ARIS Process Performance Manager

PROCESS EXTRACTORS

Version 10.0

April 2017

Inhalt

1 Textkonventionen

Im Text werden Menüelemente, Dateinamen usw. folgendermaßen kenntlich gemacht:

- Menüelemente, Tastenkombinationen, Dialoge, Dateinamen, Eingaben usw. werden **fett** dargestellt.
- Eingaben, über deren Inhalt Sie entscheiden, werden **<fett und in spitzen Klammern>** dargestellt.
- Einzeilige Beispieltexte werden am Zeilenende durch das Zeichen ↵ getrennt, z. B. ein langer Verzeichnispfad, der aus Platzgründen mehrere Zeilen umfasst.
- Dateiauszüge werden in folgendem Schriftformat dargestellt:

  `Dieser Absatz enthält einen Dateiauszug.`

2 Allgemein

Dieses Handbuch beschreibt die Arbeitsweise der PPM Prozessextraktoren SAP-2-PPM, JDBC-2-PPM und CSV-2-PPM.

Die Prozessextraktoren erzeugen aus den Datenbeständen verschiedener Quellsystemtypen XML-Dateien, die nach ARIS Process Performance Manager (kurz PPM) importiert werden können.

Bitte beachten Sie, dass dieses Handbuch keine Anwender- oder Customizing-Schulung ersetzt. Es stellt eine Referenz dar, die ergänzende Hinweise zur Online-Hilfe enthält.

3 Software-Voraussetzungen

Alle SAP-bezogenen PPM-Produkte können mit den folgenden SAP-Versionen eingesetzt werden:

- SAP R/3 > 4.6C
- SAP JCO 3.0.9

4 XML

Dieses Kapitel enthält Grundlagenwissen zu XML, welches für das Verständnis der weiteren Kapitel notwendig ist.

4.1 XML - Was ist das?

Die Abkürzung XML steht für e**X**tensible **M**arkup **L**anguage (erweiterbare Datenbeschreibungssprache). XML ist eine Metasprache zur Beschreibung von Auszeichnungssprachen wie z. B. HTML. Metasprachen liefern die Regeln, die zur Definition von Dokumenttypen benötigt werden. Auszeichnungssprachen ermöglichen die korrekte Ausgabe von Dokumenten.

4.2 Aufbau eines XML-Dokuments

Ein XML-Dokument setzt sich immer aus zwei Zeichenarten zusammen: den eigentlichen Daten und den so genannten Tags oder auch Markups. Tags sind XML-Anweisungen, die die Aufteilung des Dokuments in Speicherungseinheiten und seine logische Struktur beschreiben. Die Struktur selbst ist in einer Dokumenttypdefinition (DTD) gespeichert.

Tags werden immer paarweise in spitzen Klammern geschrieben. Zu jedem Start-Tag gehört immer ein entsprechender End-Tag.

XML-Attribute werden innerhalb der Tags verwendet. Ein Attribut darf innerhalb eines Tag nur einmal auftreten.

XML-Dokumente bestehen aus Elementen. Zwei XML-Tags und der eingeschlossene Text bilden ein Element. Leere Elemente bestehen aus nur einem Tag und enden immer mit einem Slash (**/**) vor der schließenden Klammer.

