3.2.2). Per Default wählt die Anwendung anhand der Datensatzdefinitionen entsprechende Werte. In diesen Angaben sowie den Felddefinitionen dürfen keine formatabhängigen Felder wie Satzlängfelder berücksichtigt werden.

Achtung: Werden mehrere Datendateien in eine gemeinsame physikalische Datei ausgegeben, sollte die Anwendung die Konsistenz von dieser Werte überprüfen. Eine betriebssystemseitige Prüfung findet i.d.R. nur auf Großrechnersystemen statt.

2.4. Metadatendatei

Eine Metadatendatei wird parallel zu einer Datendatei ausgegeben, wenn es für diese verlangt wird. Eine Daten- und die zugehörige Metadatendatei haben einen gemeinsamen statistischen Kontext. Eine Metadatendatei enthält somit die Metadaten zu den Daten genau einer korrespondierenden Datendatei. Das Format der Metadatendatei ist wie das der Datendatei frei definierbar, aber innerhalb eines Umsetzungsprozesses kann es nur eine und damit einheitliche Dateidefinition der Metadatendatei geben.

2.5. Überblick (Beispiel)



Erläuterung: Ein DatML/RAW-Dokument mit den Rohdatenmeldungen R1 und R2 mit den Kontext K1 sowie R3 mit den Kontext K2 wird eingelesen. Es steht eine INI-Meta mit der Definition der Metadaten-datei und eine INI-Daten mit der Definition der Datendateien D1 mit dem Kontext K1 und D2 mit dem Kontext K2 zur Verfügung. Nur D1 fordert die Ausgabe der Metadaten-datei an. Die Anwendung erzeugt für den Kontext K1 eine Datendatei1 mit den (transformierten) Meldungsdaten der Meldungen R1 und R2 und die passende Metadaten-datei sowie für den Kontext K2 eine Datendatei2 mit den (transformierten) Meldungsdaten der Meldungen R3.

3. Aufbau der INI-Dateien

Dieses Kapitel beschreibt die Abschnitte der INI-Dateien und die Schlüssel. Für die Verwendung der Abschnitte gelten die folgenden Einschränkungen:

INI-Datei	Erlaubte Abschnitte
INI-Meta	<code>meta.init</code> , <code>meta.kopf</code> , <code>meta.satz.n</code>
INI-Daten	<code>init</code> , <code>datei.init</code> , <code>datei.kopf</code> , <code>datei.satz.n</code> , <code>kontext.kontext</code>

HINWEIS: Die Reihenfolge der Schlüssel in den Abschnitten der INI-Dateien ist grundsätzlich beliebig. Aus Gründen der Einheitlichkeit wird empfohlen, die in den Beschreibungen verwendete Reihenfolge der Abschnitte und Schlüssel beizubehalten; insbesondere sollten alle Felddefinitionen ununterbrochen und mit aufsteigendem Schlüsselwert einander folgen, und die Anzahl der Felddefinitionen muß vor der ersten Felddefinition mit dem Schlüssel `anzahlFelder` deklariert werden.

Für einige optionale Schlüssel gibt es Defaultwerte; diese sind durch Unterstreichung gekennzeichnet.

Bei allen konstanten Namen, z.B. Abschnittsnamen, Schlüssel und vordefinierte Werte, ist die Groß- und Kleinschreibung zu beachten:

Beispiel:	Zulässig	Unzulässig
<code>[meta.init]</code>	x	
<code>[META.INIT]</code>		x
<code>open="extend"</code>	x	
<code>Open="extend"</code>		x
<code>open="Extend"</code>		x

3.1. INI-Meta

Die INI-Meta enthält die Dateidefinition der Metadatenfile.

Abschnitt	#	Beschreibung
<code>meta.init</code>	1	Initialisierungswerte
<code>meta.kopf</code>	0-1	Definiert den Kopfdatensatz der Metadatenfile
<code>meta.satz.n</code>	1-n	Definiert einen Datensatz der Metadatenfile

Der Kopfdatensatz ist optional. Ist er vorhanden, wird er jedem Datensatz vorangestellt. Es sind genau ein Initialisierungsabschnitt und mindestens eine Datensatzdefinition notwendig.

WICHTIG: Für die (Kopf-)Datensatzdefinitionen der INI-Meta gilt folgende Einschränkung:

- Einzelfelder der Metadatenfile dürfen *ausschließlich* mit Metadaten (`#name`) und Konstanten (`'konstante'`) gefüllt werden. Einzelfelddefinitionen in der INI-Meta dürfen daher kein Merkmal referenzieren.

3.1.1. Abschnitt `meta.init`

Der Abschnitt `meta.init` liefert Initialisierungswerte für die Beschreibung der Metadatenfile.

Schlüssel	#	Beschreibung
<code>anzahlSatzdefinitionen</code>	1	Anzahl der Datensatzdefinitionen <code>meta.satz.n</code> ; für <code>n</code> gilt: $1 \leq n \leq 256$
<code>satzformat</code>	0-1	<code>default f fest v variabel</code> Legt das Satzformat der Datei fest. Per Default wird dieses von der Anwendung bestimmt: festes Satzformat wird gewählt, wenn es nur eine Datensatzdefinition für eine Datei gibt oder alle die gleiche Satzlänge haben, ansonsten ist das Ergebnis variables Satzformat.
<code>speicherformat</code>	0-1	<code>default ebcdic ascii ascii-csv ascii-binaer</code> Bestimmt das externe Speicherformat der Datei analog zu den Möglichkeiten in STATSPEZ. Das Satzendezeichen und betriebssystemspezifische Felder werden automatisch generiert.
<code>open</code>	0-1	<code>output extend[s rotationsdauer]</code> <code>open="extend"</code> bewirkt, daß eine vorhandene Datei erweitert und nicht überschrieben wird, d.h. die Ausgabedatensätze werden an das Ende der Datei angehängt. Der optionale Wert <code>rotationsdauer</code> ermöglicht die Angabe einer Zeitraumes, nach dessen Ablauf die aktuelle Ausgabedatei rotiert, d.h. durch eine neue Instanz ersetzt wird. Dies ist vor allem in

		Implementierungen sinnvoll, die als Server laufen. <i>rotationsdauer</i> hat die Form: <i>nz</i> , wobei <i>n</i> ein ganzzahliger Wert und <i>z</i> die zu messende Zeiteinheit mit einem der folgenden Werte ist: <i>d</i> (Tage), <i>h</i> (Stunden), <i>m</i> (Minuten) und <i>s</i> (Sekunden). Beispiele: <pre>open="extend 1d" : ein Tag open="extend 1h" : eine Stunde open="extend 90m" : 90 Minuten</pre> -
<i>feldtrennzeichen</i>	0-1	Default: <code>' ; '</code> (Semikolon). Definiert das Zeichen, das bei der Ausgabe im Speicherformat <i>ascii-csv</i> als Feldtrennzeichen verwendet wird. Ist der angegebene String länger als ein Zeichen, wird nur das erste Zeichen verwendet.
<i>dezimalzeichen</i>	0-1	<code>._ ,</code> Definiert das Zeichen, das bei der Ausgabe numerischer Felder bei Bedarf als Dezimalzeichen verwendet wird. Ist der angegebene String länger als ein Zeichen, wird nur das erste Zeichen verwendet.

Beispiel:

```
[meta.init]
anzahlSatzdefinitionen="6"
```

3.1.2. Abschnitt `meta.kopf`

Ein Abschnitt mit Namen `meta.kopf` definiert den Kopfdatensatz der Metadatendatei. Der Kopfdatensatz wird jedem Datensatz vorangestellt und vor der Ausgabe neu gefüllt. Der Anwender muß darauf achten, nur solche Felder in den Kopfdatensatz zu übernehmen, die sinnvoll versorgt werden können. Wird z.B. die Satzart nicht aus den Daten entnommen, sondern per Konstante eingesetzt, ist die Definition des Feldes nur in den jeweiligen Datensatzdefinitionen sinnvoll möglich.

Schlüssel	#	Beschreibung
<code>anzahlFelder</code>	1	Anzahl der Felddefinitionen <code>fn</code>
<code>fn</code>	1-n	<code>einzelFelddefinition strukturdefinition</code> , s. 3.3 bzw. 3.4; für <code>n</code> gilt: $1 \leq n \leq \text{anzahlFelder}$

Der Schlüssel `anzahlFelder` muß den Felddefinitionen vorausgehen.

Beispiel:

```
[meta.kopf]
anzahlFelder="4"
f1="EF1 (ALN 5) #erhebung.kennung"
f2="EF2 (ALN 10)"
```

```
f3="EF3UG1 (ALN 8) #protokoll.eingang.datum"
f4="EF3UG2 (ALN 6) #protokoll.eingang.zeit"
```

3.1.3. Abschnitt `meta.satz.n`

Ein Abschnitt mit Namen `meta.satz.n` definiert einen Datensatz der Metadatendatei. Für n gilt: $1 \leq n \leq \text{anzahlSatzdefinitionen}$ aus dem Abschnitt `meta.init`.

WICHTIG: In den Datensatzdefinitionen dürfen keine formatabhängigen Felder wie Satzlängfelder berücksichtigt werden.

Schlüssel	#	Beschreibung
<code>ausgabe</code>	1	<code>open-doc close-doc close-rep</code> Verarbeitungszeitpunkt, zu dem der Satz ausgegeben wird.
<code>datensatzname</code>	0-1	Name des Datensatzes für dokumentarische Zwecke, z.B. bei der Ausgabe von Fehlertexten.
<code>anzahlFelder</code>	1	Anzahl der Felddefinitionen f_n
f_n	1-n	<code>einzelfelddefinition strukturdefinition</code> , s. 3.3 bzw. 3.4; für n gilt: $1 \leq n \leq \text{anzahlFelder}$

Neben der Anzahl der Felddefinitionen muß mit dem Schlüssel `ausgabe` spezifiziert werden, zu welchem Verarbeitungszeitpunkt der Satz ausgegeben werden soll:

Wert von <code>ausgabe</code>	Beschreibung
<code>open-doc</code>	Vor Verarbeiten der ersten Meldung, aber nach dem Einlesen der meldungsübergreifenden Metadaten. Dies entspricht dem Erreichen des ersten Elementes vom Typ <code><nachricht></code> .
<code>close-doc</code>	Beim Schließen des DatML/RAW-Dokuments.
<code>close-rep</code>	Nach Verarbeiten einer Meldung. Dies entspricht dem Erreichen eines Elementes vom Typ <code></segment></code> bzw. <code></nachricht></code> .

Beispiel:

```
[meta.satz.1]
ausgabe="open-doc"
datensatzname="Datensatz-Datum-Uhrzeit"
anzahlFelder="3"
f1="EF1 (ALN 1) '1'"
f2="EF2 (ALN 8) #datum"
f3="EF3 (ALN 6) #zeit"

[meta.satz.2]
ausgabe="close-rep"
datensatzname="Datensatz-Meldungszähler"
anzahlFelder="5"
f1="EF1 (ALN 1) '2'"
f2="EF2 (ALN 4) #berichtszeitraum.jahr"
f3="EF3 (ALN 12) #erhebung.kennung"
```

```
f4="EF4 (ALN 60) #anzFehlerhafteSaetzeMeldung"  
f5="EF5 (ALN 60) #anzFehlerhafteSaetzeMeldung"  
  
[meta.satz.3]  
ausgabe="close-doc"  
datensatzname="Datensatz-Insgesamtzaehler"  
anzahlFelder="3"  
f1="EF1 (ALN 1) '3'"  
f2="EF2 (ALN 60) #anzFehlerhafteSaetzeInsgesamt"  
f3="EF3 (ALN 60) #anzFehlerhafteSaetzeInsgesamt"
```

3.2. INI-Daten

INI-Daten enthält ein oder mehrere Dateidefinitionen und statistische Kontexte. Eine Dateidefinition besteht aus genau einem Abschnitt *datei.init*, mindestens einem Abschnitt *datei.satz* und einem optionalen Abschnitt *datei.kopf*.

Abschnitt	#	Beschreibung
<i>init</i>	0-1	Initialisierungswerte, z.B. Namen der Kontexte und Dateien
<i>kontext.kontext</i>	1-n	Definiert einen statistischen Kontext
<i>datei.init</i>	1-n	Initialisierungswerte für die Beschreibung einer Datendatei, z.B. Zuordnung zu einem statistischen Kontext, Satzartmerkmale
<i>datei.kopf</i>	0-n	Definiert einen Datensatz einer Datendatei <i>datei</i> und ggf. die Satzartwerte (Merkmalswerte), mit denen Meldungsdatensätze einer Datensatzdefinition zugeordnet werden.
<i>datei.satz.n</i>	1-n	Definiert einen Datensatz einer Datendatei <i>datei</i> und ggf. die Satzartwerte (Merkmalswerte), mit denen Meldungsdatensätze einer Datensatzdefinition zugeordnet werden.

3.2.1. Abschnitt *init*

Der Abschnitt *init* definiert die Namen und damit implizit die Anzahl der Datei- und Kontextdefinitionen.

Schlüssel	#	Beschreibung
<i>dateien</i>	1	Eine Liste von Name(n) <i>datei</i> der Dateidefinitionen <i>datei.init</i> ; mehrere Namen müssen durch Leerzeichen oder Tabs getrennt werden, und es muß mindestens ein Name vorhanden sein.
<i>externerNameDaten</i>	0-1	Template für den externen Namen der Datendatei (s. 3.12). Der Name kann eine absolute oder relative Pfadangabe enthalten.
<i>externerNameMeta</i>	0-1	Template für den externen Namen der Metadatei (s. 3.12). Der Name kann eine absolute oder relative Pfadangabe enthalten.
<i>kontexte</i>	1	Eine Liste von Name(n) <i>kontext</i> der Kontextdefinitionen <i>kontext.kontext</i> ; mehrere Namen müssen durch Leerzeichen oder Tabs getrennt werden, und es muß mindestens ein Name vorhanden sein.
<i>open</i>	0-1	<i>output</i> { <i>extend</i> [<i>rotationsdauer</i>]} <i>open</i> ="extend" bewirkt, daß eine vorhandene Datei erweitert und nicht überschrieben wird, d.h. die Ausgabedatensätze werden an das Ende der Datei angehängt. Der optionale Wert <i>rotationsdauer</i> ermöglicht die Angabe eines Zeitraumes, nach dessen Ablauf die aktuelle Ausgabedatei rotiert, d.h. durch eine neue Instanz ersetzt wird. Dies ist vor allem in

		Implementierungen sinnvoll, die als Server laufen. <i>rotationsdauer</i> hat die Form: <i>nz</i> , wobei <i>n</i> ein ganzzahliger Wert und <i>z</i> die zu messende Zeiteinheit mit einem der folgenden Werte ist: <i>d</i> (Tage), <i>h</i> (Stunden), <i>m</i> (Minuten) und <i>s</i> (Sekunden). Beispiele: <pre>open="extend 1d" : ein Tag open="extend 1h": eine Stunde open="extend 90m" : 90 Minuten</pre>
<i>satzformat</i>	0-1	<i>default</i> <i>f</i> <i>fest</i> <i>v</i> <i>variabel</i> Legt das Satzformat der Datei fest. Per Default wird dieses von der Anwendung bestimmt: festes Satzformat wird gewählt, wenn es nur eine Datensatzdefinition für eine Datei gibt oder alle die gleiche Satzlänge haben, ansonsten ist das Ergebnis variables Satzformat.
<i>speicherformat</i>	0-1	<i>default</i> <i>ebcdic</i> <i>ascii</i> <i>ascii-csv</i> <i>ascii-binaer</i> Bestimmt das externe Speicherformat der Datei analog zu den Möglichkeiten in STATSPEZ. Das Satzendezeichen und betriebssystemspezifische Felder werden automatisch generiert.
<i>feldtrennzeichen</i>	0-1	Default: <i>;</i> (Semikolon). Definiert das Zeichen, das bei der Ausgabe im Speicherformat <i>ascii-csv</i> als Feldtrennzeichen verwendet wird. Ist der angegebene String länger als ein Zeichen, wird nur das erste Zeichen verwendet.
<i>dezimalzeichen</i>	0-1	Default: <i>,</i> (Komma). Definiert das Zeichen, das bei der Ausgabe numerischer Felder bei Bedarf als Dezimalzeichen verwendet wird. Ist der angegebene String länger als ein Zeichen, wird nur das erste Zeichen verwendet.

Der Abschnitt *init* ist optional. Wenn er fehlt, wird für die Schlüssel *dateien* und *kontexte* der Defaultwert "*datei*" wirksam, d.h. ohne diesen Abschnitt kann es genau eine Kontextdefinition und eine Dateidefinition (mit beliebig vielen Datensatzdefinitionen) geben, deren Abschnittsnamen alle mit "*datei*" beginnen.

Die Schlüssel *externerNameDaten*, *externerNameMeta*, *open*, *satzformat*, *speicherformat*, *feldtrennzeichen* und *dezimalzeichen* sind an dieser Stelle globale Schlüssel. Sie können auch lokal in den Initialisierungsabschnitten *datei.init* der Dateidefinitionen angegeben werden. Lokale Schlüssel haben Vorrang vor globalen Schlüssel.

Beispiel *mit* [*init*]:

```
[init]
```

```

dateien="ku101 ku102"
kontexte="khstat"

[khstat.kontext]
erhebung.kennung="'23111'"
erhebung.text="'Grunddaten der Krankenhäuser'"
...

[ku101.init]
kontext="khstat"
anzahlSatzdefinitionen="2"
...

[ku101.satz.1]
...

[ku101.satz.2]
...

[ku102.init]
kontext="khstat"
anzahlSatzdefinitionen="1"
...

[ku102.satz.1]
...

```

Beispiel *ohne* [init]:

```

[datei.init]
kontext="datei"
anzahlSatzdefinitionen="1"
...

[datei.satz.1]
...

[datei.kontext]
erhebung.kennung="'23111'"
erhebung.text="'Grunddaten der Krankenhäuser'"
...

```

3.2.2. Abschnitt *datei.init*

Ein Abschnitt mit Namen *datei.init* definiert die Zuordnung der Dateidefinition mit dem Namen *datei* zu einem Kontext und damit allen DatML/RAW-Einzelmeldungen, die die Kriterien dieses Kontextes erfüllen. Außerdem definiert er Merkmale für die Satzartauswahl aus den Meldungsdatensätzen und Templates für die Bildung externer Dateinamen.