Einfache XML-Dokumente können Sie mit einem Texteditor erstellen. Im folgenden Beispiel wird die DTD in der XML-Datei in eckigen Klammern angegeben:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE personenliste
 [
  <!ELEMENT personenliste (nr, name, alter)>
  <!ELEMENT nr (#PCDATA)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT alter (#PCDATA)>
 ]>
<personenliste>
  <nr>001</nr>
  <name>Musterfrau, Karin</name>
  <alter>27</alter>
</personenliste>
```

Wenn Sie dieses Dokument unter einem beliebigen Namen mit der Endung **.xml** abspeichern, kann der Internet Explorer das Dokument strukturiert darstellen.

4.3 PPM und XML

PPM verwendet XML als universelles Datenformat. Die gesamte Konfiguration des PPM-Systems wird über XML-Dateien bereitgestellt.

Die PPM Prozessextraktoren lesen Daten aus unterschiedlichen Quellsystemen aus und speichern diese in PPM-konformen XML-Dateien.

5 Überblick

Dieses Kapitel gibt einen Überblick über das Extrahieren von Daten aus Anwendungssystemen zur Verwendung in ARIS Process Performance Manager. Die extrahierten Daten können durch Verwendung von geeigneten Fragment- und Mapping-Definitionen unverändert über die PPM XML-Importschnittstelle in ARIS Process Performance Manager eingelesen und weiterverarbeitet werden. Die XML-Importschnittstelle ist ausführlich im Handbuch **PPM-Datenimport** beschrieben.

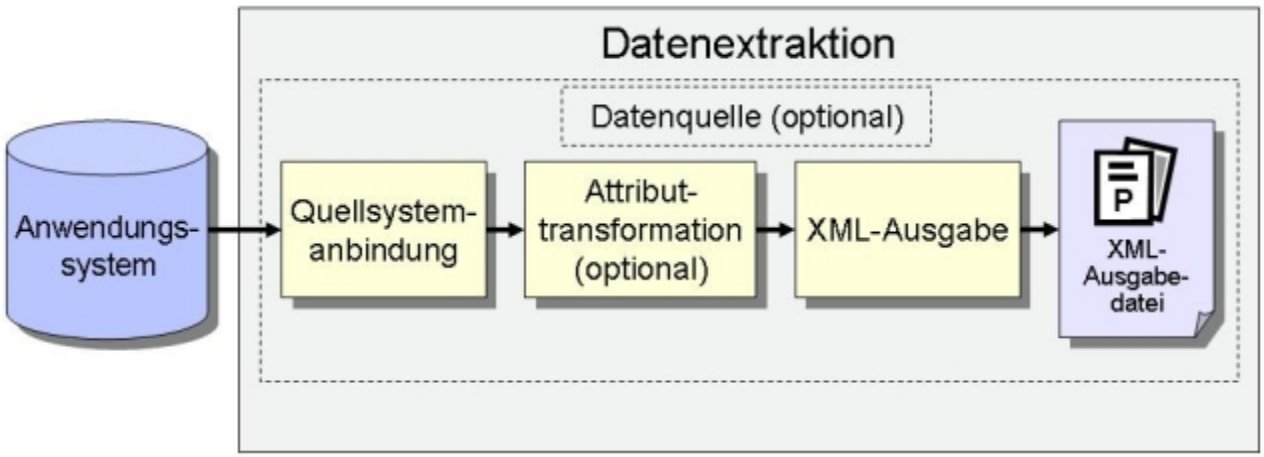
Die PPM-Prozessextraktoren CSV-2-PPM, SAP-2-PPM und JDBC-2-PPM werden über XML-Konfigurationsdateien an das jeweils auszulesende Quellsystem angepasst.

5.1 Datenextraktion

Dieses Kapitel gibt einen Überblick über das Extrahieren von Daten aus Anwendungssystemen zur Verwendung in ARIS Process Performance Manager. Die extrahierten Daten können durch Verwendung von geeigneten Fragment- und Mapping-Definitionen unverändert über die PPM XML-Importschnittstelle in ARIS Process Performance Manager eingelesen und weiterverarbeitet werden. Die XML-Importschnittstelle ist ausführlich im Handbuch **PPM Datenimport** beschrieben.

Die PPM-Prozessextraktoren CSV-2-PPM, JDBC-2-PPM und SAP-2-PPM werden über XML-Konfigurationsdateien an das jeweils auszulesende Quellsystem angepasst.

Die folgende, allgemeine Darstellung veranschaulicht die grundlegende Funktionsweise der Datenextraktion aus Quellsystemen, der anschließenden Transformation der Daten und Ausgabe in XML-Ausgabedateien.



QUELLSYSTEMANBINDUNG

Die Quellsystemanbindung geschieht über die Angabe einer Systemkonfiguration, die die erforderlichen Zugangsdaten zum auszulesenden Quellsystem enthält, wie bspw. die Systemnummer, die Zugriffsart und das Zugangskennwort des Quellsystembenutzers.

Achten Sie darauf, dass der angegebene Quellsystembenutzer eine ausreichende Zugriffsberechtigung besitzt, um die gewünschten Datenfelder auszulesen.

ATTRIBUTTRANSFORMATION (OPTIONAL)

Quellsystemattribute können gegebenenfalls für den Import ins PPM-System verändert bzw. Attributtypen neu hinzugefügt und transformiert werden. Für die Attributtransformation muss eine geeignete XML-Konfiguration erstellt werden.

XML-AUSGABE

Die ausgelesenen Daten werden in Ausgabedateien eines PPM-konformen XML-Formates geschrieben.

DATENQUELLE (OPTIONAL)

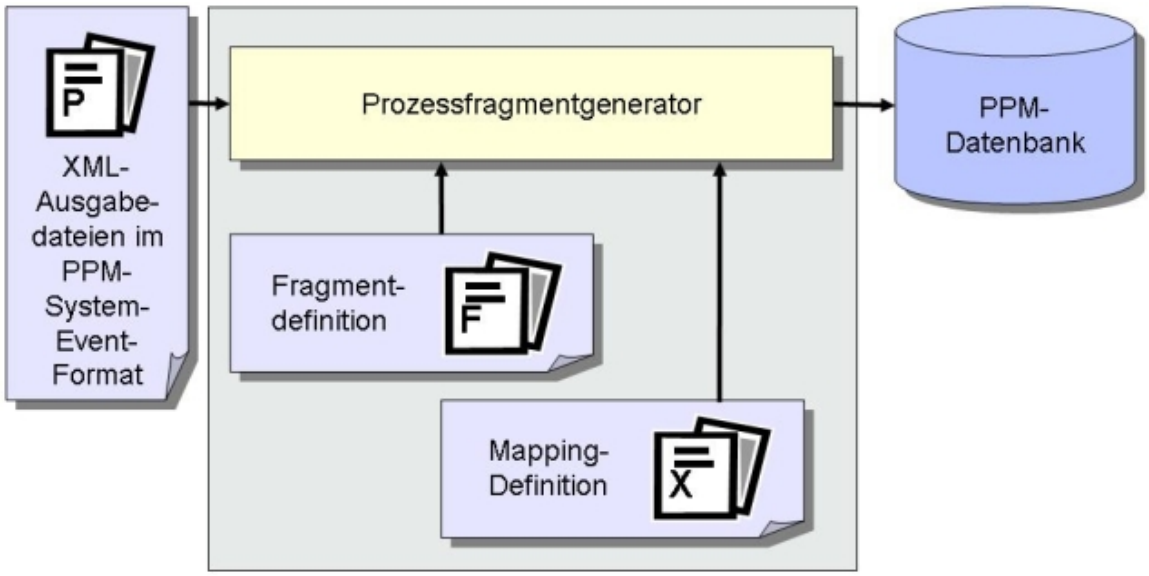
Alle für den Extraktionsvorgang notwendigen XML-Dateien sowie die Ausgabedatei(en) können in einer XML-Datei angegeben werden.

5.2 Datenimport

Neben den XML-Ausgabedateien im PPM-System-Event-Format kann optional eine Datei im im prozessinstanzunabhängigen Kennzahlen- bzw. Dimensionsdatenformat ausgegeben werden, die ihrerseits in ein PPM-System importiert werden kann. Hier wird lediglich auf den Import von XML-Ausgabedateien im PPM-System-Event-Format eingegangen.

Jedem System-Event in der erzeugten XML-Ausgabedatei wird beim Einlesen eine Fragmentdefinition zugeordnet und in der PPM-Datenbank instanziiert. An die Objekte dieser Fragmentinstanz werden die im Mapping spezifizierten Quellsystemattribute kopiert. Abschließend werden die Fragmentinstanzen in der PPM-Datenbank gespeichert.

Schema: Datenimport nach PPM im System-Event-Format



In einem weiteren Arbeitsschritt werden die in die Datenbank eingelesenen Fragmentinstanzen durch das PPM-Kommando **runppmimport** zu Prozessinstanzen zusammengesetzt und Prozesstypen zugeordnet. Nach der Kennzahlenberechnung stehen die Prozessinstanzen für ausführliche Analysen bereit.

Tipp

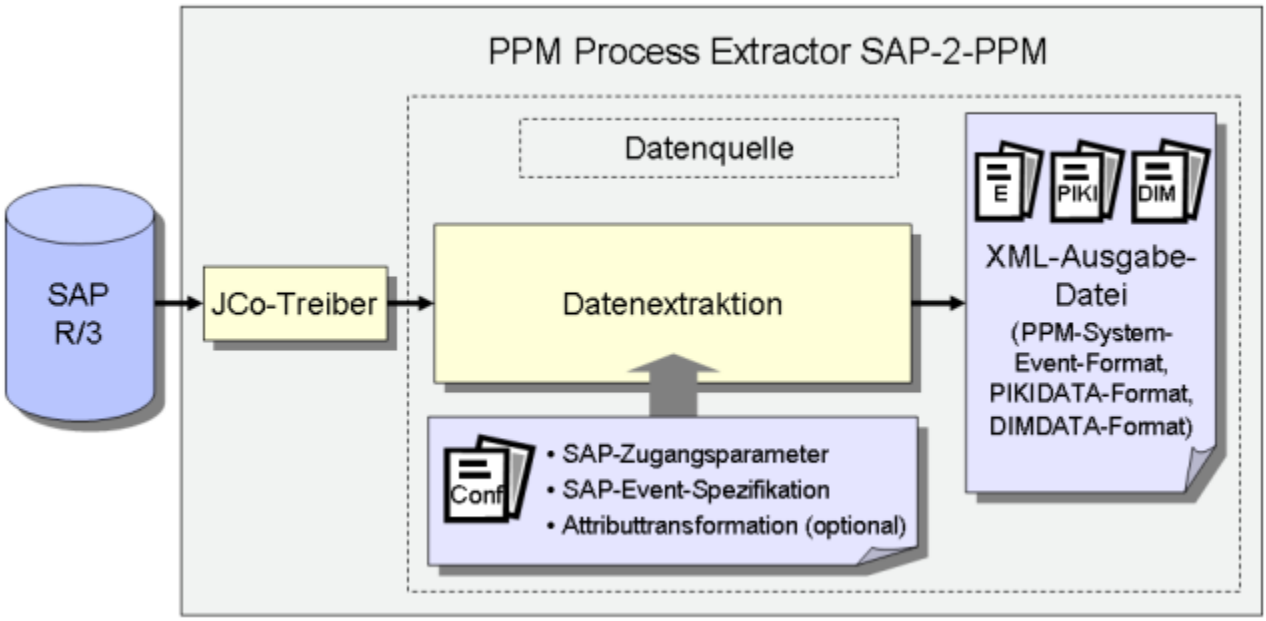
- Näheres zur Konfiguration des XML-Imports ist im Handbuch **PPM-Datenimport** beschrieben.
- Die Konfiguration aller **runppmimport**-relevanten Dateien ist ausführlich im Handbuch **PPM-Customizing** beschrieben.

6 PPM Process Extractor SAP-2-PPM

Dieses Kapitel gibt einen Überblick über die Architektur, Funktionsweise und Konfiguration von PPM Process Extractor SAP-2-PPM.

6.1 Architektur

Die folgende Abbildung veranschaulicht die Funktionalität von PPM Process Extractor SAP-2-PPM:



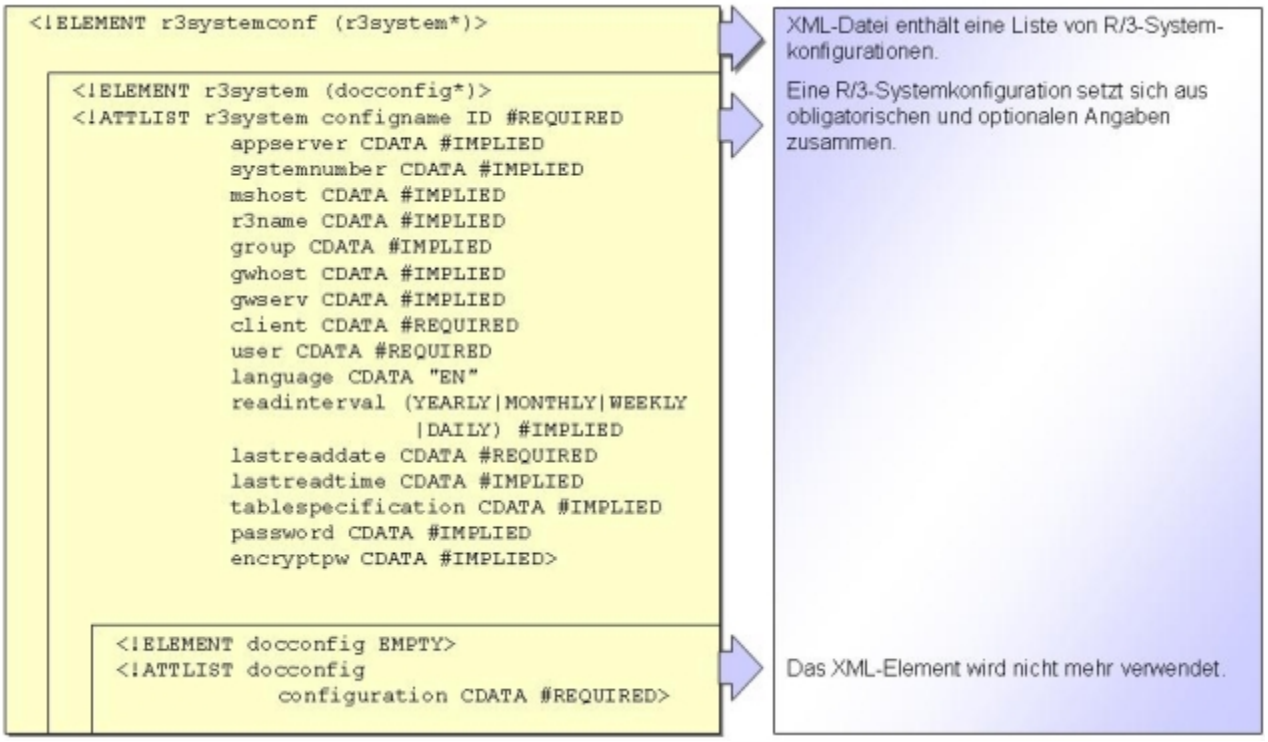
Mit Hilfe des Java Connector-Treibers (JCo) wird über die Zugangsdaten der R/3-Systemkonfiguration eine Verbindung zum SAP R/3-System hergestellt. Anschließend werden Daten aus den Tabellen des R/3-Systems mit der R/3-Tabellenkonfiguration und einer optionalen Attributtransformation über das Modul **Datenextraktion** (Seite 6) ausgelesen und in XML-Dateien eines PPM-konformen Ausgabeformates geschrieben.

Achten Sie darauf, dass der angegebene Quellsystembenutzer eine ausreichende Zugriffsberechtigung besitzt, um die gewünschten Datenfelder auszulesen und dass die Anforderungen an R/3-Quellsysteme erfüllt sind. Wenden Sie sich im Zweifelsfall an einen SAP-Administrator.

6.2 R/3-Systemkonfiguration

Die Zugangsdaten zum R/3-System werden in einer XML-Datei angegeben. Der Name dieser XML-Datei wird dem Kommandozeilenprogramm als Argument übergeben.

Das Format der XML-Datei ist durch folgende DTD vorgegeben:



XML-Attributname	Beschreibung	Beispiel
configname	Eindeutiger Name der Konfiguration	ides_doc_vdap
appserver	Rechnername oder IP-Adresse des Quellsystemrechners	172.20.210.21
systemnumber	SAP-Systemnummer	00
mshost	Name des SAP Message Host	/H/154.37.28.6/S/4435
r3name	R/3-Systemname	A20
group	Name der Application-Server-Gruppe	A20_ALL
gwhost	Rechnername des R/3-Gateways	172.20.210.32

XML-Attributname	Beschreibung	Beispiel
gwserv	Service-Nummer des R/3-Gateways	6789
client	Name des SAP-Mandanten	100
user	Name des SAP-Systembenutzers	sapuser
language	Sprache der ausgelesenen Textfelder	DE
readinterval	Wird nicht mehr verwendet. Weiterhin aus Kompatibilitätsgründen enthalten.	YEARLY
lastreaddate	Datum, ab dem Daten ausgelesen werden. Entfällt bei Verwendung einer **Datenquelle** (Seite 159), Format: **yyyymmdd**	19700101
lastreadtime	Uhrzeit, ab der Daten ausgelesen werden. Entfällt bei Verwendung einer **Datenquelle** (Seite 159), Format: **hhmmss**	125708
tablespecification	Wird nicht mehr verwendet. Weiterhin aus Kompatibilitätsgründen enthalten.	
password	Zugangskennwort zum R/3-System	meinkennwort
encryptpw	Zugangskennwort zum R/3-System in verschlüsselter Form	

Beim erstmaligen Eintrag eines R/3-Systems in die Systemkonfiguration oder Ändern des Kennworts geben Sie das Kennwort im XML-Attribut **password** an. Beim ersten Zugriff auf das R/3-System (z. B. Verbindungstest, Starten eines Auslesevorgangs) wird das Kennwort verschlüsselt und im XML-Attribut **encryptpw** gespeichert. Der Wert des XML-Attributs **password** wird gelöscht.

Die Art des Zugriffs auf ein R/3-System wir durch die Werteangabe von XML-Attributen festgelegt.

R/3-Systemtyp	XML-Attribute	Beispiel
Applikationsserver	appserver systemnumber	... configname="..." appserver="192.168.102.100" systemnumber="01" mshost="" r3name="" group="" gwhost="" gwserv="" ...
Message Host	mshost r3name group	... configname="test_group_sd" appserver="..." systemnumber="03" mshost="/H/156.243.123.6/H/154.34.37.2/S/4435/H/saproutertest/H/112.4.3.2" r3name="A20" group="A20_ALL" gwhost="" gwserv="" ...
Gateway-Server	appserver systemnumber gwhost gwserv	... configname="..." appserver="172.20.212.232" systemnumber="01" mshost="" r3name="" group="" gwhost="172.20.212.240" gwserv="6789" ...

Beispiel