Schlüssel	#	Beschreibung
<i>kontext</i>	0-1	Name <i>kontext</i> einer Kontextdefinition; der Name muß einer der Kontextnamen sein, die im Abschnitt <i>init</i> mit dem Schlüssel <i>kontexte</i> definiert

		wurden, und es muß ein entsprechender Abschnitt <code>kontext.kontext</code> existieren. Ein leerer String darf verwendet werden, um Dateidefinitionen „abzuhängen“.
<code>anzahlSatzdefinitionen</code>	1	Anzahl der Datensatzdefinitionen <code>datei.satz.n</code> ; für n gilt: $1 \leq n \leq 256$
<code>externerNameDaten</code>	0-1	Template für den externen Namen der Datendatei (s. 3.12). Der Name kann eine absolute oder relative Pfadangabe enthalten.
<code>externerNameMeta</code>	0-1	Template für den externen Namen der Metadatendatei (s. 3.12). Der Name kann eine absolute oder relative Pfadangabe enthalten.
<code>satzartmerkmale</code>	0-1	Namen der Merkmale, die für die Satzartauswahl aus den Meldungsdatensätzen verwendet werden; mehrere Namen müssen durch Leerzeichen oder Tabs getrennt werden, ein Leerstring bedeutet, daß keine Satzartauswahl stattfindet (jeder Meldungsdatensatz wird allen Datensatzdefinitionen der Datei zugeordnet). HINWEIS: Anstelle eines Merkmals kann auch ein in Frage kommendes Metadatum angegeben werden, z.B. <code>#satz.kennung</code> .
<code>satzformat</code>	0-1	<code>default f fest v variabel</code> Legt das Satzformat der Datei fest. Per Default wird dieses von der Anwendung bestimmt: festes Satzformat wird gewählt, wenn es nur eine Datensatzdefinition für eine Datei gibt oder alle die gleiche Satzlänge haben, ansonsten ist das Ergebnis variables Satzformat.
<code>speicherformat</code>	0-1	<code>default ebcdic ascii ascii-csv ascii-binaer</code> Bestimmt das externe Speicherformat der Datei analog zu den Möglichkeiten in STATSPEZ. Das Satzendezeichen und betriebssystemspezifische Felder werden automatisch generiert.
<code>metadaten</code>	0-1	<code>default ja nein</code> Legt fest, ob eine Metadatendatei ausgegeben werden soll. <code>default</code> bedeutet, daß die Ausgabe erfolgt, wenn eine Metadatendatei per INI-Meta definiert wurde. <code>ja</code> bedeutet, daß die Ausgabe unbedingt erfolgen soll, in diesem Fall ist das Fehlen der INI-Meta ein Fehler.
<code>open</code>	0-1	<code>output extend[s rotationsdauer]</code> <code>open="extend"</code> bewirkt, daß eine vorhandene Datei erweitert und nicht überschrieben wird, d.h. die Ausgabedatensätze werden an das Ende der Datei angehängt. Der optionale Wert <code>rotationsdauer</code> ermöglicht die Angabe einer Zeitraumes, nach dessen Ablauf die aktuelle Ausgabedatei rotiert, d.h. durch eine

		<p>neue Instanz ersetzt wird. Dies ist vor allem in Implementierungen sinnvoll, die als Server laufen. <code>rotationsdauer</code> hat die Form: <code>nz</code>, wobei <code>n</code> ein ganzzahliger Wert und <code>z</code> die zu messende Zeiteinheit mit einem der folgenden Werte ist: <code>d</code> (Tage), <code>h</code> (Stunden), <code>m</code> (Minuten) und <code>s</code> (Sekunden). Beispiele:</p> <pre>open="extend 1d" : ein Tag open="extend 1h" : eine Stunde open="extend 90m" : 90 Minuten</pre>
<code>feldtrennzeichen</code>	0-1	<p>Default: <code>;</code> (Semikolon). Definiert das Zeichen, das bei der Ausgabe im Speicherformat <code>ascii-csv</code> als Feldtrennzeichen verwendet wird. Ist der angegebene String länger als ein Zeichen, wird nur das erste Zeichen verwendet.</p>
<code>dezimalzeichen</code>	0-1	<p><code>._ ,</code> Definiert das Zeichen, das bei der Ausgabe numerischer Felder bei Bedarf als Dezimalzeichen verwendet wird. Ist der angegebene String länger als ein Zeichen, wird nur das erste Zeichen verwendet.</p>

Beispiel:

```
[kul01.init]
kontext="khstat"
anzahlSatzdefinitionen="5"
externerNameDaten="kul01.daten.{#datum}{#zeit}"
externerNameMeta="kul01.meta.{#datum}{#zeit}"
satzartmerkmale="satzart"
speicherformat="ascii-csv"
feldtrennzeichen="/"
dezimalzeichen="."
```

Die Schlüssel `externerNameDaten`, `externerNameMeta`, `open`, `satzformat`, `speicherformat`, `feldtrennzeichen` und `dezimalzeichen` sind an dieser Stelle lokale Schlüssel der Dateidefinition. Sie können auch global im Abschnitt `init` angegeben werden. Lokale Schlüssel haben Vorrang vor globalen Schlüssel.

3.2.3. Abschnitt `datei.kopf`

Ein Abschnitt mit Namen `datei.kopf` definiert den Kopfdatensatz einer Datendatei. Die Felder des Kopfdatensatzes werden den Feldern der Datensätze vorangestellt und vor der Ausgabe neu gefüllt. Der Anwender muß darauf achten, nur solche Felder in die Definition des Kopfdatensatzes zu übernehmen, die dort sinnvoll versorgt werden können, also nicht satzartabhängig sind. Wird z.B. die Satzart nicht aus den Daten entnommen, sondern per Konstante eingesetzt, ist die Definition des Feldes nur in den jeweiligen Datensatzdefinitionen sinnvoll möglich.

Schlüssel	#	Beschreibung
<code>anzahlFelder</code>	1	Anzahl der Felddefinitionen <code>fn</code>
<code>fn</code>	1-n	<code>einzel Felddefinition</code> <code>strukturdefinition</code> , s. 3.3 bzw. 3.4; für <code>n</code> gilt: $1 \leq n \leq \text{anzahlFelder}$

Der Schlüssel `anzahlFelder` muß den Felddefinitionen vorausgehen.

Beispiel:

```
[ku101.kopf]
anzahlFelder="3"
f1="EF1 (ALN 5) krankenhaus-nr"
f2="EF2 (NOV 4 K 0) #berichtszeitraum.jahr"
f3="EF3 (ALN 5) #erhebung.kennung"
```

3.2.4. Abschnitt *datei.satz.n*

Dieser Abschnitt definiert einen Datensatz einer Datendatei `datei` und ggf. die Satzartwerte (Merkmalswerte), mit denen Meldungsdatensätze einer Datensatzdefinition zugeordnet werden. `n` ist eine ganze positive Zahl ≥ 1 .

WICHTIG: Bei Angabe des Schlüssels `ausgabe="merkmalsgruppe::mgr_pfad"` erfolgt die Adressierung der Merkmale und Merkmalsgruppen nicht relativ zum Meldungsdatensatz, sondern relativ zu `mgr_pfad`, analog zu den Regeln für die Adressierung innerhalb von Strukturdefinitionen (3.4.2).

WICHTIG: Datensatzdefinitionen dürfen keine betriebssystemspezifischen Felder enthalten.

Schlüssel	#	Beschreibung
<code>satzartwerte</code>	0-1	Werte der Satzartmerkmale; mehrere Werte müssen durch Leerzeichen oder Tabs getrennt werden.
<code>datensatzname</code>	0-1	Name des Datensatzes für dokumentarische Zwecke, z.B. bei der Ausgabe von Fehlertexten.
<code>ausgabe</code>	0-1	<code>meldungsdatensatz</code> <code>merkmalsgruppe::mgr_pfad</code> <code>merkmal::mm_pfad[={konstante RE}]</code> Legt analog zum gleichnamigen Schlüssel im Abschnitt <code>meta.satz.n</code> fest, welches Ereignis die Ausgabe des Datensatzes bewirkt: Entweder eine (bestimmte) Instanz einer Merkmalsgruppe oder ein Merkmal (mit einem bestimmten Wert) oder der aktuelle Meldungsdatensatz (Voreinstellung).
<code>anzahlFelder</code>	1	Anzahl der Felddefinitionen <code>fn</code>
<code>fn</code>	1-n	<code>einzel Felddefinition</code> <code>strukturdefinition</code> , s. 3.3 bzw. 3.4; für <code>n</code> gilt: $1 \leq n \leq \text{anzahlFelder}$

Der Schlüssel `anzahlFelder` muß den Felddefinitionen vorausgehen.

Beispiel, Ausgabe je Meldungsdatensatz:

```
[ku101.satz.1]
satzartwerte="1"
anzahlFelder="10"
f1="EF1 (ALN 10)"
f2="EF1UG1 (ALN 8) #protokoll.eingang.datum"
f3="EF1UG2 (ALN 6) #protokoll.eingang.zeit"
USW.
```

Beispiele, Ausgabe je Instanz der Merkmalsgruppe `umsatz`; der Definitionskontext wird durch den Schlüssel `ausgabe` auf die Merkmalsgruppe `umsatz` gesetzt, die Adressierung der Merkmale erfolgt daher relativ zu dieser Merkmalsgruppe:

```
[ku101.satz.1]
satzartwerte="1"
ausgabe="merkmalsgruppe::umsatz[]"
anzahlFelder="5"
f1="BETRIEB (NOV 10 K 0) ../betriebsnummer"
f2="MONAT (NOV 2 K 0) monat"
f3="BRUTTO (NOV 10 K 2) brutto"
f4="MENGE (NOV 10 K 2) menge"
f5="EINHEIT (ALN 2) einheit"
```

Beispiel für den Schlüssel `ausgabe` mit Merkmalen:

Wenn das Hilfsmerkmal `verkehrsrichtung` den Wert 1 hat:

```
ausgabe="merkmal::verkehrsrichtung='1'"
```

Wenn das Merkmal `geschlecht` den Wert M hat:

```
ausgabe="merkmal::geschlecht='M'"
```

3.2.5. Abschnitt *kontext.kontext*

Dieser Abschnitt definiert einen Kontext. Der Kontext legt Auswahlkriterien für die Zuordnung von Einzelmeldungen zu der Dateidefinition einer Datendatei (und der mit ihr optional verbundenen Metadatei) fest. Ein Kontext wird aus `datei.init` referenziert und kann daher für mehr als eine Datendatei `datei` gelten.

Ein Auswahlkriterium wird durch einen Schlüssel und einen Vergleichswert definiert; der Schlüssel referenziert ein Metadatum, dessen Wert auf Übereinstimmung mit dem Vergleichswert geprüft wird. Die Auswertung eines Kontextes bewirkt nur dann die Zuordnung zu einer Einzelmeldung, wenn die Auswertung aller angegebenen Auswahlkriterien eine Übereinstimmung mit den Metadaten der Einzelmeldung ergibt.

Folgende Schlüssel sind ab DatML/RAW 1.0 zulässig:

Schlüssel	#	Typ; Beschreibung
<code>erhebung.kennung</code>	0-1	<i>konstante</i> <i>RE</i> ; Kennung der Erhebungsdefinition
<code>erhebung.klasse</code>	0-1	<i>konstante</i> <i>RE</i> ; Klasse der Erhebungsdefinition
<code>erhebung.text</code>	0-1	<i>konstante</i> <i>RE</i> ; Text der Erhebungsdefinition

<code>berichtszeitraum.bzr</code>	0-1	<i>konstante</i> <i>RE</i> ; Aggregierter Berichtszeitraum, s. 6.3.3
<code>berichtszeitraum.datum</code>	0-1	<i>konstante</i> <i>RE</i> ; Aggregierter Berichtszeitraum, s. 6.3.3
<code>berichtszeitraum.halbjahr</code>	0-1	<i>konstante</i> <i>RE</i> ; Normalisiertes Berichtshalbjahr <i>h</i>
<code>berichtszeitraum.jahr</code>	0-1	<i>konstante</i> <i>RE</i> ; Normalisiertes Berichtsjahr <i>yyyy</i>
<code>berichtszeitraum.monat</code>	0-1	<i>konstante</i> <i>RE</i> ; Normalisierter Berichtsmonat <i>mm</i>
<code>berichtszeitraum.quartal</code>	0-1	<i>konstante</i> <i>RE</i> ; Normalisiertes Berichtsquartal <i>q</i>
<code>berichtszeitraum.semester</code>	0-1	<i>konstante</i> <i>RE</i> ; Normalisiertes Berichtssemester <i>s</i>
<code>berichtszeitraum.spanne</code>	0-1	<i>konstante</i> ; Berichtszeitraumspanne, s. 3.2.5.1
<code>berichtszeitraum.string</code>	0-1	<i>konstante</i> <i>RE</i> ; Aggregierter Berichtszeitraum, s. 6.3.3
<code>berichtszeitraum.tag</code>	0-1	<i>konstante</i> <i>RE</i> ; Normalisierter Berichtstag <i>dd</i>
<code>berichtszeitraum.woche</code>	0-1	<i>konstante</i> <i>RE</i> ; Normalisierte Berichtswoche <i>ww</i>

Der Vergleich findet bei Konstanten (*konstante*) stets *ohne* und bei regulären Ausdrücken (*RE*) in der Regel *mit* Berücksichtigung der Groß- und Kleinschreibung statt, soweit nicht für einem regulären Ausdruck die Unterscheidung aufgehoben ist.

Der Schlüssel `berichtszeitraum.spanne` verfügt für die Abfrage von Berichtszeitraumspannen über einen erweiterten Wertebereich (3.2.5.1).

Beispiele (Konstanten):

```
erhebung.kennung="'23111'"
erhebung.klasse="'EVAS'"
erhebung.text="'Grunddaten der Krankenhäuser'"
berichtszeitraum.quartal="1"
berichtszeitraum.string="2002Q4"
```

Beispiele (Reguläre Ausdrücke):

```
erhebung.kennung="^23111" (alle mit 23111 beginnenden)
berichtszeitraum.jahr="200\d" (alle von 2000 - 2009)
berichtszeitraum.jahr="200[3-6]" (alle von 2003 - 2006)
```

Konstanten, die Metazeichen regulärer Ausdrücke enthalten, müssen in einfachen Hochkommata eingeschlossen sein. Weitere Beispiele für reguläre Ausdrücke s. Abschnitt 3.10.2.

3.2.5.1. Spannen in Berichtszeiträumen

Spannen in Berichtszeiträumen können auf unterschiedliche Weise geprüft werden. Zu unterscheiden sind *einfache* und *komplexe* Spannen:

- Eine *einfache Spanne* liegt vor, wenn nur eine Zeiteinheit abgefragt wird oder im Fall einer Abfrage mehrerer Zeiteinheiten zwischen diesen keine Abhängigkeiten bestehen, so daß sie separat geprüft werden können; Beispiel: Jeweils alle Quartale der Jahre 2002 bis 2005.
- Eine *komplexe Spanne* umfaßt mindestens zwei Zeiteinheiten, zwischen denen eine Abhängigkeit existiert; Beispiel: Juli 2002 bis Juni 2005; für 2002 sind nur die Monate ab Juli, für 2005 bis Juni, und für die dazwischenliegenden Jahre alle Monate zulässig; die Spannen für die Zeiteinheiten Jahr und Monat können daher nicht unabhängig voneinander geprüft werden.

Im Fall einfacher Spannen reichen reguläre Ausdrücke zur Abfrage:

```
berichtszeitraum.jahr="200[2-5]" (alle Jahre 2002 - 2005)
berichtszeitraum.quartal="[1-4]" (alle Quartale 1 - 4)
```

Der Schlüssel `berichtszeitraum.string` verfügt für die Abfrage komplexer Spannen über einen erweiterten Wertebereich *bzrspanne*, der die Angabe von Spannen berücksichtigt:

```
bzrspanne : bzrstring-bzrstring|bzrstring-|-bzrstring
bzrstring : wie Metadatum berichtszeitraum.string (6.3.3).
```

Die Definition einer Spanne besteht aus zwei Berichtszeitraumangaben, deren Format dem des Metadatums `berichtszeitraum.string` entspricht, verbunden mit einem Minuszeichen. Eine der Berichtszeitraumangaben darf alternativ fehlen, und wird dann so berechnet, daß jeder Berichtszeitraum paßt, der die gleichen Zeiteinheiten verwendet und keinen Konflikt mit der vorhandenen Berichtszeitraumangabe auslöst.

Beispiele:

Spanne	Meldung	Treffer
2002Q1-2005Q4	Jahr 2002, Quartal 1	ja
	Jahr 2002, kein Quartal	nein
	Jahr 2001, Quartal 1	nein
	Jahr 2007, Quartal 1	nein
2002Q1-	Jahr 2002, Monat 1	ja
	Jahr 2002, kein Monat	nein
	Jahr 2001, Monat 12	nein
	Jahr 2007, Monat 1	ja
-2005Q4	Jahr 2002, Quartal 1	ja
	Jahr 2002, kein Quartal	nein
	Jahr 2001, Quartal 1	ja
	Jahr 2007, Quartal 1	nein

3.3. Einzelfelder

Jede Datensatzdefinition ist ein separater Abschnitt der INI-Datei und besteht aus einer Folge von Einzelfeld- und/oder Strukturdefinitionen. Eine Einzelfelddefinition besteht aus einem optionalen Einzelfeldnamen, einer Datentypdeklaration in SPLV-Notation, einer optionalen Angabe der Quelle, aus der das Einzelfeld gefüllt wird, und einem optionalen Default, der verwendet wird, wenn die Auswertung der Quelle keinen Wert ergibt. Der Default darf nur zusammen mit der Quelle angegeben werden, das Pipe-Symbol (|) trennt Quelle und Default. Liefern weder die Quelle noch der Default einen Wert, wird das Einzelfeld mit typgerechten Anfangswerten initialisiert.

3.3.1. Einzelfelddefinition

Eine Einzelfelddefinition *einzelfeld* besteht mindestens aus einer Typdeklaration *splv-typ* und hat folgende Syntax:

```
einzelfeld : [s][feldname s](splv-typ)[s[quelle[s]['|'[s]default]]]
```

3.3.1.1. Operand *feldname*

Operand	Beschreibung
<i>feldname</i>	<i>splv-name</i> ; Der Name des Einzelfeldes.
<i>splv-name</i>	Ein Name gemäß den Regeln für SPLV-Namen lt. SPLV-Sprachbeschreibung.

feldname ist optional, da der Feldname für die Erzeugung eines Einzelfeldes nicht notwendig ist; es wird aber empfohlen, *feldname* aus dokumentarischen Gründen und für die Verwendung in Fehlermeldungen anzugeben.

3.3.1.2. Operand *splv-typ*

Operand	Beschreibung
<i>splv-typ</i>	Der Datentyp des Einzelfeldes in SPLV-Notation.

Der erforderliche Operand *splv-typ* definiert die Eigenschaften eines Einzelfeldes. Die SPLV-Notation für die Deklaration von Datentypen ist detailliert in der Sprachbeschreibung der Programmiersprache SPLV erläutert.

Hinweis: der Datentyp GLD wird nicht unterstützt.

Beispiele für unterstützte atomare EBCDIC-Datentypen:

<i>splv-typ</i>	Bedeutung
(ALN 5)	Ein alphanumerisches Feld der Länge 5.
(NOV 5 K 1)	Ein numerisches, vorzeichenloses Feld mit fünf Gesamtstellen und einer Nachkommastelle.
(NMV 4 K 0)	Ein numerisches, vorzeichenbehaftetes Feld mit vier Gesamtstellen ohne Nachkommastelle.

(GEP 9 K 3) Ein numerisches, gepacktes Feld mit neun Gesamtstellen und drei Nachkommastellen.

3.3.1.3. Operand *quelle*

Operand	Beschreibung
<i>quelle</i>	{ <i>quellelement</i> }[{ <i>s</i>]+ <i>s</i>]{ <i>quellelement</i> }. . .}; Eine Folge von ein oder mehreren Elementen des Typs <i>quellelement</i> ; je zwei aufeinanderfolgende Elemente werden durch den optional in Leerraum <i>s</i> eingebetteten Verkettungsoperator '+' miteinander verbunden. Das Ergebnis der Auswertung des Operanden <i>quelle</i> ist stets eine einzelne, zusammenhängende Resultatzeichenfolge. Der Resultatstring wird dem Feld als Wert zugewiesen.
<i>quellelement</i>	{ <i>meta</i> <i>mm_pfad</i> <i>konstante</i> <i>funktionsaufruf</i> }; Ein Metadatum oder ein Merkmal oder eine Konstante oder ein Funktionsaufruf. Merkmale dürfen nur in der INI-Daten verwendet werden.
<i>meta</i>	#[<i>metakontext</i>] <i>name</i> ; der Name eines Metadatums.
<i>metakontext</i>	{ <i>name</i> }. . .}; der Metadatenkontext in dem sich ein Metadatum befindet.
<i>mm_pfad</i>	[/][<i>pfadelement</i> [/]. . .]{ <i>merkmal</i> }; Adresse eines Merkmals, s. 3.6.3.
<i>funktionsaufruf</i>	<i>funktionsname</i> ([<i>param</i> [, <i>param</i>]. . .]); der Aufruf einer Funktion; Parameter sind funktionsabhängig.
<i>funktionsname</i>	<i>konst-name</i> ; der Name einer Funktion
<i>datml-name</i>	<i>name</i> ; Ein Name vom Typ <i>name</i>
<i>name</i>	Eine Folge von Buchstaben und/oder Ziffern, die mit einem Buchstaben oder einer Ziffer beginnt und endet und folgende Sonderzeichen enthalten kann: Unterstrich, Minuszeichen.
<i>string</i>	' <i>chars</i> ' <i>chars</i> ; eine in optionale Hochkommata eingeschlossene Zeichenfolge.
<i>konstante</i>	' <i>chars</i> '; eine Konstante; Hochkommata sind erforderlich.

quelle ist optional; fehlt die Angabe oder liefert sie einen leeren String als Resultat, wird das Feld mit typgerechten Anfangswerten initialisiert.