```
...
<r3system configname="ides_doc_vbap" appserver="172.20.210.211"
          systemnumber="00" client="100" user="sapuser"
          language="DE" readinterval="YEARLY"
          lastreaddate="19700101" password="testpassword">
     <docconfig configuration="SD_VBAP"/>
     <docconfig configuration="SD_MSEG"/>
</r3system>
...
```

6.3 R/3-Tabellenkonfiguration (System-Event-Spezifikation)

Die R/3-Tabellenkonfiguration legt fest, welche Tabellenfelder aus dem R/3-System ausgelesen und als Quellsystemattribute in die System-Events geschrieben werden. Sie können in der XML-Datei mehrere Tabellenkonfigurationen mit eindeutigen Namen speichern.

Die R/3-Tabellenkonfiguration setzt sich aus folgenden Teilen zusammen:

GLOBALE TABELLEN

Aus globalen Tabellen werden Informationen gelesen, die für alle System-Events geschrieben werden.

FREMDSCHLÜSSELTABELLEN

Das XML-Element **docreftable** enthält den Namen der Fremdschlüsseltabelle. Es legt fest, wie der aus der System-Event-Tabelle auszulesende Datenbereich eingeschränkt wird. Über die im XML-Element **pkfield** angegebenen Primärschlüsselfelder ist die Fremdschlüsseltabelle mit der System-Event-Tabelle und anderen Fremdschlüsseltabellen verknüpft.

SYSTEM-EVENT-TABELLE

Das XML-Element **doctable** enthält den Namen der System-Event-Tabelle. Es bestimmt die zu einem Belegfluss auszulesenden Belege. Jeder aus der System-Event-Tabelle gelesene Datensatz erzeugt in der Ausgabedatei ein System-Event (XML-Element **event**).

DATENTABELLEN

Die Informationen der System-Event-Tabelle können durch Lesen weiterer Datenfelder aus beliebigen Datentabellen ergänzt werden (z. B. wird aus der System-Event-Tabelle die Materialnummer gelesen und aus einer Datentabelle der zu dieser Nummer gehörende Beschreibungstext).

Folgendes XML-Dateigerüst veranschaulicht die Konfiguration der zu lesenden Tabellen. Im Kap. **Konfiguration des Tabellenzugriffs** (Seite 95) können Sie nachlesen, welche XML-Elemente bzw. -Attribute optional sind.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE r3systemconffields SYSTEM 'xmlextractor_tableconfiguration.dtd'>

<xmlextractor_tableconfiguration>
  <configuration name="..." printname="..." classtouse="...">
    <globaltable name="..." tablename="..." classtouse="...">
      <fieldtoread name="...">
        <textref tablename="..." reffieldname="..."
                textfieldname="..." langfieldname="..."/>
      </fieldtoread>
      ...
    </globaltable>
    <docspec>
      <docreftable name="..." tablename="..."
                                  classtouse="...">
        <condition fieldname="..."
                              logicaloperator="..." >
```

```
      <value>...</value>
    </condition>
    <pkfield name="..." fktablename="..."
            fkfieldname="..." logicaloperator="...">
    <fkpart readfrom="..." startposition="..."
                                  length="..."/>
    <prefix>
      <value>...</value>
    </prefix>
    <postfix>
      <value>...</value>
    </postfix>
  </pkfield>
  ...
</docreftable>
...
<doctable name="..." tablename="..."
              classtouse="..." >
  <condition fieldname="..." logicaloperator="..." >
    <value>...</value>
  </condition>
  <pkfield name="..." fktablename="..."
          fkfieldname="..." logicaloperator="...">
    <fkpart readfrom="..." startposition="..."
                             length="..."/>
    <prefix>
      <value>...</value>
    </prefix>
    <postfix>
      <value>...</value>
    </postfix>
  </pkfield>
  ...
  <fieldtoread name="...">
    <textref tablename="..." reffieldname="..."
          textfieldname="..." langfieldname="..."/>
  </fieldtoread>
  ...
</doctable>
</docspec>
<table name="..." tablename="..."
                              classtouse="...">
  <condition fieldname="..." logicaloperator="...">
     <value>...</value>
  </condition>
  <pkfield name="..." fktablename="..."
           fkfieldname="..." logicaloperator="...">
     <fkpart readfrom="..." startposition="..."
                                  length="..."/>
     <prefix>
       <value>...</value>
     </prefix>
     <postfix>
       <value>...</value>
     </postfix>
  </pkfield>
  ...
  <fieldtoread name="...">
     <textref tablename="..." reffieldname="..."
       textfieldname="..." langfieldname="..."/>
  </fieldtoread>
```

```
      ...
    </table>
    ...
  </configuration>
  ...
</xmlextractor_tableconfiguration>
```