Die Auswertung des Operanden *quelle* einer Einzelfelddefinition liefert als Resultat eine (ggf. leere) Zeichenfolge, die dem Datenfeld zugewiesen wird, in dessen Definition *quelle* verwendet wird. Verkettung hat keine Auswirkung auf die Regeln für die die Konstruktion des Operanden und die Auswertung der Resultatzeichenfolge; hier gelten dieselben Regeln und Einschränkungen, insbesondere die Nichtzulässigkeit der Verwendung von Merkmalen in der INI-Meta und die Konformität mit dem Datentyp des Zielfeldes.

3.3.1.4. Operand *default*

Operand	Beschreibung
---------	--------------

<code>default</code>	Wie <code>quelle</code> ; definiert den Defaultwert
----------------------	---

`default` ist optional; fehlt die Angabe oder liefert sie einen leeren String als Resultat, wird das Feld mit typgerechten Anfangswerten initialisiert.

Der Defaultwert wird wirksam, wenn die Auswertung von `quelle` einen leeren String (d.h. einen String mit der Länge Null) ergibt.

HINWEIS: Enthält die Definition der Quelle bzw. des Defaults eine Konstante, ist das Ergebnis der Auswertung niemals leer. Mit Im Falle von

```
'Vorname :' + vorname | 'k.A.'
```

wird der Default daher in keinem Fall zum Zuge kommen.

3.3.2. Beispiele

```
f1="EF1 (ALN 2) '01'"
f2="EF2 (ALN 2) satzart"
f3="EF3 (ALN 5) #erhebung.kennung"
f5="EF5 (ALN 1)"

f1="EF1 (ALN 6) #berichtszeitraum.jahr+#berichtszeitraum.monat"

f1="EF1 (ALN 80) haushalt[1]/person[2]/name"

f1="EF1 (ALN 8) substring(steuerummer,1,6) + '00'

f1="EF1 (ALN 60) 'Vorname :' + split(name,',',2)
f2="EF2 (ALN 60) 'Nachname:' + split(name,',',1)

f8="EF8 (ALN 10) '12:' + bruttoUmsatz"

f3="EF3 (NMV 3 K 0) alter | -1
```

3.4. Strukturen

Strukturdefinitionen definieren zusammengesetzte – strukturierte – Felder einer Datensatzdefinition, im Gegensatz zu Einzelfelddefinitionen, die nur atomare Felder beschreiben. Die Strukturdefinition im formalen Sinn besteht aus einer *Strukturanfangsdefinition*, ein oder mehr Einzelfeld- und/oder weiteren Strukturdefinitionen, und einer *Strukturendedefinition*. Die Strukturanfangsdefinition definiert die Eigenschaften der Struktur. Inhaltlich gefüllt wird eine Struktur durch die der Strukturdefinition nachgeordneten Felddefinitionen, also Einzelfelddefinitionen und/oder weitere Strukturdefinitionen. Die Strukturendedefinition schließt die Strukturdefinition ab.

Strukturdefinitionen entsprechen in der Summe ihrer Eigenschaften einschließlich aller Einschränkungen den aus der Programmiersprache SPLV bekannten *Strukturvereinbarungen* und *Gruppenvereinbarungen* und unterstützen dieselben Typen von Strukturen:

- Nicht wiederholbare Feldgruppen,
- Feldgruppen mit einer festen Anzahl von Wiederholungen und
- Feldgruppen mit einer variablen Anzahl von Wiederholungen.

Während in Einzelfelddefinitionen der Operand *splv-typ* zur Beschreibung der Feldeigenschaften exakt der SPLV-Syntax folgt, wurde in Strukturanfangsdefinitionen für die Beschreibung der Struktureigenschaften mit dem Operanden *strukturtyp* eine eigene Syntax gewählt. Der Grund ist, daß SPLV eine Struktur aus der Abwesenheit einer Datentypdeklaration – also *splv-typ* – ableitet, wobei der Strukturname obligatorisch ist, wohingegen für den Zweck des INI-basierten Mappings die Namen von Feldern und Strukturen irrelevant sind, ebenso wie die in SPLV verwendete Angabe der Hierarchiestufe.

3.4.1. Strukturanfangsdefinition

Eine Strukturanfangsdefinition *strukturanfang* hat folgende Syntax:

```
strukturanfang : [strukturname] (strukturtyp) [mmgr_pfad]
```

3.4.1.1. Operand *strukturname*

Operand	Beschreibung
<i>strukturname</i>	<i>splv-name</i> ; Der Name der Struktur.
<i>splv-name</i>	Ein Name gemäß den Regeln für SPLV-Namen lt. SPLV-Sprachbeschreibung.

strukturname ist optional, da der Strukturname für die Erzeugung einer Struktur nicht notwendig ist; es wird empfohlen, *strukturname* aus dokumentarischen Gründen und für die Verwendung in Fehlermeldungen anzugeben.

3.4.1.2. Operand *strukturtyp*

Operand	Beschreibung
---------	--------------

<i>strukturtyp</i>	<i>str wfg vwfg</i> ; eine Struktur, eine wiederholte Feldgruppe oder eine variable Wiederholte Feldgruppe gemäß den SPLV-Konventionen.
<i>str</i>	STR [<i>s str-laenge</i>]; Eine nicht wiederholte Feldgruppe.
<i>wfg</i>	WFG <i>s str-anzahl</i> [<i>s str-laenge</i>]; Eine wiederholte Feldgruppe mit einer festen Anzahl von Instanzen.
<i>vwfg</i>	VWFG <i>s str-anzahl</i> [<i>s str-laenge</i>]; Eine wiederholte Feldgruppe mit einer variablen Anzahl von Instanzen.
<i>str-laenge</i>	<i>int_1..n</i> ; die Länge der Struktur.
<i>str-anzahl</i>	<i>int_1..n</i> ; die (maximale) Anzahl der Instanzen der Struktur.
<i>s</i>	Leerraum

strukturtyp ist erforderlich. Bei allen Strukturtypen ist die Angabe einer Länge optional; fehlt sie, wird die anhand der nachgeordneten Felddefinitionen berechnete verwendet, ist sie vorhanden, müssen angegebene und berechnete Länge übereinstimmen.

Beispiele für Strukturtypen:

<i>strukturtyp</i>	Bedeutung
(STR 15)	Ein nicht wiederholte Struktur mit der Gesamtlänge 15 Byte.
(WFG 8)	Eine wiederholte Feldgruppe mit 8 Instanzen; die Länge einer Instanz ergibt sich aus der Auswertung der nachgeordneten Felddefinitionen.
(WFG 18 25)	Eine wiederholte Feldgruppe mit 18 Instanzen von je 25 Byte Länge.
(VWFG 12)	Eine variable wiederholte Feldgruppe mit maximal 12 Instanzen; die Länge einer Instanz ergibt sich aus der Auswertung der nachgeordneten Felddefinitionen.
(VWFG 12 10)	Eine variable wiederholte Feldgruppe mit maximal 12 Instanzen von je 10 Byte Länge.

3.4.1.3. Operand *mmgr_pfad*

Der Operand *mmgr_pfad* ist detailliert im Abschnitt 3.6.3 beschrieben.

mmgr_pfad ist optional; fehlt die Angabe, oder ist die Merkmalsgruppe mit dem angegebenen Namen oder dem angegebenen Index nicht vorhanden, ist abhängig vom Strukturtyp wie folgt vorzugehen:

- bei einfachen Strukturen und wiederholten Feldgruppen mit fester Anzahl der Wiederholungen eine Instanz der Struktur mit typgerecht initialisierten Feldern zu erzeugen,
- bei wiederholten Feldgruppen mit variabler Anzahl der Wiederholungen wird keine Instanz der Feldgruppe erzeugt.

3.4.2. Nachgeordnete Felddefinitionen

Strukturanfangs- und Strukturendedefinition umschließen die nachgeordneten Felddefinitionen. Es gibt keine syntaktischen Unterschiede zwischen nachgeordneten Felddefinitionen und solchen, die nicht Bestandteil einer Struktur sind.

In einer Strukturdefinition gelten jedoch für die Spezifikation der Operanden *quelle* und *mmgr_pfad* nachgeordneter Felddefinitionen folgende besondere Regeln:

- Ist der Wert des Operanden *mmgr_pfad* einer Strukturanfangsdefinition eine Referenz auf eine Merkmalsgruppe, wird er den Operanden *quelle* und *mmgr_pfad* der nachgeordneten Felddefinitionen unter Einfügung eines Schrägstriches als Verkettungszeichen vorangestellt wenn diese relative Referenzen auf Merkmale und Merkmalsgruppen enthalten; absolute Referenzen bleiben unverändert (s. 3.6.3).
- Fehlt der Operand *mmgr_pfad*, dürfen die nachgeordneten Felddefinitionen keine relativen Referenzen enthalten; die betroffenen Datensatzfelder dürfen nur mit Konstanten und aus absolut adressierten Merkmalen befüllt oder implizit mit Anfangswerten initialisiert werden.

3.4.3. Strukturendedefinition

Die Strukturendedefinition *strukturende* markiert das Ende einer Strukturdefinition und hat folgende Syntax:

```
strukturende : [s](END)
```

Beispiel:

```
f23="(END)"
```

3.4.4. Beispiele

Einfache Struktur; sie wird gefüllt aus der Merkmalsgruppe *adresse* mit dem Index 1:

```
f11="EF8 (STR) adresse[]"
f12="EF8U1 (ALN 5) plz"
f13="EF8U2 (ALN 60) ort"
f14="EF8U3 (ALN 60) strasse"
f15=" (END)"
```

Wiederholte Feldgruppe mit fester Anzahl; sie wird gefüllt für alle Merkmalsgruppen *person* in allen Merkmalsgruppen *haushalt*:

```
f11="EF8 (WFG 10) haushalt[]/person[]"
f12="EF8U1 (NOV 3 K 0) alter"
f13="EF8U2 (ALN 1) geschlecht"
f14="EF8U3 (NOV 3 K 0) beruf"
f15=" (END)"
```

Wiederholte Feldgruppe mit variabler Anzahl und indexbildendem Wertmerk-

mal monat:

```
f11="EF8      (VWFG 12) monatsumsatz[monat]"
f12="EF8U1   (NOV 2 K 0) monat"
f13="EF8U2   (NOV 6 K 0) brutto"
f14="EF8U3   (NOV 6 K 0) menge"
f15="        (END)"
```

Die folgenden Beispiele erweitern das zweite Beispiel oben. Es wird angenommen, daß der Meldungsdatensatz ein Gebäude repräsentiert, mit je einer Merkmalsgruppe `haushalt` pro Haushalt und darin je eine Merkmalsgruppe `person` pro Person. `haushalt` enthält ein Merkmal `haushaltsnummer`, und auf Satzebene existiert ein Merkmal `gebaeudenummer`. Es wird über alle Haushalte hinweg für jede Person die definierte Struktur (eine Feldgruppe mit einer festen Anzahl von Wiederholungen) erzeugt.

Die Struktur des Meldungsdatensatzes (ohne Werte):

```
<satz>
  <mm name="gebaeudenummer">..</mm>
  <mmgr name="haushalt">
    <mm name="haushaltsnummer">..</mm>
    <mmgr name="person">
      <mm name="alter">..</mm>
      <mm name="geschlecht">..</mm>
      <mm name="beruf">..</mm>
    </mmgr>
  </mmgr>
  usw.
</satz>
```

Mit relativer Adressierung der Merkmale `haushaltsnummer` und `gebaeudenummer`:

```
f11="EF8      (WFG 10) haushalt[]/person[]"
f12="EF8U1   (NOV 3 K 0) alter"
f13="EF8U2   (ALN 1) geschlecht"
f14="EF8U3   (NOV 3 K 0) beruf"
f15="EF8U4   (NOV 3 K 0) ../haushaltsnummer"
f16="EF8U5   (NOV 3 K 0) ../../gebaeudenummer"
f15="        (END)"
```

Mit absoluter Adressierung der Merkmale `haushaltsnummer` und `gebaeudenummer`; in diesem Fall muß die Anwendung erkennen, daß `haushalt[]` im Operanden `mmgr_pfad` der Strukturanfangsdefinition liegt und dessen aktuellen Index verwenden.

```
f11="EF8      (WFG 10) haushalt[]/person[]"
f12="EF8U1   (NOV 3 K 0) alter"
f13="EF8U2   (ALN 1) geschlecht"
f14="EF8U3   (NOV 3 K 0) beruf"
```

```
f15="EF8U4 (NOV 3 K 0) /haushaltsnummer"  
f16="EF8U5 (NOV 3 K 0) /haushalt[]/gebaeudenummer"  
f15="      (END)"
```

3.5. Namen

Namen sind anwenderseitig definierte Bezeichner für bestimmte syntaktische Elemente, z.B. Merkmale oder die variablen Teile der Abschnittsnamen. Namen stammen entweder aus dem DatML/RAW-Dokument, den INI-Dateien oder aus einer SPLV-Datensatzbeschreibung:

- Namen aus DatML/RAW-Dokumenten haben Typ *datml-name*; sie haben meist eine fachliche Bedeutung, zum Beispiel als Merkmalsname;
- Namen aus den SPLV-Datensatzbeschreibungen haben Typ *splv-name*; sie repräsentieren SPLV-Objekte, z.B. die Felder eines Datensatzes.
- Variable Namen aus den INI-Dateien haben den Typ *name*; sie stellen Bezüge zwischen Elementen der INI-Dateien her; Beispiel: Abschnittsnamen.
- Konstante Namen aus den INI-Dateien haben den Typ *konst-name*; sie kommen als Aufzählungstyp, Funktionsname und in der Wertangabe von INI-Schlüsseln vor.

Namen haben die folgenden syntaktischen und semantischen Beschränkungen:

Name	Beschreibung
<i>name</i>	Eine Folge von Buchstaben und/oder Ziffern, die mit einem Buchstaben oder einer Ziffer beginnt und endet und folgende Sonderzeichen enthalten kann: Unterstrich, Minuszeichen.
<i>splv-name</i>	Es gelten die für SPLV-Namen geltenden Restriktionen.
<i>datml-name</i>	Ein beliebige Folge druckbarer Zeichen, die folgende Zeichen nicht enthalten darf: Anführungszeichen, Hochkomma.
<i>konst-name</i>	Ein konstanter, vordefinierter Name

WICHTIG: Beim Vergleich von Namen gelten typabhängige Regeln:

- Für die Typen *name*, *datml-name* und *konst-name* wird Groß- und Kleinschreibung berücksichtigt.
- Für den Typ *splv-name* wird Groß- und Kleinschreibung ignoriert.

3.6. Merkmale und Merkmalsgruppen

Die statistischen Werte werden in Form von wertebehafteten Merkmalen geliefert; Merkmale können in einem Meldungsdatensatz in Merkmalsgruppen organisiert sein.

3.6.1. Merkmale

DatML/RAW unterscheidet zwischen den Merkmalstypen Wertmerkmal, Ordnungsmerkmal und Hilfsmerkmal, die durch unterschiedliche Elementtypen dargestellt werden. Der Name eines Merkmals ist in allen diesen Elementtypen der Wert des Attributes `name` und wird behandelt wie eine Zeichenfolge vom Typ `datml-name` (s. 3.3). Der Vergleich der Merkmalsnamen geschieht daher *ohne* Berücksichtigung von Groß- und Kleinschreibung. Die Regeln für Namenskonflikte lt. DatML/RAW-Spezifikation sind zu beachten.

3.6.2. Merkmalsgruppen

In DatML/RAW haben Merkmalsgruppen die Funktion, logisch zusammengehörende Merkmale und ggf. weitere Merkmalsgruppen in einer Struktur zusammenzufassen und innerhalb eines Meldungsdatensatzes wiederholbar zu machen. Sie werden in DatML/RAW durch das Element `<mmgr>` mit einem `name`-Attribut dargestellt. Merkmalsgruppen werden mit ihrem Namen und einem Index referenziert, der gemäß den Regeln der DatML/RAW-Spezifikation gebildet werden muß. Merkmale und Merkmalsgruppen haben einen gemeinsamen Namensraum, d.h. sie müssen sich bereits durch ihren Namen unterscheiden.

Merkmalsgruppen werden in folgenden Fällen verwendet:

- Befüllen von einfachen und wiederholten Strukturen (3.4),
- Ausgabe von Datensätzen steuern (nur in Datendateien, 2.3.4.1).

3.6.3. Adressierung

Alle Merkmale haben – bezogen auf die Meldung und den aktuellen Meldungsdatensatz – einen gemeinsamen Namensraum, d.h. alle Merkmale müssen sich ungeachtet ihres Merkmalstyps allein durch ihre Namen unterscheiden. Die Angabe des Merkmalsnamens reicht daher zur Adressierung eines Merkmals aus, und der Merkmalstyp ist für das Mappen von Merkmalen auf Datensatzfelder grundsätzlich irrelevant.

In einigen Fällen, in denen nur Merkmale eines bestimmten Typs zulässig sind, wird der entsprechende Merkmalstyp vorausgesetzt, d.h. das Merkmal wird nur unter den Merkmalen dieses Typs gesucht. Die Tabelle zeigt die Abhängigkeiten zwischen der Verwendung eines Merkmals und den Merkmalstypen:

Verwendung	Default-Merkmalstyp
Operanden <code>quelle</code> und <code>default</code> in Einzelfelddefinitionen (3.3.1)	Kein Default; es sind alle Merkmalstypen zulässig.

Schlüssel <i>ausgabe</i> (3.2.4)	Kein Default; es sind alle Merkmalstypen zulässig.
Externe Dateinamen (3.12)	Hilfsmerkmal; es sind nur Hilfsmerkmale zulässig.
Merkmalsgruppenindex (3.6.2)	Wertmerkmal; es sind nur Wertmerkmale zulässig.

Merkmalsgruppen benutzen denselben Namensraum, sie müssen daher von den Merkmalen verschiedene Namen haben.

Befindet sich ein Merkmal oder eine Merkmalsgruppe innerhalb einer (Hierarchie von übergeordneten) Merkmalsgruppe(n), ist es meist notwendig, den Pfad dorthin anzugeben (siehe 3.4.2).

Ausgangspunkt der Hierarchie von Merkmalen und Merkmalsgruppen ist der Meldungsdatensatz. In der Merkmalsgruppenhierarchie ist er der Wurzelknoten, an dem eine Struktur aus Merkmalsgruppen und Merkmalen hängt, die wie ein hierarchisches Dateisystem eine baumartige Form annimmt. Geht man in dieser Analogie weiter, entspricht eine Merkmalsgruppe einem Ordner und ein Merkmal einer Datei.

Die Adressierung von Dateien und Ordner in einem Dateisystem ist den meisten Anwendern geläufig. Ausgehend von einem bestimmten Knoten im Dateisystem wird der Pfad angegeben, der zum Zielknoten führt. Der gleiche Mechanismus wird auch verwendet, um Knoten in hierarchischen Dokumenten zu adressieren, z.B. mit [XPath](#). Es liegt daher nahe, diese Form der Pfadangaben für die Adressierung von Merkmalsgruppen zu wählen.