6.3.1 Globale Metadaten

Mit dem XML-Element **globaltable** können Metadaten aus den PPM Prozessextraktoren SAP-2-PPM und JDBC-2-PPM als globale System-Event-Attribute in die Ausgabedatei übernommen werden. Typische Daten für globale System-Event-Attribute sind z. B. die Namen der Konfigurationen, die zum Auslesen verwendet wurden. Die Daten selbst werden aus den in der Kommandozeile angegebenen Argumentwerten ermittelt. Für nicht in der Kommandozeile angegebene Argumente wird der Vorgabewert ermittelt.

Die Groß-/Kleinschreibung der Schlüsselwörter wird berücksichtigt.

Globale System-Event-Attribute sind für alle System-Events der aktuellen Ausgabedatei gültig.

Schlüsselwort	Wert aus Kommandozeilenargument
SYSTEM_CONFIG_FILE_NAME	-systemconfig <filename> ...
SYSTEM_CONFIG_NAME	-systemconfig ... <configname>
TABLE_CONFIG_FILE_NAME	-tableconfig <filename> ...
TABLE_CONFIG_NAME	-tableconfig ... <configname>
BEGIN_DATE,	-begindate <dd.MM.yyyy>
BEGIN_TIME	-begintime <hh:mm:ss>
END_DATE	-enddate <dd.MM.yyyy>
END_TIME	-endtime <hh:mm:ss>
CPD	-cpd <int>
OUT_FILE_NAME	-outfile <filename>
PIKI_DATA_MAPPING_FILE_NAME	-pikidatamapping <filename>...
PIKI_DATA_MAPPING_NAME	-pikidatamapping ... <pcname>
DIM_DATA_MAPPING_FILE_NAME	-dimdatamapping <filename> ...
DIM_DATA_MAPPING_NAME	-dimdatamapping ... <dimname>

Kann für ein Schlüsselwort kein Wert ermittelt werden, wird das Schlüsselwort selbst in die Ausgabedatei geschrieben. Auf diese Weise können Sie Konstanten in die Ausgabedatei übernehmen.

Beispiel

Der folgende Dateiauszug zeigt die Konfiguration, die zum Erzeugen von globalen System-Event-Attributen verwendet wird:

```
<globaltable name="INFO">
  <fieldtoread name="SYSTEM_CONFIG_NAME"/>
</globaltable>
```

Der über die Kommandozeile mit -systemconfig **<configname>** angegebene Konfigurationsname, z. B. **sapppm** , wird als globales System-Event-Attribut **INFO-SYSTEM_CONFIG_NAME** in die Ausgabedatei geschrieben:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <attribute type="INFO-SYSTEM_CONFIG_NAME ">
    sapppm
  </attribute>
  ...
  <event>
    ...
  </event>
  ...
</eventlist>
```

Durch Angabe einer geeigneten Java-Klasse im XML-Attribut **classtouse** können auch beliebige andere Quellsystemtabellen ausgelesen werden.

6.3.2 System-Events sortieren

Sie können die System-Events der XML-Ausgabe gemäß dem Inhalt von Feldern, die Sie als Sortierkriterien festlegen, in alphanumerisch aufsteigender Reihenfolge sortieren. Die Angabe mehrerer Felder als Sortierkriterien ist möglich. Im Sortiervorgang findet dabei eine Priorisierung vom erstgenannten bis zum letztgenannten Feld statt, d. h. es wird zunächst gemäß den Inhalten des erstgenannten Feldes sortiert, dann nach den Inhalten des zweitgenannten Feldes usw.

Tipp

Nutzen Sie die Sortiermöglichkeit für einen effizienteren XML-Import der extrahierten Daten ins PPM-System. Dadurch, dass System-Events, die zu einer Prozessinstanz gehören, aufgrund einer Sortierung in den XML-Ausgabedateien unmittelbar aufeinander folgen, ist es möglich, diese bereits beim XML-Import über den Prozessschlüssel in eine EPK zu importieren.

In der Tabellenkonfiguration werden dazu im XML-Element **parameter** Felder als Sortierkriterien für die System-Event-Tabelle (**doctable**) und die referenzierten Fremdschlüsseltabellen (**docreftable**) angegeben.

Beispiel (Auszug aus der Tabellenkonfiguration)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
                'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
  <configuration name="BANF">
    <docspec>
      <doctable name="EBAN">
        <parameter name="ORDER_BY">
          <value>BNFPO</value>
          <value>BANFN</value>
        </parameter>
        <condition fieldname="ERDAT"
```

```
                logicaloperator="creationtimestamp">
        <value>yyyyMMdd</value>
      </condition>
      <pkfield name="BANFN"/>
      <pkfield name="BNFPO"/>
      <pkfield name="ERDAT"/>
      <fieldtoread name="BSART"/>
      <fieldtoread name="KONNR"/>
      <fieldtoread name="KTPNR"/>
      <fieldtoread name="LIFNR"/>
      <fieldtoread name="MFRNR"/>
    </doctable>
  </docspec>
 </configuration>
</xmlextractor_tableconfiguration>
```

Gemäß den Angaben der Konfiguration **BANF** (XML-Element **parameter**) werden die System-Events nach der Positionsnummer der Bestellanforderungsnummer (Feld **BNFPO**) und in zweiter Priorität nach der Bestellanforderungsnummer (Feld **BANFN**) aufsteigend sortiert in die Ausgabedatei(en) geschrieben.

Die folgende Tabelle zeigt alle Konfigurationsmöglichkeiten:

XML-Element / -Attribut	Beschreibung
parameter	Angabe eines Sortierparameters für eine System-Event-Tabelle (**doctable**) bzw. eine Fremdschlüsseltabelle (**docreftable**)
name	Parametername. Es muss **ORDER_BY** angegeben werden.
value	Angabe eines Sortierfeldes. Das Feld muss als Primärschlüsselfeld (**pkfield**) angegeben sein.

Sie können den Sortierparameter in PPM Customizing Toolkit einstellen. Aktivieren Sie dazu die Registerkarte **System-Event** im Modul **Datenextraktion** der Modulgruppe **Prozess-Merge**. Wechseln Sie in den Bearbeitungsmodus und wählen Sie **System-Event bearbeiten** aus dem Kontextmenü auf der System-Event-Tabelle. Im Schritt **Parameter festlegen** können Sie die gewünschten Einstellungen vornehmen.

6.3.3 Werte eines Feldes aufsummieren

Werte eines Feldes können durch Angabe von Parametern im jeweiligen **table**-Element der Tabellenkonfiguration je nach Datentyp entweder ganzzahlig oder dezimal aufsummiert werden. Dazu ist eine eigene Klasse zu verwenden.

Beispiel

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
```

```
                'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
  <configuration name="SUM">
    <docspec>
      <docreftable name="...">
        ...
      </docreftable>
      <doctable name="...">
        ...
      </doctable>
    </docspec>
    <table name="VBEP"
       classtouse="com.idsscheer.ppm.xmlextractortools.
            extractor.sap2ppm.ZNumberOperation_sap2ppm">
      <parameter name="FIELDS">
        <value>BMENG</value>
        <value>WMENG</value>
      </parameter>
      <parameter name="OPERATION">
        <value>SUM_DECIMAL</value>
      </parameter>
      <pkfield name="VBELN" fktablename="VBAP"
                              fkfieldname="VBELN"/>
      <pkfield name="POSNR" fktablename="VBAP"
                              fkfieldname="POSNR"/>
    </table>
  </configuration>
</xmlextractor_tableconfiguration>
```

In der Beispielkonfiguration ist festgelegt, dass jeweils für jedes der Fließkommazahl-Felder **BMENG** bzw. **WMENG** die Summe mehrerer Werte, die zu einer Verkaufsbelegposition (**POSNR**) eines Verkaufsbelegs (**VBELN**) gehören, berechnet und diese in ein neues System-Event-Attribut geschrieben wird. Der Name des neuen Attributs wird gebildet aus **<tablename>-<operator><fieldname>**, z. B. **VBEP-SUM_DECIMALWMENG**. Für die Aufsummierung von Feldwerten wird die Klasse **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZNumberOperation_sap2ppm** verwendet.

Zum Verkaufsbeleg **5696** existieren im SAP-System folgende Positionsnummern:

VBELN	POSNR	WMENG	BMENG
5696	10	353,000	103,000
5696	10	215,000	52,000
5696	10	127,000	313,000
5696	20	89,000	79,000
5696	20	456,000	29,000
5696	30	23,000	269,000
5696	30	87,000	159,000
5696	30	124,000	256,000

VBELN	POSNR	WMENG	BMENG
5696	30	59,000	237,000

Beim Auslesen mit der Beispielkonfiguration werden pro Positionsnummer für jedes der Felder **BMENG** bzw. **WMENG** die Dezimalsummen gebildet und folgende System-Events erzeugt:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
<event>
  <attribute type="VBAP-POSNR">000010</attribute>
  <attribute type="VBAP-VBELN">0000005696</attribute>
  <attribute type="VBEP-SUM_DECIMALBMENG">468.000</attribute>
  <attribute type="VBEP-SUM_DECIMALWMENG">695.000</attribute>
</event>
<event>
  <attribute type="VBAP-POSNR">000020</attribute>
  <attribute type="VBAP-VBELN">0000005696</attribute>
  <attribute type="VBEP-SUM_DECIMALBMENG">108.000</attribute>
  <attribute type="VBEP-SUM_DECIMALWMENG">545.000</attribute>
</event>
<event>
  <attribute type="VBAP-POSNR">000030</attribute>
  <attribute type="VBAP-VBELN">0000005696</attribute>
  <attribute type="VBEP-SUM_DECIMALBMENG">921.000</attribute>
  <attribute type="VBEP-SUM_DECIMALWMENG">293.000</attribute>
</event>
</eventlist>
```

6.3.4 Felder mehrwertig auslesen

Mittels einer geeigneten Konfiguration des jeweiligen **table**-Elements der Tabellenkonfiguration können alle unterschiedlichen Werte eines zu extrahierenden Feldes (**fieldtoread**) in ein System-Event geschrieben werden. Dazu ist eine eigene Klasse zu verwenden.