Eine Pfadangabe, die eine Merkmalsgruppe (oder ein nachgeordnetes Merkmal) adressiert, hat folgende Eigenschaften:

- Sie kann *relativ* sein, d.h. vom aktuellen Knoten ausgehen, oder *absolut*, d.h. vom aktuellen Meldungsdatensatz aus adressieren; ein absoluter Pfad beginnt mit einem Schrägstrich.
- Zwei Elemente einer Pfadangabe werden durch einen Schrägstrich getrennt.
- An jedem Punkt ist es möglich, durch Angabe eines Punktes den aktuellen Knoten und durch Angabe zweier aufeinanderfolgender Punkte den Elternknoten anzusprechen.

Aus diesen Eigenschaften leitet sich die Syntax der Operanden für die Adressierung von Merkmalen und Merkmalsgruppen ab; sie unterliegen verschiedenen Restriktionen:

- *pfad* adressiert entweder ein Merkmal oder einer Merkmalsgruppe,
- *mm_pfad* adressiert ausschließlich ein Merkmal,
- *mmgr_pfad* adressiert ausschließlich eine Merkmalsgruppe,

Operand	Beschreibung
<i>pfad</i>	<code>[/][pfadelement[/]...]{merkmalsgruppe merkmal}</code> Eine Pfadangabe <i>pfad</i> besteht aus einer Folge von ein oder

	mehr optionalen, durch Schrägstriche getrennten Pfadelementen <i>pfadelement</i> , der ein Schrägstrich vorausgeht, wenn es sich um eine absolute Pfadangabe handelt, und die mit der Angabe eines Merkmals <i>merkmal</i> oder einer Merkmalsgruppe <i>merkmalsgruppe</i> endet.
<i>mm_pfad</i>	<code>[/][pfadelement[/]..]{merkmal}</code> Eine Pfadangabe <i>pfad</i> besteht aus einer Folge von ein oder mehr optionalen, durch Schrägstriche getrennten Pfadelementen <i>pfadelement</i> , der ein Schrägstrich vorausgeht, wenn es sich um eine absolute Pfadangabe handelt, und die mit der Angabe eines Merkmals <i>merkmal</i> endet.
<i>mmgr_pfad</i>	<code>[/][pfadelement[/]..]{merkmalsgruppe}</code> Eine Pfadangabe <i>pfad</i> besteht aus einer Folge von ein oder mehr optionalen, durch Schrägstriche getrennten Pfadelementen <i>pfadelement</i> , der ein Schrägstrich vorausgeht, wenn es sich um eine absolute Pfadangabe handelt, und die mit der Angabe einer Merkmalsgruppe <i>merkmalsgruppe</i> endet.
<i>pfadelement</i>	<code>pfadelement: merkmalsgruppe '.' '..'</code> Eine Pfadelement <i>pfadelement</i> adressiert ausgehend von der durch die vorausgehenden Pfadelemente erreichten aktuellen Position entweder eine Merkmalsgruppe <i>merkmalsgruppe</i> , den aktuellen Knoten (.) oder den Elternknoten (..).
<i>merkmalsgruppe</i>	<code>merkmalsgruppe: datml-name '['[index]']'</code> Der Name einer Merkmalsgruppe, gefolgt vom Merkmalsgruppenindex, der in eckigen Klammern eingeschlossen ist.
<i>index</i>	<code>int_1..n</code> ; Ein Merkmalsgruppenindex als direkter Index in Form einer ganzen Zahl größer Null.

Der Index *index* ist stets ein *direkter Index*, d.h. eine natürliche Zahl größer Null. Die eckigen Klammern sind immer anzugeben. Zwischen dem Merkmalsgruppennamen und der öffnenden Klammer sowie innerhalb der Klammern ist kein Leerraum zulässig. Die Indexangabe *index* selbst darf fehlen und wird kontextabhängig ersetzt:

- Ein fehlender Index wird durch den Wert 1 ersetzt, wenn der Kontext nur die Ansprache einer einzelnen Instanz einer Merkmalsgruppe erlaubt, z.B. bei der Zuweisung zur einer nicht wiederholten Struktur:
- ein fehlender Index wird nacheinander durch die Indizes der vorhandenen Instanzen der Merkmalsgruppe ersetzt, wenn der Kontext die Ansprache aller Instanzen verlangt, z.B. bei der Zuweisung zur einer wiederholten Struktur oder im Schlüssel *ausgabe* einer Datensatzdefinition.

Der Wert eines Index ist fehlerhaft, wenn er

- nicht numerisch ist oder < 1 ,
- in der Zuweisung zu einer wiederholten Feldgruppe größer ist als die maximale Zahl von Wiederholungen.

Hilfsmerkmale sind in der Struktur eines DatML/RAW-Dokumentes den Meldungsdatensätzen übergeordnet und daher logisch Bestandteil jedes nachgeordneten Meldungsdatensatzes. Da sie den Namensraum mit gewöhnlichen Merkmalen teilen,

können sie wie Merkmale adressiert werden, die direkt einem Meldungsdatensatz nachgeordnet, also nicht Bestandteil einer Merkmalsgruppe sind.

Beispiele:

Pfad

`/haushalt[]`

`/haushalt[5]`

`person[]`

`/berichtseinheitID`

`../lfdnummer`

Erläuterung

Alle Merkmalsgruppen `haushalt` im aktuellen Meldungsdatensatz

Die Merkmalsgruppe `haushalt` im aktuellen Meldungsdatensatz mit dem Index 5

Die Merkmalsgruppe `person` im aktuellen Knoten.

Das Merkmal oder Hilfsmerkmal `berichtseinheitID` im aktuellen Knoten.

Das Merkmal `lfdnummer` im Elternknoten des aktuellen Knotens.

Absolute Adressierung ist vor allem notwendig, wenn aus dem Kontext einer Merkmalsgruppe heraus ein Knoten angesprochen wird, der nicht im aktuellen Pfad liegt, z.B. ein Merkmal, das direkt dem Meldungsdatensatz oder einer anderen Merkmalsgruppe nachgeordnet ist. Hilfsmerkmale müssen aus dem Kontext einer Merkmalsgruppe heraus immer absolut adressiert werden, da sie logisch direkt dem Satzknotten nachgeordnet sind.

3.7. Metadaten

Metadaten sind z.B. Berichtszeitraum, Absender etc. Da bestimmte Typen von Metadaten (z.B. eine Straße) in mehreren Kontexten erlaubt sind, muß die Referenzierung den Kontext enthalten. Der Kontext, der den lokalen Namen des Metadatum qualifiziert, wird als Pfad mit durch Punkt getrennten Teilnamen spezifiziert.

Eine Liste der zulässigen Kontexte und Metadaten findet sich in Kapitel 6.

3.8. Konstanten

Eine Konstante *konstante* wird durch ein in Hochkommata eingeschlossenes Literal dargestellt:

```
konstante : ['']literal['']
```

Die Hochkommata können entfallen, wenn keine Verwechslungsgefahr mit anderen syntaktischen Elementen besteht, z.B. wenn für einen Schlüssel nur ein Wert zulässig ist.

- Konstanten werden ohne explizite Notation nur dort verwendet, wo einem Schlüssel eine Folge von ein oder mehr einfachen alphanumerischen Werten zugewiesen wird.

Beispiele:

```
dateien="ku101 ku102"  
dateien=""'ku101' 'ku102'""  
  
satzartwerte="1 2"  
satzartwerte=""'1' '2'""  
  
f1="EF1 (ALN 1) 'A'"
```

Konstanten werden *nicht* verwendet:

- Wo einem Schlüssel eine Folge von ein oder mehr *numerischen* Werten zugewiesen wird.
- Wo einem Schlüssel ein Wert aus einer Liste möglicher *vordefinierter* Werte zugewiesen wird.

In diesen Fällen darf der Wert *nicht* in Hochkommata eingeschlossen werden. Beispiele:

```
anzahlSatzdefinitionen="5"  
ausgabe="open-doc"
```

3.9. Funktionen

Funktionen erlauben die (parametrisierbare) Ausführung bestimmter Aktionen. Eine Funktion wird aufgerufen mit ihrem Funktionsnamen, gefolgt von einem Paar runder Klammern, welche eine ggf. leere, mit Kommata separierte Liste der Funktionsparameter einschließen:

```
funktionsaufruf : funktionsname([param[,param]..])
funktionsname : konst-name
```

3.9.1. Stringverarbeitung

Die Funktionen `collapse()` und `normalize()` behandeln Leerraumzeichen wie Tabulatoren und Zeilenumbrüche; sie sind u.a. notwendig, um mehrzeilige Texte, wie sie zum Beispiel in Kommentarfeldern vorkommen können, in einzeilige zu konvertieren.

`split()` und `substring()` ermöglichen den Zugriff auf Teilzeichenfolgen. Mit ihnen können einzelne Elemente aus einer zusammengesetzten oder formatierten Zeichenfolge oder Werte aus einer trennzeichenseparierten Liste entnommen werden.

3.9.1.1. collapse()

<code>collapse()</code>	
Bereich	Zeichenmanipulation
Syntax	<code>collapse(z)</code> <code>z : meta mm_pfad</code>
Rückgabewert	Zeichenkette

`collapse(z)` normalisiert zunächst die Zeichenkette `z` wie mit einem Aufruf der Funktion `normalize(z)`, komprimiert anschließend jede Folge von Leerzeichen zu einem einzelnen Leerzeichen und entfernt schließlich alle Leerzeichen am Beginn und am Ende der Zeichenkette. Besteht `z` nach der Normalisierung nur aus Leerzeichen, gibt die Funktion ein einzelnes Leerzeichen zurück.

Beispiel:

Eingabewert	Rückgabewert
A drum, a drum, Macbeth doth come!	A drum, a drum, Macbeth doth come!

3.9.1.2. normalize()

<code>normalize()</code>	
Bereich	Zeichenmanipulation
Syntax	<code>normalize(z)</code> <code>z : meta mm_pfad</code>
Rückgabewert	Zeichenkette

`normalize()` normalisiert eine Zeichenkette, indem sie Tabulatoren, Wagenrücklauf (Carriage Return) und Zeilenvorschub (Linefeed) in einfache Leerzeichen (Blanks, ASCII #0x20, EBCDIC #0x40) konvertiert. Die Eingabezeichenkette *z* bleibt unverändert, wenn sie keine dieser Zeichen enthält.

Beispiel:

Eingabewert	Rückgabewert
<code>A drum, a drum, Macbeth doth come!</code>	<code>A drum, a drum, Macbeth doth come!</code>

3.9.1.3. split()

<code>split()</code>	
Bereich	Teilzeichenketten
Syntax	<pre>split(z[, [t][, i]]) z : meta mm_pfad t : konstante_1..1 ; ein Trennzeichen i : Ganzzahliger Index; für i gilt: 0 <= i</pre>
Rückgabewert	Zeichenkette

`split()` zerlegt eine Zeichenkette *z* anhand eines Trennzeichens *t* und liefert die *i*-te Teilzeichenkette zurück, die leer ist, wenn *z* leer ist oder die Teilzeichenkette tatsächlich leer ist oder *i* Null ist oder größer als die Anzahl der Teilzeichenketten. Wenn *t* nicht in *z* enthalten ist, liefert die Funktion *z*.

Für ein fehlendes Trennzeichen *t* wird das Leerzeichen, für einen fehlenden Index *i* der Wert `1` angenommen.

Beginnt *z* mit *t*, liefert der Zugriff auf die Teilzeichenkette mit dem Index `1` eine leere Zeichenkette.

Beispiel String `05.Stnr2375-A97.000` in Merkmal `steuernummer`:

Funktionsaufruf	Rückgabewert
<code>split(steuernummer, '.')</code>	<code>05</code>
<code>split(steuernummer, '.', 1)</code>	<code>05</code>
<code>split(steuernummer, '.', 2)</code>	<code>Stnr2375-A97</code>
<code>split(steuernummer, '-', 2)</code>	<code>-A97.000</code>
<code>split(steuernummer, '/', 1)</code>	<code>05.Stnr2375-A97.000</code>
<code>split(steuernummer, '.', 0)</code>	<code>Leerstring</code>
<code>split(steuernummer, '.', 4)</code>	<code>Leerstring</code>

Beispiel String `+a++b` in Merkmal `ausdruck`:

Funktionsaufruf	Rückgabewert
<code>split(ausdruck, '+', 1)</code>	<code>Leerstring (leer vor Trennzeichen)</code>
<code>split(ausdruck, '+', 2)</code>	<code>a</code>
<code>split(ausdruck, '+', 3)</code>	<code>Leerstring (leer zwischen Trennzeichen)</code>

```

split(ausdruck, 'a', 1)      +
split(ausdruck, 'a', 2)      ++b
split(ausdruck, 'b', 2)      Leerstring (Leer nach Trennzeichen)

```

3.9.1.4. substring()

substring()	
Bereich	Teilzeichenketten
Syntax	<pre> substring(z[, [s][, e]]) z : meta mm_pfad s : Ganzzahliger Index; für s gilt: -Länge(z) <= -s <= -1 und 1 >= s <= Länge(z) e : Ganzzahliger Index; für e gilt: e => s </pre>
Rückgabewert	Zeichenkette

`substring()` liefert eine Teilzeichenkette der Zeichenkette z , beginnend mit dem Zeichen mit dem Startindex s bis einschließlich des Zeichens mit dem Endeindex e , maximal jedoch bis zum Ende der Zeichenkette z . Die Teilzeichenkette ist leer, wenn z leer ist oder s größer ist als e oder als die Länge von z , und sie ist kürzer als die aufgrund der Indizes möglichen maximalen Länge, wenn e größer ist als die Länge von z .

Wenn s einen negativen Wert hat, wird der Startindex nicht vom Beginn, sondern von Ende der Zeichenkette z her ermittelt: Das erste Zeichen der Teilzeichenkette ist dann der Index mit dem Wert $\text{Länge}(z) - s + 1$. Wenn der errechnete Startindex kleiner als 1 ist, wird er durch 1 ersetzt, d.h. die Teilzeichenkette beginnt mit dem ersten Zeichen der Zeichenkette z . Ansonsten gelten die Regeln für einen positiven Startindex. Mit einem negativem Startindex können auf einfache Weise die letzten s Zeichen einer Zeichenkette selektiert werden.

Für einen fehlenden Startindex s wird 1, für einen fehlenden Endeindex e die Länge der Zeichenkette z angenommen.

Beispiel String `05.Stnr2375-A97.000` in Merkmal `steuernummer`:

Funktionsaufruf	Rückgabewert
<code>substring(steuernummer)</code>	<code>05.Stnr2375-A97.000</code>
<code>substring(steuernummer, 1)</code>	<code>05.Stnr2375-A97.000</code>
<code>substring(steuernummer, , 2)</code>	<code>05</code>
<code>substring(steuernummer, 2, 6)</code>	<code>5.Stn</code>
<code>substring(steuernummer, 25)</code>	leerer String (s größer als Länge von z)
<code>substring(steuernummer, 13, 25)</code>	<code>A97.000</code> (e größer als Länge von z)
<code>substring(steuernummer, -5)</code>	<code>7.000</code> (die letzten fünf Zeichen von z)

3.10. Reguläre Ausdrücke

Reguläre Ausdrücke (*RE*) sind eine Methode, Muster für das Durchsuchen von Daten nach Zeichenfolgen zu konstruieren. Es gibt zahlreiche Implementierungen für Sprachen wie Java, Perl, usw. Die Referenzimplementierung für diese Beschreibung ist das Java-Package `java.util.regex`. Diese Spezifikation beschränkt sich auf eine Grundmenge von Funktionalitäten dieses Packages, von denen angenommen wird, daß sie auch von anderen Implementierungen unterstützt werden. Die Syntaxbeschreibung basiert auf der Syntax regulärer Ausdrücke für XML-Schema Datentypen.

Reguläre Ausdrücke sind in den Abschnitten vom Typ `kontext.kontext` als Werte der meisten dort erlaubten Schlüssel zulässig (s. 3.2.5). Der aktuelle Wert des durch einen Schlüssel referenzierten Metadatums wird auf Übereinstimmung mit dem regulären Ausdruck geprüft. Dies erlaubt z.B. den einfachen Vergleich von Teilzeichenketten.

3.10.1. Syntax

Die folgende Beschreibung regulärer Ausdrücke beschränkt sich auf die folgenden vier elementaren Komponenten:

- Eine Angabe, nach welchen Zeichen zu suchen ist; diese Komponente wird als **Atom** bezeichnet.
- Für ein Atom kann angegeben werden, wie oft es auftreten soll; diese Komponente nennt man **Quantor**.
- Ein **Anker** definiert eine Grenze, z.B. den Zeilenanfang.
- Ein regulärer Ausdruck kann schließlich aus mehreren Alternativen bestehen, d.h. aus mit Oder verknüpften Teilausdrücken; ein solcher Teilausdruck heißt **Zweig**.

Ein regulärer Ausdruck *RE* besteht mindestens aus einem Zweig mit mindestens einem Atom:

```

RE : zweig[|zweig]..
zweig : [anker]{atom[quantor]}..[anker]
atom : literal|escapesequenz|zeichenklasse|PRE
literal : char
escapesequenz : .|\escape
zeichenklasse : '['[nicht-op]zeichenmenge..'']
nicht-op : ^
zeichenmenge : zeichen|zeichenbereich[ausschlussmenge]
zeichen : literal|escapesequenz
ausschlussmenge : &&' zeichen|zeichenbereich'
zeichenbereich : literal'-literal
PRE : '('RE')

```

3.10.1.1. Atom

Ein Atom definiert eine Menge von Zeichen und ist entweder ein Literal *literal*, eine Escapesequenz *escapesequenz*, eine Zeichenklasse *zeichenklasse* oder ein

vollständiger geklammerter regulärer Ausdruck *PRE* (*parenthesised regular expression*).

<i>atom</i>	
<i>literal</i>	Ein Zeichen, das für sich selbst steht
<i>escapesequenz</i>	Eine Escapesequenz, die eine Menge von Zeichen darstellt.
<i>zeichenklasse</i>	Eine in eckigen Klammern eingeschlossene Liste oder ein Bereich von Zeichen; eine Zeichenklasse kann negiert werden.
<i>PRE</i>	Ein vollständiger, in runden Klammern eingeschlossener regulärer Ausdruck.

3.10.1.1.1. Literal

Ein Literal ist ein Zeichen, das für sich selbst steht; Groß- und Kleinschreibung sind grundsätzlich relevant: *a* ergibt einen Treffer für jedes Vorkommen von *a*, *A* für jedes Vorkommen von *A* und so weiter. Die Folge von Literalen *ABC* steht für *A* gefolgt von *B* gefolgt von *C*. Jedes dieser Literale ist ein separates Atom. In dem regulären Ausdruck *ABC?* ist daher nur der Buchstabe *C* optional.

3.10.1.1.2. Escapesequenz

Escapesequenzen repräsentieren ein oder mehr Zeichen; sie beginnen mit einem umgekehrten Schrägstrich: *\d* steht für eine Folge von Ziffern, *\D* für eine Folge von Nicht-Ziffern, *\t* für einen Tabulator usw. Der Punkt (.) als spezielle Escapesequenz steht für ein beliebiges Zeichen, ausgenommen Zeilenende.

<i>escapesequenz</i>	
.	Ein beliebiges Zeichen
\\	Der linksgeneigte Schrägstrich
<i>\0n</i>	Das Zeichen mit dem oktalen Wert <i>0n</i> ($0 \leq n \leq 7$)
<i>\0nn</i>	Das Zeichen mit dem oktalen Wert <i>0nn</i> ($0 \leq n \leq 7$)
<i>\0mnn</i>	Das Zeichen mit dem oktalen Wert <i>0mnn</i> ($0 \leq m \leq 3, 0 \leq n \leq 7$)
<i>\0xhh</i>	Das Zeichen mit dem hexadezimalen Wert <i>0xhh</i>
<i>\0xhhhh</i>	Das Zeichen mit dem hexadezimalen Wert <i>0xhhhh</i>
<i>\t</i>	Tabulator
<i>\n</i>	Zeilenvorschub (Newline)
<i>\r</i>	Wagenrücklauf (Carriage Return)

Die folgenden Escapesequenzen repräsentieren vordefinierte Zeichenklassen; sie sind also eine Kurzform, die anstelle der ausformulierten Zeichenklasse verwendet werden kann.

<i>escapesequenz</i>	
<i>\d</i>	Eine Ziffer: <i>[0-9]</i>
<i>\D</i>	Ein Zeichen außer einer Ziffer: <i>[^\d]</i>
<i>\s</i>	Leerraum: <i>[\t\n\x0B\f\r]</i>
<i>\S</i>	Ein Zeichen außer Leerraum: <i>[^\s]</i>
<i>\w</i>	Eine Wortzeichen: <i>[a-zA-Z_0-9]</i>
<i>\W</i>	Ein Zeichen außer einem Wortzeichen: <i>[^\w]</i>

Mit Escapesequenzen können Zeichen maskiert werden:

<i>escapesequenz</i>	
<code>\c</code>	Maskiert das folgende Zeichen <i>c</i>
<code>\Q</code>	Maskiert alle folgenden Zeichen bis <code>\E</code>
<code>\E</code>	Ende der Maskierung alle Zeichen ab dem vorausgehenden <code>\Q</code>

3.10.1.1.3. Zeichenklasse

Sogenannte Zeichenklassen beschreiben Bereiche von Zeichen; sie sind in eckigen Klammern eingeschlossen: `[ABC]` ergibt einen Treffer, wenn das zu prüfende Zeichen einer der Buchstaben *A*, *B* oder *C* ist, `[A-Z]` ergibt einen Treffer für jeden Großbuchstaben und `[0-3]` für jede Ziffer kleiner als vier. Der Nicht-Operator `^` (Circumflex) kehrt die Menge der Zeichen um: `[^0-3]` enthält also jede Ziffer *größer* als drei.

Einzelzeichen und Bereichsangaben können kombiniert werden: `[a-zA-Z_0-9]` ergibt einen Treffer für alle Buchstaben, den Unterstrich und alle Ziffern.

Ausschlußmengen ermöglichen die Reduktion eines zuvor definierten Zeichenbereiches um bestimmte Zeichen; sie werden als Zeichenklasse angegeben, welcher der Ausschlußoperator `&&` vorausgeht: `[A-Z&&[^F-N]]` bedeutet alle Großbuchstaben außer *F* bis *N*, und `200[\d&&[^5]]` steht für die Zahl von 2000 bis 2009 außer 2005.

Achtung: Innerhalb einer Zeichenklasse ändert sich die Bedeutung einiger Metazeichen: der Punkt (`.`) verliert seine Bedeutung als Metazeichen, und das Minuszeichen (`-`) zeigt einen Bereich an.

3.10.1.1.4. Geklammerter regulärer Ausdruck

Ein Atom kann auch ein vollständiger, in runden Klammern eingeschlossener regulärer Ausdruck sein. Ein häufiger Anwendungsfall sind komplexe Alternativen innerhalb eines äußeren regulären Ausdrucks: `AB(\d{3}|-[CEF]\d{2})` bedeutet *AB* gefolgt von drei Ziffern *oder* gefolgt von einem Minuszeichen und *C*, *E* oder *F* und zwei Ziffern.

3.10.1.2. Quantor

Quantoren legen für ein Atom fest, wie häufig Zeichen aus der durch das Atom definierten Zeichenmenge auftreten dürfen bzw. müssen. Standardmäßig muß die angegebene Zeichenfolge genau einmal erscheinen: `b?` steht für ein oder kein Zeichen *b*, `a+` steht für eine Folge von ein oder mehr Zeichen *a*, `\d{3}` für genau drei und `\d{3,7}` für drei bis sieben Ziffern. Da ein Atom ein geklammerter regulärer Ausdruck sein kann, können auch ganze reguläre Ausdrücke quantifiziert werden: `(Jo)+` ergibt einen Treffer für alle Zeichenfolgen *Jo*, *JoJo*, *JoJoJo* usw.

<i>quantor</i>	Bedeutung
<code>*</code>	0-n Instanzen des vorherigen Ausdrucks

?	0-1 Instanzen des vorherigen Ausdrucks
+	1-n Instanzen des vorherigen Ausdrucks
{ <i>n</i> }	Genau <i>n</i> Instanzen des vorherigen Ausdrucks
{ <i>n</i> , }	Mindestens <i>n</i> Instanzen des vorherigen Ausdrucks
{ <i>n</i> , <i>m</i> }	Mindestens <i>n</i> und höchsten <i>m</i> Instanzen des vorherigen Ausdrucks

3.10.1.3. Anker

Anker werden benutzt, um auszudrücken, daß ein Atom an einer bestimmten Stelle oder Grenze stehen soll.

<i>anker</i>	Bedeutung
^	Zeilenanfang (bzw. Stringanfang)
\$	Zeilenende (bzw. Stringende)

3.10.1.4. Zweig

Schließlich kann ein regulärer Ausdruck durch das Pipe-Symbol (|) in Zweige unterteilt sein; das Pipe-Symbol stellt eine Oder-Verknüpfung dar, d.h. jeder Zweig ist eine Alternative im regulären Ausdruck: $\backslash d\{3\} | x\{3\}$ steht für eine Folge von drei Ziffern *oder* xxx.

3.10.2. Beispiele

RE	Erläuterung
a	Ist wahr, wenn in einer Zeichenfolge der Buchstabe a vorkommt.
^a	Ist wahr, wenn eine Zeichenfolge mit dem Buchstaben a beginnt.
a\$	Ist wahr, wenn eine Zeichenfolge mit dem Buchstaben a endet.
^a\$	Ist wahr, wenn eine Zeichenfolge mit demselben Buchstaben a beginnt und endet, d.h. nur aus einem Buchstaben a besteht .
a{3}	Ist wahr, wenn eine Zeichenfolge die Teilzeichenfolge aaa enthält.
a+	Ist wahr für jede Zeichenfolge a, aa, aaa usw.
ba?	Ist wahr für jede Zeichenfolge b, ba, baa, baaa usw.
(ba)+	Ist wahr für jede Zeichenfolge ba, baba, bababa usw.
[ABC]	Ist wahr für jedes Zeichen A oder B oder C.
[ABC]{2}	Ist wahr für AA, AB , AC , BA , BB , BC , CA, CB und CC.
200[0-9] 200\d	Sind wahr für 2000, 2001 bis 2009.
200([0-3] [5-9]) 200[0-9&&[^4]] 200[\d&&[^4]]	Sind wahr für 2001 bis 2003 und 2005 bis 2009.
(200[0-9])\$	Sind wahr, wenn eine Zeichenfolge mit einer Zahl von

<code>(200\d)\$</code>	2000 bis 2009 endet.
<code>\d</code>	Ist wahr für jede Ziffer von 0 bis 9.
<code>\d{5}</code>	Ist wahr für jede fünfstellige Zahl.
<code>\d{4,8}</code>	Ist wahr für jede vier- bis achtstellige Zahl.
<code>^\d{5}[\-\/]</code>	Ist wahr, wenn eine Zeichenfolge mit einer fünfstelligen Zahl beginnt, gefolgt von einem Minuszeichen oder einem Schrägstrich; das Minuszeichen muß durch einen linksgeigten Schrägstrich (\) maskiert werden, da es innerhalb einer Zeichenklasse steht und somit ohne Maskierung als Bereichszeichen interpretiert würde.
<code>([0-9A-F]{2}){4}</code>	Ist wahr für die hexadezimale Repräsentation einer vier Byte langen Zeichenfolge, wobei alle Buchstaben groß geschrieben sein müssen, also <code>0AF3445C</code> , aber nicht <code>0af3445c</code> .

3.11. Schlüssel und Operanden

3.11.1. Schlüssel (Alphabetisch)

Die Schlüssel für den Abschnitt `kontext.kontext` sind separat in Abschnitt 3.2.5 erläutert.

Schlüssel	Beschreibung
<code>anzahlFelder</code>	Anzahl der Felddefinitionen <code>fn</code>
<code>anzahlSatzdefinitionen</code>	Anzahl der Datensatzdefinitionen <code>meta.satz.n</code> ; für <code>n</code> gilt: $1 \leq n \leq 256$
<code>ausgabe (INI-Daten)</code>	<code>meldungsdatensatz merkmalsgruppe::mgr_pfad merkmal::mm_pfad[={konstante RE}]</code> Legt analog zum gleichnamigen Schlüssel im Abschnitt <code>meta.satz.n</code> fest, welches Ereignis die Ausgabe des Datensatzes bewirkt: Entweder eine (bestimmte) Instanz einer Merkmalsgruppe oder ein Merkmal (mit einem bestimmten Wert) oder der aktuelle Meldungsdatensatz (Voreinstellung).
<code>ausgabe (INI-Meta)</code>	<code>open-doc close-doc close-rep</code> Verarbeitungszeitpunkt, zu dem der Satz ausgegeben wird.
<code>dateien</code>	Eine Liste von Name(n) <code>datei</code> der Dateidefinitionen <code>datei.init</code> ; mehrere Namen müssen durch Leerzeichen oder Tabs getrennt werden, und es muß mindestens ein Name vorhanden sein.
<code>datensatzname</code>	Name des Datensatzes für dokumentarische Zwecke, z.B. bei der Ausgabe von Fehlertexten.
<code>dezimalzeichen</code>	<code>. ,</code> Definiert das Zeichen, das bei der Ausgabe numerischer Felder bei Bedarf als Dezimalzeichen verwendet wird. Ist der angegebene String länger als ein Zeichen, wird nur das erste Zeichen verwendet.
<code>externerNameDaten</code>	Template für den externen Namen der Datendatei (s. 3.12). Der Name kann eine absolute oder relative Pfadangabe enthalten.
<code>externerNameMeta</code>	Template für den externen Namen der Metadatei (s. 3.12). Der Name kann eine absolute oder relative Pfadangabe enthalten.
<code>feldtrennzeichen</code>	Default: <code>;</code> (Semikolon). Definiert das Zeichen, das bei der Ausgabe im Speicherformat <code>ascii-csv</code> als Feldtrennzeichen verwendet wird. Ist der angegebene String länger als ein Zeichen, wird nur das erste Zeichen verwendet.
<code>fn</code>	<code>einzelfelddefinition strukturdefinition</code> , s. 3.3 bzw. 3.4; für <code>n</code> gilt: $1 \leq n \leq \text{anzahlFelder}$
<code>kontext</code>	Name <code>kontext</code> einer Kontextdefinition; der Name muß einer der Kontextnamen sein, die im Abschnitt <code>init</code> mit

	dem Schlüssel <code>kontexte</code> definiert wurden, und es muß ein entsprechender Abschnitt <code>kontext.kontext</code> existieren. Ein leerer String darf verwendet werden, um Dateidefinitionen „abzuhängen“.
<code>kontexte</code>	Eine Liste von Name(n) <code>kontext</code> der Kontextdefinitionen <code>kontext.kontext</code> ; mehrere Namen müssen durch Leerzeichen oder Tabs getrennt werden, und es muß mindestens ein Name vorhanden sein.
<code>metadaten</code>	<code>default</code> <code>ja</code> <code>nein</code> Legt fest, ob eine Metadatendatei ausgegeben werden soll. <code>default</code> bedeutet, daß die Ausgabe erfolgt, wenn eine Metadatendatei per INI-Meta definiert wurde. <code>ja</code> bedeutet, daß die Ausgabe unbedingt erfolgen soll, in diesem Fall ist das Fehlen der INI-Meta ein Fehler.
<code>open</code>	<code>output</code> <code>extend</code> [<code>s</code> <code>rotationsdauer</code>] <code>open="extend"</code> bewirkt, daß eine vorhandene Datei erweitert und nicht überschrieben wird, d.h. die Ausgabedatensätze werden an das Ende der Datei angehängt. Der optionale Wert <code>rotationsdauer</code> ermöglicht die Angabe einer Zeitraumes, nach dessen Ablauf die aktuelle Ausgabedatei rotiert, d.h. durch eine neue Instanz ersetzt wird. Dies ist vor allem in Implementierungen sinnvoll, die als Server laufen. <code>rotationsdauer</code> hat die Form: <code>nz</code> , wobei <code>n</code> ein ganzzahliger Wert und <code>z</code> die zu messende Zeiteinheit mit einem der folgenden Werte ist: <code>d</code> (Tage), <code>h</code> (Stunden), <code>m</code> (Minuten) und <code>s</code> (Sekunden. Beispiele: <code>open="extend 1d"</code> : ein Tag <code>open="extend 1h"</code> : eine Stunde <code>open="extend 90m"</code> : 90 Minuten -
<code>satzartmerkmale</code>	Namen der Merkmale, die für die Satzartauswahl aus den Meldungsdatensätzen verwendet werden; mehrere Namen müssen durch Leerzeichen oder Tabs getrennt werden, ein Leerstring bedeutet, daß keine Satzartauswahl stattfindet (jeder Meldungsdatensatz wird allen Datensatzdefinitionen der Datei zugeordnet). HINWEIS: Anstelle eines Merkmals kann auch ein in Frage kommendes Metadatum angegeben werden, z.B. <code>#satz.kennung</code> .
<code>satzartwerte</code>	Werte der Satzartmerkmale; mehrere Werte müssen durch Leerzeichen oder Tabs getrennt werden.
<code>satzformat</code>	<code>default</code> <code>f</code> <code>fest</code> <code>v</code> <code>variabel</code> Legt das Satzformat der Datei fest. Per Default wird dieses von der Anwendung bestimmt: festes Satzformat wird gewählt, wenn es nur eine Datensatzdefinition für eine Datei gibt oder alle die gleiche Satzlänge haben, ansonsten ist das Ergebnis variables Satzformat.
<code>speicherformat</code>	<code>default</code> <code>ebcdic</code> <code>ascii</code> <code>ascii-csv</code>

	<p><code>ascii-binaer</code></p> <p>Bestimmt das externe Speicherformat der Datei analog zu den Möglichkeiten in STATSPEZ. Das Satzende-kennzeichen und betriebssystemspezifische Felder werden automatisch generiert.</p>
--	--

3.11.2. Operanden (Alphabetisch)

Operand	Beschreibung
<code>bzrspanne</code>	<code>bzrstring-bzrstring bzrstring- -bzrstring</code> ; Eine Berichtszeitraumangabe mit einer geschlossenen oder in die Vergangenheit oder Zukunft offenen Spanne, aus ein oder zwei Elementen des Typs <code>bzrstring</code>
<code>bzrstring</code>	Eine Berichtszeitraumangabe wie sie beim Zugriff auf den Berichtszeitraum mit dem Metadatum <code>berichtszeitraum.string</code> erzeugt wird (6.3.3).
<code>datml-name</code>	<code>name</code> ; Ein Name vom Typ <code>name</code>
<code>default</code>	Wie <code>quelle</code> ; definiert den Defaultwert
<code>feldname</code>	<code>splv-name</code> ; Der Name des Einzelfeldes.
<code>funktionsaufruf</code>	<code>funktionsname([param[,param]..])</code> ; der Aufruf einer Funktion; Parameter sind funktionsabhängig.
<code>funktionsname</code>	<code>konst-name</code> ; der Name einer Funktion
<code>index</code>	<code>int_1..n merkmal</code> ; Ein Merkmalsgruppenindex, entweder als direkter Index in Form einer ganzen Zahl größer Null oder als indirekter Index in Form des Namens eines Merkmals der Merkmalsgruppe. Der Wert dieses Merkmals muß wiederum eine ganze positive Zahl sein.
<code>konstante</code>	<code>'chars'</code> ; eine Konstante; Hochkommata sind erforderlich.
<code>konst-name</code>	Ein konstanter, vordefinierter Name
<code>merkmal</code>	<code>datml-name</code> ; der Name eines Merkmals.
<code>merkmalsgruppe</code>	<code>merkmalsgruppe: datml-name'[['index]']'</code> Ein Merkmalsgruppenindex, entweder als direkter Index in Form einer ganzen Zahl größer Null oder als indirekter Index in Form des Namens eines Merkmals der Merkmalsgruppe. Der Wert dieses Merkmals muß wiederum eine ganze positive Zahl sein.
<code>meta</code>	<code>#[metakontext]name</code> ; der Name eines Metadatums.
<code>metakontext</code>	<code>{name.}..</code> ; der Metadatenkontext in dem sich ein Metadatum befindet.
<code>mm_pfad</code>	<code>[/][pfadelement[/]..]{merkmal}</code> Eine Pfadangabe <code>pfad</code> besteht aus einer Folge von ein oder mehr optionalen, durch Schrägstriche getrennten Pfadelementen <code>pfadelement</code> , der ein Schrägstrich vorausgeht, wenn es sich um eine absolute Pfadangabe handelt, und die mit der Angabe eines Merkmals <code>merkmal</code> endet.
<code>mmgr_pfad</code>	<code>[/][pfadelement[/]..]{merkmalsgruppe}</code> Eine Pfadangabe <code>pfad</code> besteht aus einer Folge von ein oder mehr optionalen, durch Schrägstriche getrennten Pfadelementen <code>pfadelement</code> , der ein Schrägstrich vorausgeht, wenn es sich

	um eine absolute Pfadangabe handelt, und die mit der Angabe einer Merkmalsgruppe <i>merkmalsgruppe</i> endet.
<i>pfad</i>	<code>[/][pfadelement[/]..]{merkmalsgruppe merkmal}</code> Eine Pfadangabe <i>pfad</i> besteht aus einer Folge von ein oder mehr optionalen, durch Schrägstriche getrennten Pfadelementen <i>pfadelement</i> , der ein Schrägstrich vorausgeht, wenn es sich um eine absolute Pfadangabe handelt, und die mit der Angabe eines Merkmals <i>merkmal</i> oder einer Merkmalsgruppe <i>merkmalsgruppe</i> endet.
<i>pfadelement</i>	<code>pfadelement: merkmalsgruppe '.' '..'</code> Eine Pfadelement <i>pfadelement</i> adressiert ausgehend von der durch die vorausgehenden Pfadelemente erreichten aktuellen Position entweder eine Merkmalsgruppe <i>merkmalsgruppe</i> , den aktuellen Knoten (.) oder den Elternknoten (..).
<i>name</i>	Eine Folge von Buchstaben und/oder Ziffern, die mit einem Buchstaben oder einer Ziffer beginnt und endet und folgende Sonderzeichen enthalten kann: Unterstrich, Minuszeichen.
<i>quelle</i>	<code>{quelleelement}[[s]+s]quelleelement</code> ..]; Eine Folge von ein oder mehreren Elementen des Typs <i>quelleelement</i> ; je zwei aufeinanderfolgende Elemente werden durch den optional in Leerraum <i>s</i> eingebetteten Verkettungsoperator '+' miteinander verbunden. Das Ergebnis der Auswertung des Operanden <i>quelle</i> ist stets eine einzelne, zusammenhängende Resultatzeichenfolge. Der Resultatstring wird dem Feld als Wert zugewiesen.
<i>quelleelement</i>	<code>{meta mm_pfad konstante funktionsaufruf}</code> ; Ein Metadatum oder ein Merkmal oder eine Konstante oder ein Funktionsaufruf. Merkmale dürfen nur in der INI-Daten verwendet werden.
<i>RE</i>	Ein regulärer Ausdruck; siehe 3.10
<i>s</i>	Leerraum
<i>splv-name</i>	Ein Name gemäß den Regeln für SPLV-Namen lt. SPLV-Sprachbeschreibung.
<i>str</i>	<code>STR [s laenge]</code> ; Eine nicht wiederholte Feldgruppe.
<i>str-laenge</i>	<code>int_1..n</code> ; die Länge der Struktur.
<i>str-anzahl</i>	<code>int_1..n</code> ; die (maximale) Anzahl der Instanzen der Struktur.
<i>string</i>	<code>'chars' chars</code> ; eine in optionale Hochkommata eingeschlossene Zeichenfolge.
<i>strukturname</i>	<code>splv-name</code> ; Der Name der Struktur.
<i>strukturtyp</i>	<code>str wfg vwfg</code> ; eine Struktur, eine wiederholte Feldgruppe oder eine variable wiederholte Feldgruppe gemäß den SPLV-Konventionen.
<i>vwfg</i>	<code>VWFG s anzahl [s laenge]</code> ; Eine wiederholte Feldgruppe mit einer variablen Anzahl von Instanzen.
<i>wfg</i>	<code>WFG s anzahl [s laenge]</code> ; Eine wiederholte Feldgruppe mit einer festen Anzahl von Instanzen.