Beispiel

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
                'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
  <configuration name="MULTIPLE_VALUES">
    <docspec>
      <docreftable name="...">
        ...
      </docreftable>
      <doctable name="...">
        ...
      </doctable>
    </docspec>
    <table name="VBEP" classtouse="com.idsscheer.ppm.
              xmlextractortools.extractor.sap2ppm.
                       ZTableMultipleValues_sap2ppm">
      <pkfield name="VBELN" fktablename="VBAP"
                              fkfieldname="VBELN"/>
      <pkfield name="POSNR" fktablename="VBAP"
                              fkfieldname="POSNR"/>
```

```
    <fieldtoread name="WMENG"/>
    <fieldtoread name="EDATU"/>
  </table>
  <table name="VBAK" classtouse="com.idsscheer.ppm.
            xmlextractortools.extractor.sap2ppm.
                    ZTableMultipleValues_sap2ppm">
    <pkfield name="VBELN" fktablename="VBEP"
                            fkfieldname="VBELE"/>
    <fieldtoread name="VBTYP"/>
  </table>
 </configuration>
</xmlextractor_tableconfiguration>
```

In der Beispielkonfiguration ist durch Verwendung der Klasse **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZTableMultipleValues_sap2ppm** für alle Datentabellen (**table**-Elemente) festgelegt, dass alle auszulesenden Felder (**fieldtoread**-Elemente) mehrwertig ausgelesen werden, wenn in der Quellsystemdatenbank mehrere Werte zu einem Feld existieren. Das mehrwertig ausgelesene Feld erscheint in der XML-Ausgabedatei als gleichnamige Attribute innerhalb eines System-Event's:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
 <event>
  <attribute type="VBAP-POSNR">000010</attribute>
  <attribute type="VBAP-VBELN">0000004969</attribute>
  <attribute type="VBEP-EDATU">19970103</attribute>
  <attribute type="VBEP-EDATU">19970107</attribute>
  <attribute type="VBEP-WMENG">138.000</attribute>
  <attribute type="VBEP-WMENG">317.000</attribute>
  <attribute type="VBEP-WMENG">496.000</attribute>
 </event>
</eventlist>
```

6.3.5 Aus gelesenen Attributwerten neue Attributwerte konkatenieren

Die Klasse **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZTableConcatFKs_sap2ppm** kann verwendet werden, um während des Auslesevorgangs aus einem SAP-System neue System-Event-Attribute zu erzeugen, indem die Werte bereits ausgelesener Attribute konkateniert werden. Die neu erzeugten System-Event-Attribute können in weiteren Datentabellenkonfigurationen (XML-Element **table**) referenziert werden.

Beispiel

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
                'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
 <configuration name="SD">
  <docspec>
   <doctable name="RBKP">
    <condition logicaloperator="eq" fieldname="GJAHR">
```

```
        <value>2005</value>
      </condition>
      <pkfield name="BELNR"/>
      <pkfield name="GJAHR"/>
    </doctable>
  </docspec>
  <table name="RBKP_NEW" classtouse="com.idsscheer.ppm.
                    xmlextractortools.extractor.
                sap2ppm.ZTableConcatFKs_sap2ppm">
    <pkfield name="BELNR" fktablename="RBKP"
                          fkfieldname="BELNR"/>
    <pkfield name="GJAHR" fktablename="RBKP"
                          fkfieldname="GJAHR"/>
    <fieldtoread name="AWKEY_GENERATED"/>
  </table>
 </configuration>
</xmlextractor_tableconfiguration>
```

Wenn Sie die Klasse **ZTableConcatFKs_sap2ppm** verwenden, müssen Sie genau ein **fieldtoread**-Element angeben. Das Element enthält den Namen des zu erzeugenden System-Event-Attributs. Im Beispiel wird ein neues System-Event-Attribut mit dem Namen **RBKP_NEW-AWKEY_GENERATED** erzeugt. Der Wert des Attributs setzt sich aus den Werten der angegebenen **pkfield**-Elemente zusammen.
Bei den **pkfield**-Elementen können die üblichen Operationen **fkpart**, **prefix** und **postfix** verwendet werden.

Ein mit obiger Konfiguration erzeugtes System-Event könnte bspw. wie folgt aussehen:

```
...
<event>
  <attribute type="RBKP-BELNR">5105601204</attribute>
  <attribute type="RBKP-GJAHR">2005</attribute>
  <attribute type="RBKP_NEW-AWKEY_GENERATED">51056012042005
  </attribute>
</event>
...
```

Sonderfall

In der **table**-Konfiguration kann das spezielle **pkfield**-Element **MANDANT** verwendet werden, welches beim Auslesen automatisch die Nummer des SAP-Mandanten hinzufügt. Hierbei müssen die XML-Attribute **name**, **fktablename** und **fkfieldname** den Wert **MANDANT** haben.

Beispiel (inkl. postfix-Element)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
                'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
  <configuration name="SD">
    <docspec>
      <doctable name="RBKP">
        <condition logicaloperator="eq" fieldname="GJAHR">
          <value>2005</value>
        </condition>
        <pkfield name="BELNR"/>
        <pkfield name="GJAHR"/>
      </doctable>
    </docspec>
```

```
    <table name="RBKP_NEWER" classtouse="com.idsscheer.
                   ppm.xmlextractortools.extractor.
                    sap2ppm.ZTableConcatFKs_sap2ppm">
      <pkfield name="MANDANT" fktablename="MANDANT"
                              fkfieldname="MANDANT">
        <postfix>
          <value> test </value>
        </postfix>
      </pkfield>
      <pkfield name="BELNR" fktablename="RBKP"
                               fkfieldname="BELNR"/>
      <pkfield name="GJAHR" fktablename="RBKP"
                               fkfieldname="GJAHR"/>
      <fieldtoread name="AWKEY_GENERATED"/>
    </table>
  </configuration>
</xmlextractor_tableconfiguration>
```

Ein System-Event, das beim Auslesen aus dem Mandanten 800 erzeugt wird, könnte bspw. wie folgt aussehen:

```
...
<event>
  <attribute type="RBKP-BELNR">5105601204</attribute>
  <attribute type="RBKP-GJAHR">2005</attribute>
  <attribute type="RBKP_NEWER-AWKEY_GENERATED"
               >800 test 51056012042005</attribute>
</event>
...
```

Das neu erzeugte Attribut **RBKP_NEWER-AWKEY_GENERATED** kann anschließend wie jedes andere SAP-Tabellenfeld in weiteren **table**-Konfigurationen referenziert werden.