3.12. Templates für externe Dateinamen

Die Templates für die Generierung externer Dateinamen dürfen eine absolute bzw. relative Verzeichnisnamen sowie variable Bestandteile enthalten. Als variable Bestandteile sind ausgewählte Metadaten und Hilfsmerkmale zulässig. Ein variabler Bestandteil wird durch Einschluß in geschweiften Klammern von den konstanten Teilen des Templates isoliert:

```
'{ 'meta|merkmal' }'.
```

Beispiel mit Metadaten (Zeilenumbruch wg. Platzmangels):

```
ku101.daten.{#absender.kennung}.{#berichtszeitraum.jahr}
ku101.meta.{#erhebung.kennung}.{#berichtszeitraum.jahr}
```

Beispiel mit Hilfsmerkmal `verkehrsrichtung`:

```
intra.daten.{#absender.kennung}.{verkehrsrichtung}
```

Folgende Metadaten sind als variabler Bestandteil im externen Namen einer Daten- oder Metadatendatei erlaubt:

Metadaten in externen Dateinamen		
Kontext	#kontextname	metadatum
Absender	#absender	.bundesland .kennung
	#empfaenger	.bundesland .kennung
	#berichtspflichtiger	.bundesland .kennung
	#berichtsempfaenger	.bundesland .kennung
Statistischer Kontext	#berichtszeitraum	.datum .jahr .monat .tag .halbjahr .quartal .semester .woche
	#erhebung	.kennung .text
	#material	.kennung .name
Dokument	#protokoll.eingang	.datum .zeit
Anwendung	#dokument	.absoluter-name .absoluter-pfad .name-und-typ .name .typ
	#datum	Format: <i>yyyymmdd</i>
	#jahr	Format: <i>yyyy</i>

	#monat	Format: <i>mm</i>
	#tag	Format: <i>dd</i>
	#zeit	Format: <i>hhmmss</i>
	#stunde	Format: <i>hh</i>
	#minute	Format: <i>mm</i>
	#sekunde	Format: <i>ss</i>

Ist kein Wert für einen variablen Bestandteil vorhanden, wird der dem Typ entsprechende Anfangswert verwendet, z.B.

- ein Leerstring, wenn der Typ des Metadatums ein beliebiger String ist,
- eine entsprechende Zahl von Nullen, wenn es sich um eine Datums- oder Zeitangabe handelt und diese nicht oder nur teilweise (nur das Jahr etc.) zur Verfügung steht, z.B. `20030000` für `#datum`.

Hinweis: Bei rotierenden Ausgabedateien (`open="extend rotationsdauer"`) ist es notwendig, die Dateiinstanzen durch geeignete Maßnahmen wie etwa Zeitstempel zu unterscheiden. Wie dies erreicht wird, ist Sache der implementierenden Anwendung. Unabhängig davon kann in einem Template ein Zeitstempel durch Verwendung der Metadaten `#datum` und `#zeit` erzeugt werden.

4. Verarbeitung [*nicht-normativ*]

Die Verarbeitung ist anwendungsabhängig; die folgende Punkte dienen zur Orientierung bei der Anwendungsentwicklung (siehe auch die DatML/RAW-Spezifikation in der jeweiligen Version):

- Aus Sicht des INI-basierten Mappings ist die Einzelmeldung die wichtigste logische Bezugseinheit innerhalb des DatML/RAW-Dokumentes. Alle *meldungsbezogenen* Metadaten des statistischen Kontextes einer Einzelmeldung stehen zur Verfügung, sobald im Segmentbaum einer DatML/RAW-Nachricht der erste Datensatz einer Einzelmeldung erreicht ist.
 - Wegen der möglichen hierarchischen Anordnung von Segmenten sollten die Metadaten nachrichten- und segmentknotenweise auf einen Stack gelegt werden, damit die nachrichten- und segmentspezifischen Metadaten beim Verlassen des Segmentes in Richtung Elternknoten oder beim Übergang in die nächste Nachricht entfernt werden können, da sie ihre Gültigkeit verlieren.
 - Alle *meldungsübergreifenden* Metadaten stehen zur Verfügung, sobald der erste Nachrichtenknoten erreicht ist.
 - Die Zuordnung der Meldung zu einer oder mehreren Ausgabedateien erfolgt wie unter Punkt 2.3.2 beschrieben.
 - Die Zuordnung der Datensätze einer Meldung zu einer oder mehreren Datensatzdefinitionen erfolgt wie unter Punkt 2.3.3 beschrieben.
 - Die Zuordnung eines Merkmals zu einem Einzelfeld erfolgt über den Merkmalsnamen (s. 3.6)
 - Bei Einzelfelddefinitionen Evaluierung des Operanden *quelle* (s. 3.3); das Resultat ist eine ggf. leere Zeichenfolge, der Resultatstring.
 - Bei Strukturanfangsdefinitionen Evaluierung des Operanden *pfad* (s. 3.4); das Resultat ist die entweder Zuordnung einer Merkmalsgruppe oder die Initialisierung aller Instanzen der Struktur mit Anfangswerten.
 - Der Resultatstring wird in das Einzelfeld unter Berücksichtigung dessen Eigenschaften übertragen; ein leerer Resultatstring bewirkt das Setzen des Einzelfeldes auf Anfangswerte.
-

5. Einfaches Beispiel *[nicht-normativ]*

Ein DatML/RAW-Dokument enthält eine Datenlieferung mit zwei Satzarten des Materials KU101. In jedem Datensatz ist die Satzart über das Merkmal `satzart` angegeben, das jeweils nach EF1 übernommen wird (die Satzart könnte alternativ als konstanter Wert eingetragen werden). Das Berichtsjahr wird unverändert nach EF4 übernommen.

Hinweis: Die Felder EF1- EF4 könnten auch als Kopfdatensatz `[KU101.kopf]` definiert sein.

```
[init]
dateien="KU101"
kontexte="KU101"

[KU101.kontext]
erhebung.kennung="'23111'"
erhebung.klasse="'EVAS'"
berichtszeitraum.jahr="'2002'"

[KU101.init]
kontext="KU101"
anzahlSatzdefinitionen="2"
externerNameDaten="kul01.daten.{#datum}{#zeit}"
externerNameMeta="kul01.meta.{#datum}{#zeit}"
satzartmerkmale="satzart"
metadaten="ja"

[KU101.satz.1]
satzartwerte="1"
anzahlFelder="..."
f1="EF1 (NOV 1 K 0) satzart"
f2="EF2 (NOV 4 K 0) kh-nr"
f3="EF3 (NOV 3 K 0) kh-traeger"
f4="EF4 (NOV 4 K 0) #berichtszeitraum.jahr"
f5="EF5 (ALN 200) #berichtspflichtiger.adresse"
u.s.w.

[KU101.satz.2]
satzartwerte="2"
anzahlFelder="..."
f1="EF1 (NOV 1 K 0) satzart"
f2="EF2 (NOV 4 K 0) kh-nr"
f3="EF3 (NOV 3 K 0) kh-traeger"
f4="EF4 (NOV 4 K 0) #berichtszeitraum.jahr"
f5="EF5 (NOV 3 K 0) abteilung"
u.s.w.
```

Die passende INI-Meta ist hier nicht beschrieben.

6. Metadaten nach Bereichen

Ein Metadatum, wie z.B. der Berichtszeitraum oder der Name eines Absenders, wird stets mit einem vordefinierten Namen referenziert. Da bestimmte Typen von Metadaten in verschiedenen Kontexten vorkommen können (z.B. eine Straße als Teil einer Adresse, die wiederum zu einem Berichtspflichtigen oder einem Absender gehören kann), muß die Referenzierung das Metadatum qualifizieren, damit der Kontext und damit das Metadatum eindeutig sind. Der so spezifizierte Kontext wird als *Metadatenkontext* bezeichnet (nicht zu verwechseln mit dem *statistischen Kontext* der Meldung). Der Metadatenkontext wird als Folge von Teilnamen spezifiziert, die jeder mit einem Punkt enden:

Operand	Beschreibung
<i>meta</i>	# <i>[metakontext]</i> <i>name</i> ; der Name eines Metadatum
<i>metakontext</i>	{ <i>name.</i> } <i>..</i> ; der ein- oder mehrteilige Name des Metadatenkontextes in dem sich ein Metadatum befindet.

Metadaten mit einem immer eindeutigem oder implizitem Metadatenkontext müssen nicht qualifiziert referenziert werden.

Beispiel:

```
f1="EF1 (ALN 8) #berichtszeitraum
f2="EF2 (ALN 20) #material.name
f3="EF3 (ALN 60) #berichtspflichtiger.nname
f4="EF4 (ALN 60) #berichtspflichtiger.kontakt.kommunikation
```

Die folgenden Tabellen zeigen die zulässigen Metadatenkontexte und daran gebundene Metadaten in der Unterteilung nach ihrer Sichtbarkeit; die Kriterien hierfür sind die Dokumentebenen bzw. die Anwendung. Für einige Metadatenkontexte gibt es ein Default-Metadatum; dieses ist in der Liste der Metadaten jeweils durch Unterstreichung gekennzeichnet.

6.1. Metadaten auf Dokumentebene

Die folgenden Metadatenkontexte sind im statischen Teil eines DatML/RAW-Dokumentes enthalten, also vor der ersten Nachricht.

6.1.1. Metadatenkontext **absender.***

#kontextname	metadatum
#absender	adresse anrede berechtigung bundesland hausnummer ¹⁾ email ¹⁾ fax ¹⁾ kennung klasse kommunikation ¹⁾ kreis

	land memo nachname niederlassung <u>nname</u> organisation ort postfach postfachleitzahl postfachort postleitzahl strasse telefon ¹⁾ titel url ¹⁾ vname vorname zeilen
#absender.kontakt	adresse ¹⁾ anrede bundesland ¹⁾ hausnummer ¹⁾ email fax <u>kommunikation</u> kreis ¹⁾ land ¹⁾ nachname niederlassung nname organisation ort ¹⁾ postfach ¹⁾ postfachleitzahl ¹⁾ postfachort ¹⁾ postleitzahl ¹⁾ strasse ¹⁾ telefon titel url vname vorname zeilen
#absender.korrektur	wie #absender ¹⁾
#absender.korrektur.kontakt	wie #absender.kontakt

¹⁾ Diese Metadatenkontexte bzw. Metadaten werden ab DatML/RAW 2.0 unterstützt.

6.1.2. Metadatenkontext `empfaenger.*`

#kontextname	Metadatum
--------------	-----------

¹⁾ Mit Ausnahme der Metadaten `berechtigung` und `memo`

#empfaenger	wie #absender
#empfaenger.kontakt	wie #absender.kontakt

6.1.3. Metadatenkontext `protokoll.*`

#kontextname	Metadatum
#protokoll.eingang	anwendung dateiname ²⁾ datentraegertyp ²⁾ datentraeger-id ²⁾ datum eingangsstempel ¹⁾ fragebogen ²⁾ hersteller ¹⁾ server ¹⁾ lieferart ¹⁾ lieferformattyp ²⁾ lieferformatname ²⁾ liefermedium ²⁾ version ¹⁾ zeit zertifikat ¹⁾
#protokoll.eingang.beleg ¹⁾	formular-id ¹⁾ formularname ¹⁾
#protokoll.eingang.datei ¹⁾	anwendung dateiname ¹⁾ datenformat ¹⁾ datenformattyp ¹⁾ datentraeger datentraegertyp datum dokumentid ¹⁾ hersteller ¹⁾ server ¹⁾ upload ¹⁾ version ¹⁾ zeit zertifikat ¹⁾
#protokoll.eingang.online ¹⁾	formular-id ¹⁾ formularname ¹⁾
#protokoll.instanz	anwendung dateiname ¹⁾ datum dokumentid ¹⁾ hersteller ¹⁾ server ¹⁾ version ¹⁾ zeit zertifikat ¹⁾

¹⁾ Diese Metadatenkontexte bzw. Metadaten werden ab DatML/RAW 2.0 unterstützt.

²⁾ Diese Metadaten werden nur noch aus Kompatibilitätsgründen unterstützt. Es sollten die entsprechenden Metadaten aus den Kontexten `protokoll.eingang.online`, `protokoll.eingang.beleg` und `protokoll.eingang.datei`, verwendet werden.

Die Metadaten `lieferart` und `upload` müssen von der Anwendung versorgt werden; die Werte sind:

Metadatum	Wert
<code>lieferart</code>	1 = Online-Formular 2 = Papierformular (Beleg) 3 = Lieferdatei
<code>upload</code>	0 = Falsch (Nein) 1 = Wahr (Ja)

In DatML/RAW 2.0 wurden die Strukturen für die Speicherung von Protokolldaten völlig überarbeitet. Zwischen den neu eingeführten Metadaten und den in den Versionen 1.0.x unterstützen bestehen Übereinstimmungen und Zusammenhänge.

Die Tabelle zeigt die jeweils korrespondierenden Metadaten der Versionen 1.0.x und 2.0. In den Bemerkungen ist angegeben, ob der Wert aus einem DatML/RAW-Dokument der Version 2.0 übernommen werden kann oder ob es notwendig ist, das korrespondierende Metadatum der Version 1.0.x auf einen spezifischen Wert zu setzen.

Metadatum 1.0.x	Metadatum 2.0	Bemerkung
<code>dateiname</code>	<code>datei.dateiname</code>	Wert übernehmen
<code>datentraegertyp</code>	<code>datei.datentraegertyp</code>	Wert übernehmen
<code>datentraeger-id</code>	<code>datei.datentraeger</code>	Wert übernehmen
<code>fragenbogen</code>	<code>beleg.formular-id</code> <code>online.formular-id</code>	Wert übernehmen
<code>lieferformatname</code>	<code>datei.datenformat</code>	Wert übernehmen
<code>lieferformattyp</code>	<code>datei.datenformattyp</code>	Ist für Online-Formulare auf <code>html</code> zu setzen.
<code>liefermedium</code>	<code>lieferart</code>	Ist für Online-Formulare auf <code>internet</code> , sonst auf <code>datentraeger</code> zu setzen.

6.2. Metadaten auf Nachrichtenebene

6.2.1. Metadatenkontext `nachricht.*`

#kontextname	Metadatum
<code>#nachricht</code> ¹⁾	<code>id</code> ¹⁾

¹⁾ Diese Metadatenkontexte bzw. Metadaten werden ab DatML/RAW 2.0 unterstützt.

Die ID der aktuellen Nachricht und Datum und Uhrzeit ihrer Erzeugung, soweit vorhanden.

6.3. Metadaten auf Meldungsebene

Die folgenden Metadatenkontexte existieren auf Meldungsebene, d.h., sie können für jede Meldung andere Werte annehmen .

6.3.1. Metadatenkontext `berichtsempfaenger.*`

#kontextname	Metadatum
#berichtsempfaenger	wie #empfaenger

6.3.2. Metadatenkontext `berichtspflichtiger.*`

#kontextname	Metadatum
#berichtspflichtiger	wie #absender

6.3.3. Metadatenkontext `berichtszeitraum.*`

#kontextname	Metadatum
#berichtszeitraum	bzr datum jahr monat semester ¹⁾ string tag halbjahr quartal woche

¹⁾ Diese Metadatenkontexte bzw. Metadaten werden ab DatML/RAW 2.0 unterstützt.

Einige Metadaten aggregieren verschiedene Elemente der Berichtszeitraumangabe:

- `bzr` liefert alle vorhandenen Komponenten des Berichtszeitraumes als zusammenhängenden String *ohne* Formatierungszeichen
- `datum` liefert das aus den Berichtszeitraum zusammengesetzte Datum in der Form `yyyy[mm[dd]]`.
- `string` liefert alle vorhandenen Komponenten des Berichtszeitraumes als zusammenhängenden String *mit* Formatierungszeichen in der Form:

```
yyyy[Hh|Qq|Ss|Www|mm[dd]]
```

Das Jahr wird immer und vierstellig erzeugt, es folgt entweder ein Halbjahr `Hh`

oder ein Quartal *Qq* oder ein Semester *Ss* oder eine Woche *Www* oder der Monat *mm* mit oder ohne Tag *dd*. Beispiele:

200212	Dezember 2002
20020926	26. September 2002
2002H1	1. Halbjahr 2002
2002Q4	Viertes Quartal 2002
2002S2	Wintersemester 2002
2002W26	26. Woche 2002

6.3.4. Metadatenkontext *erhebung.**

#kontextname	Metadatum
#erhebung	<u>kennung</u> klasse text

6.3.5. Metadatenkontext *material.**

#kontextname	Metadatum
#material	<u>kennung</u> klasse name

6.3.6. Metadatenkontext *meldung.**

#kontextname	Metadatum
#meldung ¹⁾	<u>id</u> ¹⁾

¹⁾Diese Metadatenkontexte bzw. Metadaten werden ab DatML/RAW 2.0 unterstützt.

Die ID der aktuellen Meldung und Datum und Uhrzeit ihrer Erzeugung, soweit vorhanden.

6.4. Metadaten auf Satzebene

Auf Ebene des Meldungsdatensatzes ist der Zugriff auf das Attribut *kennung* des Elementes *<satz>* möglich.

6.4.1. Metadatenkontext *satz.**

#kontextname	metadatum
#satz	<u>kennung</u> ; Attribut <i>kennung</i> .

6.5. Metadaten auf Merkmalsebene

Auf Merkmalsebene kann nur auf die Metadaten expliziter Nichtantworten zugegriffen werden.

6.5.1. Metadatenkontext `nichtantwort.*`

Das DatML/RAW-Element `<na>` erlaubt es, ausdrücklich anzugeben, daß ein Merkmal bewußt nicht mit einem Wert versorgt wird. Der Grund einer solchen Nichtantwort kann als Elementinhalt abgelegt werden.

#kontextname	metadatum
#nichtantwort ¹⁾	text ¹⁾ ; Textinhalt des Elementes <code><na></code>

¹⁾ Diese Metadatenkontexte bzw. Metadaten werden ab DatML/RAW 2.0 unterstützt.

6.6. Metadaten auf Anwendungsebene

Die Anwendung stellt ebenfalls Metadaten bereit, im wesentlichen handelt es sich um Protokoll- und Hilfsdaten.

6.6.1. Allgemeine Metadaten

#kontextname	metadatum
#anwendung	Name der erzeugenden Anwendung
#anzFehlerfreieSaetzeMeldung	Satzzähler je Meldung
#anzFehlerhafteSaetzeMeldung	
#anzGeleseneSaetzeMeldung	
#anzFehlerfreieSaetzeInsgesamt	Satzzähler insgesamt (alle ausgegebenen Meldungen)
#anzFehlerhafteSaetzeInsgesamt	
#anzGeleseneSaetzeInsgesamt	
#datum	Aktuelles Datum; Format: <code>yyyymmdd</code>
#meldungsnummer	Laufende Meldungsnummer (Ausgabe)
#version	Version der erzeugenden Anwendung
#startDatum	Datum, an dem die Anwendung gestartet wurde; Format: <code>yyyymmdd</code>
#startZeit	Uhrzeit, zu der die Anwendung gestartet wurde; Format: <code>hhmmss</code>
#zeit	Aktuelle Uhrzeit; Format: <code>hhmmss</code>

6.6.2. Metadatenkontext `dokument.*`

Die Anwendung stellt in diesem Metadatenkontext Informationen über das aktuelle Eingabedokument bereit..

#kontextname	metadatum
#dokument	.absoluter-name .absoluter-pfad .name-und-typ .name .typ

Beispiel:

Dokument /dateneingang/intra/2005-01/br10000.200501.xml:

```
absoluter-name = /dateneingang/intra/2005-01/br10000.xml
absoluter-pfad = /dateneingang/intra/2005-01
name-und-typ  = br10000.200501.xml
name          = br10000.200501
typ           = xml
```

6.7. Metadatum zeilen

In den DatML/RAW-Elementtypen `<identifikation>` und `<kontakt>` ist es zulässig, anstelle der spezifischen Elementtypen ein zeilenbasiertes Inhaltsmodell zu verwenden. Dieses besteht aus einer Folge von Elementen vom Typ `<zeile>` mit dem Attribut `zeilennummer`. Das Metadatum `zeilen` referenziert ein Element vom Typ `<zeile>` mit dem Attribut `zeilennummer` und dem Attributwert `n`, wobei gilt $1 \leq n \leq 16$. Beispiele:

`#absender.zeile1` referenziert das Element:

```
<absender><identifikation><zeile zeilennummer="1">...
```

`#absender.kontakt.zeile3` referenziert das Element:

```
<absender><kontakt><zeile zeilennummer="1">...
```

6.8. Aggregierte Metadaten

Ein aggregiertes Metadatum repräsentiert einen Satz zusammengehörender Metadaten, z.B. eine Adresse, die aus Straße, Ort usw. besteht. Teilweise gibt es eine direkte Entsprechung zu DatML/RAW-Elementen wie `<adresse>`, teilweise handelt es sich um abgeleitete Metadaten, z.B. `nname`.

metadatum	Beschreibung
<code>.adresse</code>	Alle Komponenten von <code><adresse></code> außer <code><zusatz></code> , jeweils durch Komma getrennt.
<code>.bzt</code> <code>.datum</code> <code>.string</code>	Komponenten des Berichtszeitraumes (<code><berichtszeitraum></code>), s. 6.3.3
<code>.kommunikation</code>	<code><telefon></code> , <code><email></code> , <code><fax></code> und <code><url></code> , jeweils durch Komma getrennt.
<code>.nname</code>	<code><nachname></code> und <code><vorname></code> , in dieser Reihenfolge und durch ein <i>Komma</i> getrennt.
<code>.vname</code>	<code><vorname></code> und <code><nachname></code> , in dieser Reihenfolge und durch ein <i>Leerzeichen</i> getrennt.

Beispiele:

```
#absender.adresse  
#berichtspflichtiger.nname
```

7. STATSPEZ-Unterstützung [*nicht-normativ*]

Die Hauptfunktion der INI-Dateien ist die Beschreibung der Ausgabedatensätze und der auf diese abzubildenden Daten und Metadaten. Eine INI-Datei ist daher im Kern eine um zusätzliche Angaben und Funktionen erweiterte Datensatzbeschreibung (DSB).

Eine Änderung in einer DSB erzwingt meist auch eine Anpassung der INI-Dateien. Es liegt daher nahe, DSBs zusätzliche Informationen beizufügen, die die Generierung einer INI-Datei ohne manuelle Nachbearbeitung erlauben, idealerweise in einer anwendungsneutralen Form, da DSBs im statistischen Programmierverbund mit verschiedenen Anwendungen erzeugt und gepflegt werden. Die wichtigste dieser Anwendungen ist STATSPEZ, und sie kann DSBs in einem INI-basierten Format exportieren.

Diese Spezifikation beschreibt im folgenden eine Methode, Zusatzinformationen für die vollständige automatische Generierung von INI-Dateien in einer DSB abzulegen, und beschreibt, wie solche DSBs aus STATSPEZ heraus exportiert und mit dem vorhandenen Umsetzer Dsb2Ini in eine INI-Datei konvertiert werden können.

7.1. Voraussetzungen

Die Lösung nutzt die Möglichkeit, DSBs mit Kommentaren zu versehen. Sie besteht aus wenigen syntaktischen und semantischen Regeln, die die Einbettung spezieller Anweisungen in die Kommentierung einer DSB erlauben. Eine solche DSB kann in STATSPEZ erzeugt oder z.B. im DSB-Format in eine STATSPEZ-Anwendung integriert werden. Die Generierung der INI-Datei setzt dann voraus, daß die DSB im STATSPEZ-Format aus STATSPEZ exportiert wird. Aus der exportierten DSB kann wahlweise eine INI-Daten oder eine INI-Meta erzeugt werden.

Für die Generierung einer INI-Datei werden schließlich benötigt:

- Eine aus STATSPEZ im STATSPEZ-Format exportierte DSB, die möglichst alle erforderlichen Zusatzinformationen enthält (s.u.);
- Der Umsetzer Dsb2Ini, eine Java-Applikation, die als `dsb2ini.jar` mit voreingestellter Main-Klasse bereitgestellt wird, sowie eine geeignete Java-Laufzeitumgebung, vorzugsweise JRE 1.4.0 oder höher.

7.2. Vorgehensweise

Die notwendigen Schritte hängen davon ab, ob bereits eine passende DSB in STATSPEZ vorhanden ist oder importiert werden kann, und ob diese ggf. bereits alle Anweisungen enthält (s.u.):

1. DSB des zu erzeugenden Datenmaterials in STATSPEZ importieren oder erstellen.
 2. Anweisungen in die Kommentierung eintragen.
 3. DSB im Format STATSPEZ exportieren.
 4. Dsb2Ini in `dsb2ini.jar` mit der exportierten DSB als Eingabe aufrufen.
-

7.3. Anweisungen

Anweisungen liefern Steuerinformationen für die Generierung der INI-Datei selbst und solche, die nur weitergereicht und erst bei der Verarbeitung eines DatML/RAW-Dokumentes ausgewertet werden. Außerdem geben sie die Quelle an, aus der ein Einzelfeld gefüllt wird.

7.3.1. Syntax

Anweisungen haben folgende Syntax:

```
<[s]anweisungskopf[s]:[s]anweisungskoerper[s]>
```

Der Anweisungskopf enthält den Namen der Anweisung, der Anweisungskörper die für die Ausführung der Anweisung erforderlichen Daten. Beide sind durch einen Doppelpunkt getrennt, und die gesamte Anweisung ist in spitze Klammern eingeschlossen. Die Bestandteile können durch optionalen Leerraum getrennt sein.

Der Anweisungskörper kann Schlüssel-Wert-Paare enthalten; der Wert wird ohne *zusätzliche* Begrenzungszeichen angegeben, führender und folgender Leerraum wird allerdings entfernt, in den Werte enthaltene Begrenzungszeichen bleiben jedoch erhalten.

Bei Anweisungsnamen ist die Groß- und Kleinschreibung zu beachten; für Schlüssel und Werte aus den Ini-Dateien gelten deren Regeln (s. 3)

7.3.1.1. Operanden

Operand	Beschreibung
<i>s</i>	Leerraum.
<i>anweisungskopf</i>	<i>name</i> ; der Name der Anweisung.
<i>anweisungskoerper</i>	Die Syntax des Anweisungskörpers ist anweisungsspezifisch. In der Summe der Anweisungen ist möglich: <i>schluessel-wert</i> : (s.u.). <i>quelle</i> (s. 3.3.1.3)
<i>schluessel-wert</i>	<i>[s]schluessel[s]=[s]wert[s]</i>

7.3.2. Gültigkeitsbereich

Anweisungen haben einen Gültigkeitsbereich: DSB, Datensatz oder Feld. Der Gültigkeitsbereich legt fest, wo in einer DSB eine Anweisung platziert werden kann:

Gültigkeitsbereich	Angabeort	
	DSB <i>ohne</i> Satzarten	DSB <i>mit</i> Satzarten
DSB	In den Kommentarzeilen, die der ersten Felddeklaration vorausgehen	In den Kommentarzeilen des <i>Kopfdatensatzes</i> , die der ersten Felddeklaration vorausgehen

Datensatz	wie DSB	In den Kommentarzeilen des <i>Satzart</i> datensatzes, die der ersten Felddeklaration vorausgehen
Feld	In der Deklarationszeile oder den unmittelbar folgenden Kommentarzeilen.	In der Deklarationszeile oder den unmittelbar folgenden Kommentarzeilen.

7.3.3. Anweisung `init`

Name	<code>init</code>
Gültigkeitsbereich	DSB
Funktion	Liefert ausgewählte Schlüssel für die Abschnitte <code>meta.init</code> (s. 3.1.1) bzw. <code>datei.init</code> (s. 3.2.2)
Anweisungskörper	Ein Schlüssel-Wert-Paar aus einem der folgenden Schlüssel, die im Abschnitt <code>init</code> erlaubt sind: <ul style="list-style-type: none"> • <code>speicherformat</code> • <code>externerNameDaten</code> • <code>externerNameMeta</code> • <code>feldtrennzeichen</code> • <code>dezimalzeichen</code> • <code>open</code>
Beispiele	<code><init:speicherformat=ebcdic></code> <code><init:open=extend></code>

Die Schlüssel `externerNamedaten` und `externerNameMeta` werden nur bei Generierung der INI-Daten ausgewertet.

7.3.4. Anweisung `kontext`

Name	<code>kontext</code>
Gültigkeitsbereich	DSB
Funktion	Liefert Schlüssel für den Abschnitt <code>kontext</code> (s. 3.2.5)
Anweisungskörper	Ein Schlüssel-Wert-Paar das im Abschnitt <code>kontext</code> erlaubt ist.
Beispiele	<code><kontext:erhebung.kennung=23111></code> <code><kontext:erhebung.klasse=EVAS></code>

Die Anweisung `kontext` wird nur bei Generierung der INI-Daten ausgewertet.

7.3.5. Anweisung `kopf`

Name	<code>kopf</code>
Gültigkeitsbereich	Datensatz
Funktion	Liefert Belegungen für Felder des Kopfdatensatzs
Anweisungskörper	Ein Schlüssel-Wert-Paar mit folgendem Format (s. 3.3) <code><i>feld</i>=<i>quelle</i></code>

Beispiele	<code><kopf:EF3='1'></code> <code><kopf:EF2=#erhebung.kennung></code>
-----------	--

Der Zweck der Anweisung `kopf` ist die satzartabhängige Belegung von Feldern des Kopfdatensatzes, beispielsweise um diesen mit dem Satzartwert zu versorgen.

7.3.6. Anweisung `satz`

Name	<code>satz</code>
Gültigkeitsbereich	Datensatz
Funktion	Liefert ausgewählte <code>satz</code> - bzw. <code>satzartbezogene</code> Schlüssel der Abschnitte <code>meta.satz.n</code> (3.1.3) bzw. <code>datei.satz.n</code> (3.2.4).
Anweisungskörper	Der Schlüssel <code>ausgabe</code> für den Abschnitt <code>[meta.satz.n]</code> bzw. <code>datei.satz.n</code> Der Schlüssel <code>satzartwerte</code> für den Abschnitt <code>[datei.satz.n]</code>
Beispiel	<code><satz:ausgabe=close-rep></code> <code><satz:ausgabe=merkmal::satzart='E'></code> <code><satz:satzartwerte=E></code>

7.3.7. Anweisung `xml`

Name	<code>xml</code>
Gültigkeitsbereich	Feld
Funktion	Liefert den Operanden <code>quelle</code> für eine Einzelfelddefinition (s. 3.3.1.3) bzw. den Operanden <code>pfad</code> für eine Strukturdefinition (s. 3.4.1.3).
Anweisungskörper	Ein Operand vom Typ <code>quelle</code> oder <code>pfad</code>
Beispiele	<code><xml:UmsatzVorSteuern></code> <code><xml:?verkehrsrichtung></code> <code><xml:#absender.adresse></code> <code><xml:'9999'></code> <code><xml:monatsumsatz[]></code>

Die Anweisung `xml` ist in Einzelfeldern und Feldgruppen erlaubt.

Bei Feldgruppen ist zu beachten:

- Der Operand der `xml`-Anweisung muß vom Typ `pfad` sein.
- Die Strukturendefinition wird automatisch generiert.

Bei Merkmalen gilt eine Erweiterung, die der Unterstützung externer Prozesse dient:

- Hilfsmerkmale können als solche gekennzeichnet werden, indem dem Merkmalsnamen unmittelbar ein Fragezeichen (?) vorangestellt wird.

- Für das Mapping ist das Fragezeichen zur Zeit noch irrelevant; es wird jedoch empfohlen, bei Hilfsmerkmalen von dieser Kennzeichnungsmöglichkeit Gebrauch zu machen.

7.4. Aufruf von Dsb2Ini

Die Anwendung Dsb2Ini benötigt drei Parameter:

Parameter #	Bedeutung
1	<i>meta daten</i> Steuert, ob eine INI-Meta oder eine INI-Daten erzeugt wird.
2	<i>dsb-eingabedatei</i> Die DSB im Format STATSPEZ
3	<i>ini-ausgabedatei</i> Die zu erzeugende INI-Datei

Beispiel für den Aufruf aus einer DOS-Shell:

```
java -jar dsb2ini.jar daten ku105-daten.dsb ku105-daten.ini  
java -jar dsb2ini.jar meta ku105-meta.dsb ku105-meta.ini
```

7.5. Organisatorisches

Diese Beschreibung, Beispiele, der Umsetzer Dsb2Ini und weitere Informationen sind zunächst über den Autor erhältlich, zukünftig auch über das Internet unter

www.statspez.de

8. Änderungen seit Version 1.1

8.1. Kompatible Änderungen

#	Änderung	
A1-1.2	Art	Neue Funktionen <code>collapse()</code> und <code>normalize()</code>
	Details	Siehe Abschnitt 3.9.1
A2-1.2	Art	Neue Metadaten <code>berichtszeitraum.bzr</code> , <code>berichtszeitraum.spanne</code> und <code>berichtszeitraum.string</code>
	Details	Siehe 6.3.3
A3-1.2	Art	Neue Schlüssel <code>berichtszeitraum.bzr</code> , <code>berichtszeitraum.spanne</code> und <code>berichtszeitraum.string</code> in Abschnitten des Typs <code>kontext.kontext</code>
	Details	Siehe 6.3.3, 3.2.5
A4-1.2	Art	Abfrage von Berichtszeitraumspannen
	Details	Siehe 3.2.5.1
A5-1.2	Art	Erweiterte Template für externe Dateinamen
	Details	Siehe 3.12

9. Einschränkungen der Version 1.2

Das INI-basierte Mapping unterstützt mit Ausnahme einiger wenig gebräuchlicher Strukturen DatML/RAW bis *einschließlich* Version 2.0.

#	Einschränkung	
E1-1.1	Komponente	Element <code><memo></code>
	Details	Das Element wird nur unterstützt, wo ausdrücklich angegeben.
E2-1.1	Komponente	Berichtszeitraum
	Details	Uhrzeitangaben (Element <code><string></code> , Formatklasse <code>datum</code>) werden z.Z. nicht ausgewertet.

10. Anhang *[nicht-normativ]*

10.1. Beispiele

10.1.1. INI-Daten

Es handelt sich um eine mit Dsb2Ini (siehe Kapitel 7) generierte INI-Daten für die Erzeugung von Erhebungsdatensätzen der Intrahandelsstatistik (aus Formatierungsgründen nacheditiert).

```
*
* *** Dies ist eine automatisch generierte Datei ***
*
*           Erzeugt am : 03.11.2004 11:17:26
* Erzeugt von Anwendung : Dsb2Ini, Version 1.1
*
* *** Ausgewählte organisatorische Angaben der DSB ***
*
*           Freigabestatus : k.A.
*           ASP-Name      :
*           Stand         : Oktober 2004
*           Bearbeiter    : Michael Schäfer, IIC4
*           Land          :
*
* *** WICHTIGE HINWEISE ***
* =====
*
* Vor Verwendung dieser Datei müssen ggf .folgende Schritte ausgeführt
* werden, um die Vollständigkeit der Datei festzustellen bzw. herzustellen:
*
* 1) Falls verschiedene Satzarten geliefert werden:
*   - Namen der Satzartmerkmale prüfen und ggf. eintragen:
*   - Abschnitt: [DSB-IN#SGXML.2005.init]
*   - Schlüssel: satzartmerkmale
*   - Satzartwert(e) bei den Datensatzbeschreibungen prüfen und
*     ggf. eintragen
*   - Abschnitte: [DSB-IN#SGXML.2005.satz.<nummer>]
*   - Schlüssel: satzartwerte
*
* 2) Bei Feldern, die mit konstanten Werten ungleich Anfangswert versorgt
* werden, die jeweiligen Werte eintragen:
*   - Beispiel: Die Satzart wird nicht aus einem Merkmal übernommen,
*     sondern per Konstante (das Satzartfeld darf in diesem Fall
*     NICHT im Kopf-Asp, sondern MUSS im Satzart-Asp deklariert sein!)
*
* 3) Ggf. die Namen der Daten- und Metadatenfile anpassen:
*   - Abschnitt: [DSB-IN#SGXML.2005.init]
*   - Schlüssel: externerNameDaten, externerNameMeta
*
* 4) Den Abschnitt[DSB-IN#SGXML.2005.kontext] nachbearbeiten:
*   - relevante Schlüssel entkommentieren
*   - Werte eintragen
*
* [init]
* dateien="DSB-IN#SGXML.2005"
* kontexte="DSB-IN#SGXML.2005"
*
* [DSB-IN#SGXML.2005.init]
* kontext="DSB-IN#SGXML.2005"
* anzahlSatzdefinitionen="2"
* externerNameDaten="intra402.txt"
* externerNameMeta="DSB-IN#SGXML.2005.meta"
* satzartmerkmale="Verkehrsrichtung"
```

```
satzformat="fest"
speicherformat="ascii"
feldtrennzeichen=";"
dezimalzeichen=","
metadaten="default"
open="output"

[DSB-IN#SGXML.2005.kontext]
* ---Beginn generierte Schlüssel---
erhebung.kennung="511([12]1)?"
* ---Ende generierte Schlüssel---

* Bitte ggf. die fuer den Meldungskontext relevanten Schlüssel entkommen-
* tieren und mit Werten versehen soweit nicht bereits generiert (s.o.!)
* - Beispiel
*   erhebung.kennung="23111"
*   erhebung.klasse="EVAS"
*   berichtszeitraum.jahr="2003"
*erhebung.kennung="hier_evas_nr_eintragen"
*erhebung.klasse="EVAS"
*erhebung.text="hier_text_eintragen"
*berichtszeitraum.datum="jjjjmmtt"
*berichtszeitraum.jahr="jjjj"
*berichtszeitraum.monat="mmm"
*berichtszeitraum.tag="tt"
*berichtszeitraum.halbjahr="1..2"
*berichtszeitraum.quartal="1..4"
*berichtszeitraum.woche="1..53"

[DSB-IN#SGXML.2005.satz.1]
satzartwerte="E"
datensatzname="EINGANG"
anzahlFelder="39"
f1="EF1 (NOV 001 K 00) '1'"
f2="EF2 (NOV 001 K 00) '1'"
f3="EF3U1 (NOV 002 K 00) Anmeldemonat"
f4="EF3U2 (NOV 002 K 00) '00'"
f5="EF3U3 (NOV 006 K 00) LaufendeNummer"
f6="EF4U1 (ALN 002) "
f7="EF4U2 (NOV 002 K 00) split(#absender.kennung, '.',1)"
f8="EF4U3 (NOV 011 K 00) split(#absender.kennung, '.',2)"
f9="EF4U4 (NOV 003 K 00) split(#absender.kennung, '.',3)"
f10="EF5 (ALN 001) "
f11="EF6 (ALN 003) Versendungsland"
f12="EF7 (NOV 002 K 00) Bestimmungsregion"
f13="EF8 (ALN 002) "
f14="EF9 (NOV 002 K 00) Geschaeftsart1+Geschaeftsart2"
f15="EF10 (NOV 001 K 00) Verkehrszweig"
f16="EF11 (ALN 001) "
f17="EF12 (ALN 006) "
f18="EF13 (ALN 006) "
f19="EF14 (NOV 008 K 00) Warennummer"
f20="EF15 (ALN 003) Ursprungsland"
f21="EF16 (NOV 005 K 00) Verfahren"
f22="EF17 (NOV 011 K 00) Eigenmasse"
f23="EF18 (NOV 011 K 00) BesondereME"
f24="EF19 (ALN 002) "
f25="EF20 (NOV 011 K 00) Rechnungsbetrag"
f26="EF21 (NOV 011 K 00) StatistischerWert"
f27="EF22U1 (NOV 002 K 00) #berichtszeitraum.monat | Anmeldemonat"
f28="EF22U2 (NOV 002 K 00) substring(#berichtszeitraum.jahr,-2)"
f29="EF23 (NOV 001 K 00) '2'"
f30="EF24 (ALN 002) "
f31="EF25UG1 (ALN 90) firma | #berichtspflichtiger.organisation"
f32="EF25U4 (ALN 030) strasse | #berichtspflichtiger.strasse"
f33="EF25U5 (ALN 006) plz | #berichtspflichtiger.postleitzahl"
f34="EF25U6 (ALN 030) ort | #berichtspflichtiger.ort"
f35="EF26 (ALN 105) Warenbezeichnung"
f36="EF27U1 (ALN 002) "
```

```
f37="EF27U2 (ALN 002) land | split(#berichtspflichtiger.kennung, '.',1)"
f38="EF27U3 (ALN 011) steuernummer | split(#berichtspflichtiger.kennung, '.',2)"
f39="EF27U4 (ALN 003) zusatz | split(#berichtspflichtiger.kennung, '.',3)"

[DSB-IN#SGXML.2005.satz.2]
satzartwerte="V"
datensatzname="VERSAND"
anzahlFelder="39"
f1="EF1 (NOV 001 K 00) '2'"
f2="EF2 (NOV 001 K 00) '2'"
f3="EF3 (ALN 001) "
f4="EF4U1 (NOV 002 K 00) Anmeldemonat"
f5="EF4U2 (NOV 002 K 00) '00'"
f6="EF4U3 (NOV 006 K 00) LaufendeNummer"
f7="EF5U1 (ALN 002) "
f8="EF5U2 (NOV 002 K 00) split(#absender.kennung, '.',1)"
f9="EF5U3 (NOV 011 K 00) split(#absender.kennung, '.',2)"
f10="EF5U4 (NOV 003 K 00) split(#absender.kennung, '.',3)"
f11="EF6 (ALN 003) Bestimmungsland"
f12="EF7 (NOV 002 K 00) Ursprungsregion"
f13="EF8 (NOV 002 K 00) Geschaeftsart1+Geschaeftsart2"
f14="EF9 (NOV 001 K 00) Verkehrszweig"
f15="EF10 (ALN 001) "
f16="EF11 (ALN 006) "
f17="EF12 (ALN 006) "
f18="EF13 (NOV 008 K 00) Warennummer"
f19="EF14 (ALN 003) "
f20="EF15 (ALN 002) "
f21="EF16 (NOV 005 K 00) Verfahren"
f22="EF17 (NOV 011 K 00) Eigenmasse"
f23="EF18 (NOV 011 K 00) BesondereME"
f24="EF19 (ALN 002) "
f25="EF20 (NOV 011 K 00) Rechnungsbetrag"
f26="EF21 (NOV 011 K 00) StatistischerWert"
f27="EF22U1 (NOV 002 K 00) #berichtszeitraum.monat | Anmeldemonat"
f28="EF22U2 (NOV 002 K 00) substring(#berichtszeitraum.jahr,-2)"
f29="EF23 (NOV 001 K 00) '2'"
f30="EF24 (ALN 002) "
f31="EF25UG1 (ALN 90) firma | #berichtspflichtiger.organisation"
f32="EF25U4 (ALN 030) strasse | #berichtspflichtiger.strasse"
f33="EF25U5 (ALN 006) plz | #berichtspflichtiger.postleitzahl"
f34="EF25U6 (ALN 030) ort | #berichtspflichtiger.ort"
f35="EF26 (ALN 105) Warenbezeichnung"
f36="EF27U1 (ALN 002) "
f37="EF27U2 (ALN 002) land | split(#berichtspflichtiger.kennung, '.',1)"
f38="EF27U3 (ALN 011) steuernummer | split(#berichtspflichtiger.kennung, '.',2)"
f39="EF27U4 (ALN 003) zusatz | split(#berichtspflichtiger.kennung, '.',3)"
```

10.2. Mapping der Metadaten auf DatML/RAW-Elemente

Die folgenden Tabellen geben für jedes Metadatum den korrespondierenden Knoten in einem DatML/RAW-Dokument an, und zwar als Xpath-Ausdruck, der auf ein Element oder ein Attribut zeigt.

Die Tabellen sind alphabetisch aufsteigend nach Metadatenkontexten geordnet und für jeden ersten Teilnamen eines Metadatenkontextes in einem eigenen Abschnitt enthalten.

Nicht enthalten sind aggregierte Metadaten (6.8); für diese gilt das Mapping der jeweiligen Einzelmetadaten.

10.2.1. Mapping für DatML/RAW 1.0.x

10.2.1.1. Metadatenkontext `absender.*`

#absender	Xpath relativ zu <code>absender</code>
anrede	<code>identifikation/identitaet/person/anrede</code>
berechtigung	<code>berechtigung</code>
bundesland	<code>identifikation/adresse/bundesland</code>
kennung	<code>kennung</code>
klasse	<code>kennung/@klasse</code>
kreis	<code>identifikation/adresse/kreis</code>
land	<code>identifikation/adresse/land</code>
memo	<code>memo</code>
nachname	<code>identifikation/identitaet/person/nachname</code>
niederlassung	<code>identifi.../identitaet/organisation/niederlassung</code>
organisation	<code>identifikation/identitaet/organisation/name</code>
ort	<code>identifikation/adresse/ort</code>
postfach	<code>identifikation/adresse/postfach</code>
postfachleitzahl	<code>identifikation/adresse/postfachleitzahl</code>
postfachort	<code>identifikation/adresse/postfachort</code>
postleitzahl	<code>identifikation/adresse/postleitzahl</code>
strasse	<code>identifikation/adresse/strasse</code>
titel	<code>identifikation/identitaet/person/titel</code>
vorname	<code>identifikation/identitaet/person/vorname</code>
zeile[n]	<code>identifikation/zeilen</code>

#absender.kontakt	Xpath relativ zu <code>absender/kontakt</code>
anrede	<code>identitaet/person/anrede</code>
email	<code>email</code>
fax	<code>fax</code>
nachname	<code>identitaet/person/nachname</code>
niederlassung	<code>identitaet/organisation/niederlassung</code>
organisation	<code>identitaet/organisation/name</code>
telefon	<code>telefon</code>
titel	<code>identitaet/person/titel</code>
url	<code>url</code>
vorname	<code>identitaet/person/vorname</code>
zeile[n]	<code>zeilen</code>

#absender.korrektur	Xpath relativ zu absender/korrektur/identifikation
anrede	identitaet/person/anrede
bundesland	adresse/bundesland
kreis	adresse/kreis
land	adresse/land
nachname	identitaet/person/nachname
niederlassung	identitaet/organisation/niederlassung
organisation	identitaet/organisation/name
ort	adresse/ort
postfach	adresse/postfach
postfachleitzahl	adresse/postfachleitzahl
postfachort	adresse/postfachort
postleitzahl	adresse/postleitzahl
strasse	adresse/strasse
titel	identitaet/person/titel
vorname	identitaet/person/vorname
zeile[n]	zeilen

#absender.korrektur.kontakt	Xpath relativ zu absender/korrektur/kontakt
anrede	identitaet/person/anrede
email	email
fax	fax
nachname	identitaet/person/nachname
niederlassung	identitaet/organisation/niederlassung
organisation	identitaet/organisation/name
telefon	telefon
titel	identitaet/person/titel
url	url
vorname	identitaet/person/vorname
zeile[n]	zeilen

10.2.1.2. Metadatenkontext **berichtspflichtiger.***

#berichtspflichtiger, XPath relativ zu nachricht/berichtspflichtiger oder segment/berichtspflichtiger
wie absender.

10.2.1.3. Metadatenkontext **berichtsempfaenger.***

#berichtsempfaenger, XPath relativ zu nachricht/berichtsempfaenger oder segment/berichtsempfaenger
wie empfaenger.

10.2.1.4. Metadatenkontext **berichtszeitraum.***

#berichtszeitraum	XPath relativ zu nachricht oder segment
jahr	berichtszeitraum/jahr
monat	berichtszeitraum/monat
tag	berichtszeitraum/tag
halbjahr	berichtszeitraum/halbjahr

quartal	berichtszeitraum/quartal
woche	berichtszeitraum/woche

10.2.1.5. Metadatenkontext **empfaenger.***

#empfaenger	Xpath relativ zu empfaenger
anrede	identifikation/identitaet/person/anrede
bundesland	identifikation/adresse/bundesland
kennung	kennung
klasse	kennung/@klasse
kreis	identifikation/adresse/kreis
land	identifikation/adresse/land
nachname	identifikation/identitaet/person/nachname
niederlassung	identifi.../identitaet/organisation/niederlassung
organisation	identifikation/identitaet/organisation/name
ort	identifikation/adresse/ort
postfach	identifikation/adresse/postfach
postfachleitzahl	identifikation/adresse/postfachleitzahl
postfachort	identifikation/adresse/postfachort
postleitzahl	identifikation/adresse/postleitzahl
strasse	identifikation/adresse/strasse
titel	identifikation/identitaet/person/titel
vorname	identifikation/identitaet/person/vorname
zeile[n]	identifikation/zeilen

#empfaenger.kontakt	Xpath relativ zu empfaenger/kontakt
anrede	identitaet/person/anrede
email	email
fax	fax
nachname	identitaet/person/nachname
niederlassung	identitaet/organisation/niederlassung
organisation	identitaet/organisation/name
telefon	telefon
titel	identitaet/person/titel
url	url
vorname	identitaet/person/vorname
zeile[n]	zeilen

10.2.1.6. Metadatenkontext **erhebung.***

erhebung	XPath relativ zu nachricht oder segment
kennung	erhebung/kennung
klasse	erhebung/kennung/@klasse
text	erhebung/text

10.2.1.7. Metadatenkontext **material.***

material	XPath relativ zu nachricht oder segment
kennung	material/kennung
klasse	material/kennung/@klasse

name	material/name
------	---------------

10.2.1.8. Metadatenkontext `protokoll.*`

#protokoll.eingang	XPath relativ zu <code>protokoll/eingang</code>
anwendung	@anwendung
dateiname	@dateiname
datentraeger-id	@datentraeger
datentraegertyp	@datentraegertyp
datum	@datum
fragebogen	@fragebogen
lieferformatname	@lieferformatname
lieferformattyp	@lieferformattyp
zeit	@uhrzeit

#protokoll.instanz	XPath relativ zu <code>protokoll/instanz</code>
anwendung	@anwendung
datum	@datum
zeit	@uhrzeit

10.2.1.9. Metadatenkontext `satz.*`

#satz	XPath relativ zu <code>nachricht</code> oder <code>satz</code>
kennung	satz/@kennung

10.2.2. Mapping für DatML/RAW 2.0

10.2.2.1. Metadatenkontext `absender.*`

#absender	Xpath relativ zu <code>absender</code>
anrede	identifikation/identitaet/person/anrede
berechtigung	berechtigung
bundesland	identifikation/adresse/bundesland
hausnummer	identifikation/adresse/hausnummer
email	identifikation/email
fax	identifikation/fax
kennung	kennung
klasse	kennung/@klasse
kreis	identifikation/adresse/kreis
land	identifikation/adresse/land
memo	memo
nachname	identifikation/identitaet/person/nachname
niederlassung	identifi.../identitaet/organisation/niederlassung
organisation	identifikation/identitaet/organisation/name
ort	identifikation/adresse/ort
postfach	identifikation/adresse/postfach
postfachleitzahl	identifikation/adresse/postfachleitzahl
postfachort	identifikation/adresse/postfachort
postleitzahl	identifikation/adresse/postleitzahl
strasse	identifikation/adresse/strasse

telefon	identifikation/telefon
titel	identifikation/identitaet/person/titel
url	identifikation/url
vorname	identifikation/identitaet/person/vorname
zeile[n]	identifikation/zeilen

#absender.kontakt	Xpath relativ zu absender/kontakt
anrede	identitaet/person/anrede
bundesland	adresse/bundesland
hausnummer	adresse/hausnummer
email	email
fax	fax
kreis	adresse/kreis
land	adresse/land
nachname	identitaet/person/nachname
niederlassung	identitaet/organisation/niederlassung
organisation	identitaet/organisation/name
ort	adresse/ort
postfach	adresse/postfach
postfachleitzahl	adresse/postfachleitzahl
postfachort	adresse/postfachort
postleitzahl	adresse/postleitzahl
strasse	adresse/strasse
telefon	telefon
titel	identitaet/person/titel
url	url
vorname	identitaet/person/vorname
zeile[n]	zeilen

#absender.korrektur	Xpath relativ zu absender/korrektur/identifikation
anrede	identitaet/person/anrede
bundesland	adresse/bundesland
hausnummer	adresse/hausnummer
email	email
fax	fax
kreis	adresse/kreis
land	adresse/land
nachname	identitaet/person/nachname
niederlassung	identitaet/organisation/niederlassung
organisation	identitaet/organisation/name
ort	adresse/ort
postfach	adresse/postfach
postfachleitzahl	adresse/postfachleitzahl
postfachort	adresse/postfachort
postleitzahl	adresse/postleitzahl
strasse	adresse/strasse
telefon	telefon
titel	identitaet/person/titel
url	url
vorname	identitaet/person/vorname
zeile[n]	zeilen

#absender.korrektur.kontakt	Xpath relativ zu absender/korrektur/kontakt
anrede	identitaet/person/anrede
bundesland	adresse/bundesland

hausnummer	adresse/hausnummer
email	email
fax	fax
kreis	adresse/kreis
land	adresse/land
nachname	identitaet/person/nachname
niederlassung	identitaet/organisation/niederlassung
organisation	identitaet/organisation/name
ort	adresse/ort
postfach	adresse/postfach
postfachleitzahl	adresse/postfachleitzahl
postfachort	adresse/postfachort
postleitzahl	adresse/postleitzahl
strasse	adresse/strasse
telefon	telefon
titel	identitaet/person/titel
url	url
vorname	identitaet/person/vorname
zeile[n]	zeilen

10.2.2.2. Metadatenkontext **berichtspflichtiger.***

#berichtspflichtiger, XPath relativ zu `nachricht/berichtspflichtiger` oder `segment/berichtspflichtiger`

wie `absender`.

10.2.2.3. Metadatenkontext **berichtsempfaenger.***

#berichtsempfaenger, XPath relativ zu `nachricht/berichtsempfaenger` oder `segment/berichtsempfaenger`

wie `empfaenger`.

10.2.2.4. Metadatenkontext **berichtszeitraum.***

#berichtszeitraum	XPath relativ zu <code>nachricht</code> oder <code>segment</code>
jahr	berichtszeitraum/jahr
monat	berichtszeitraum/monat
semester	berichtszeitraum/semester
tag	berichtszeitraum/tag
halbjahr	berichtszeitraum/halbjahr
quartal	berichtszeitraum/quartal
woche	berichtszeitraum/woche

10.2.2.5. Metadatenkontext **empfaenger.***

#empfaenger

wie `absender`, mit Ausnahme der Metadaten `berechtigung` und `memo`.

#empfaenger.kontakt
wie absender.kontakt.

10.2.2.6. Metadatenkontext **erhebung.***

erhebung	XPath relativ zu nachricht oder segment
kennung	erhebung/kennung
klasse	erhebung/kennung/@klasse
text	erhebung/text

10.2.2.7. Metadatenkontext **material.***

material	XPath relativ zu nachricht oder segment
kennung	material/kennung
klasse	material/kennung/@klasse
name	material/name

10.2.2.8. Metadatenkontext **meldung.***

meldung	XPath relativ zu nachricht oder segment
id	datensegment/meldungsID

10.2.2.9. Metadatenkontext **nachricht.***

nachricht	Xpath relativ zu nachricht
id	nachrichtenID

10.2.2.10. Metadatenkontext **nichtantwort.***

#nichtantwort	XPath relativ zu satz oder mmgr
text	mm/na

10.2.2.11. Metadatenkontext **protokoll.***

#protokoll.eingang	XPath relativ zu protokoll/dateneingang
anwendung	anwendung/anwendungsname
dateiname	datei/dateiname
datentraeger-id	datei/datentraeger
datentraegertyp	datei/datentraeger/@typ
datum	datum
eingangsstempel	eingangsstempel
fragebogen	onlineformular/formularID
lieferformatname	datei/datenformat
lieferformattyp	datei/datenformat/@typ
zeit	uhrzeit

#protokoll.eingang.beleg	XPath relativ zu <code>protokoll/dateneingang</code>
formular-id	<code>papierformular/formularID</code>
formularname	<code>papierformular/formularname</code>

#protokoll.eingang.datei	XPath relativ zu <code>protokoll/dateneingang</code>
anwendung	<code>datei/anwendung/anwendungsname</code>
dateiname	<code>datei/dateiname</code>
datenformat	<code>datei/datenformat</code>
datenformattyp	<code>datei/datenformat/@typ</code>
datentraeger	<code>datei/datentraeger</code>
datentraegertyp	<code>datei/datentraeger/@typ</code>
datum	<code>datei/datum</code>
dokumentid	<code>datei/dokumentID</code>
hersteller	<code>datei/anwendung/hersteller</code>
server	<code>datei/anwendung/server</code>
upload	<code>datei/upload</code>
version	<code>datei/anwendung/version</code>
zeit	<code>datei/uhrzeit</code>
zertifikat	<code>datei/anwendung/zertifikat</code>

#protokoll.eingang.online	XPath relativ zu <code>protokoll/dateneingang</code>
formular-id	<code>onlineformular/formularID</code>
formularname	<code>onlineformular/formularname</code>

#protokoll.instanz	XPath relativ zu <code>protokoll/dokumentinstanz</code>
anwendung	<code>anwendung/anwendungsname</code>
dateiname	<code>datei/dateiname</code>
datum	<code>datum</code>
dokumentid	<code>dokumentID</code>
hersteller	<code>anwendung/hewsteller</code>
server	<code>anwendung/server</code>
version	<code>anwendung/version</code>
zeit	<code>uhrzeit</code>
zertifikat	<code>anwendung/zertifikat</code>

10.2.2.12. Metadatenkontext `satz.*`

#satz	XPath relativ zu <code>nachricht/datensegment</code> bzw. <code>nachricht/datensegment</code>
kennung	<code>satz/@kennung</code>