Beispiel

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
                'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
  <configuration name="SD">
    <docspec>
      <doctable name="RBKP">
        <condition logicaloperator="eq" fieldname="GJAHR">
          <value>2005</value>
        </condition>
        <pkfield name="BELNR"/>
        <pkfield name="GJAHR"/>
      </doctable>
    </docspec>
    <table name="RBKP_NEW" classtouse="com.idsscheer.
                  ppm.xmlextractortools.extractor.
                     sap2ppm.ZTableConcatFKs_sap2ppm">
      <pkfield name="BELNR" fktablename="RBKP"
                               fkfieldname="BELNR"/>
      <pkfield name="GJAHR" fktablename="RBKP"
                               fkfieldname="GJAHR"/>
      <fieldtoread name="AWKEY_GENERATED"/>
    </table>
    <table name="BKPF">
      <pkfield name="AWKEY" fktablename="RBKP_NEW"
                      fkfieldname="AWKEY_GENERATED"/>
```

```
      <fieldtoread name="BELNR"/>
    </table>
  </configuration>
</xmlextractor_tableconfiguration>
```

Ein mit dieser Konfiguration erzeugtes System-Event könnte bspw. wie folgt aussehen:

```
...
<event>
  <attribute type="RBKP-BELNR">5105601204</attribute>
  <attribute type="RBKP-GJAHR">2005</attribute>
  <attribute type="RBKP_NEW-AWKEY_GENERATED"
                      >5105601204 2005</attribute>
  <attribute type="BKPF-BELNR">5100004691</attribute>
</event>
...
```

6.3.6 Tabellen verknüpfen

Die Kopf-, Beleg- und Datentabellen sind über Primärschlüsselbeziehungen miteinander verknüpft, die im XML-Element **pkfield** angegeben werden.

XML-Element	XML-Attribut	Beschreibung
pkfield	name	Name des Primärschlüssels (Tabellenspaltenname)
	fktablename	Identifizierer der über Fremdschlüssel-beziehung referenzierten Tabelle
	fkfieldname	Name des Fremdschlüssels (Tabellenspaltenname)
	logicaloperator	Mögliche Werte: **eq**, **neq**, **gt**, **geq**, **lt**, **leq**. Vorgabewert: **eq**
prefix		Zeichenkette, die vor dem gelesenen Fremdschlüsselwert eingefügt wird
postfix		Zeichenkette, die nach dem gelesenen Fremdschlüsselwert eingefügt wird
fkpart	readfrom (optional)	Richtung der Teilzeichenkette zur Bildung des Fremdschlüssels Mögliche Werte: **left**, **right** Vorgabewert: **left**
	startposition	Position, ab der die Teilzeichenkette gebildet wird

XML-Element	XML-Attribut	Beschreibung
	length (optional)	Länge der Teilzeichenkette Vorgabewert: Anfang bzw. Ende der Zeichenkette, abhängig vom Wert des XML-Attributs **readfrom**
value		Wertangabe für Bedingungen, Präfix und Postfix

Beispiel

Der folgende Dateiauszug zeigt die Konfiguration zum Auslesen der Datenbankfelder **VBELN** und **POSNR** der Tabelle **VBAP**. Die Fremdschlüsselbeziehungen werden über die Inhalte der Felder **TABKEY** der Tabelle **CDPOS** hergestellt.
Die ersten zehn Zeichen des Feldes **TABKEY** enthalten den Wert, der in der Tabelle **VBAP** dem Feld **VBELN** zugeordnet ist und die folgenden sechs Zeichen den Wert, der in der Tabelle **VBAP** dem Feld **POSNR** zugeordnet ist.

```
<table name="VBAP">
   <pkfield name="VBELN" fktablename="CDPOS"
                         fkfieldname="TABKEY">
      <fkpart startposition="0" length="10"/>
   </pkfield>
   <pkfield name="POSNR" fktablename="CDPOS"
                         fkfieldname="TABKEY">
      <fkpart startposition="10" length="6"/>
   </pkfield>
   <fieldtoread name="VBELN"/>
   <fieldtoread name="POSNR"/>
</table>
```

6.3.6.1 Vergleich von Fremdschlüsselbeziehungen

Standardmäßig werden die im XML-Element **pkfield** angegebenen Fremdschlüsselbziehungen auf Gleichheit (Vorgabewert: **logicaloperator="eq"**) geprüft. Das gilt für entsprechende Definitionen in der Belegtabelle (**doctable**), Belegkopftabelle (**docreftable**) und Datentabelle (**table**).
Durch die Angabe der folgenden Operatoren können Sie auch andere Vergleiche durchführen:

- **neq** (ungleich)
- **lt** (kleiner als)
- **gt** (größer als)
- **geq** (größer gleich)
- **leq** (kleiner gleich)

Die Vergleiche erfolgen lexikographisch gemäß der Unicode-Zeichenliste, d. h. der Wert **00999** ist kleiner als der Wert **9**. Es können auch alphanumerische Textwerte verglichen werden.

Beispiel

Zu jedem Datumswert **EEINV-EINZDAT** soll der zugehörige Bezugszeitraum ermittelt werden. Dieser stellt ein Zeitintervall mit einem Anfangs- und Enddatum dar. Alle drei Datumsfelder (**EEINV-EINZDAT**, **EANLH-VON**, **EANLH-BIS**) sollen ausgelesen werden. Der folgende Dateiauszug zeigt die entsprechende Konfiguration mit den Fremdschlüsselvergleichen **kleiner gleich** bzw. **größer gleich** für das Auslesen von Feldwerten aus der Tabelle **EANLH**:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
               'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
  <configuration name="ISU">
    <docspec>
      <doctable name="EEINV">
        <pkfield name="EINZBELEG"/>
        <pkfield name="VERTRAG"/>
      </doctable>
    </docspec>
    <table name="EEINV">
      <pkfield name="EINZBELEG" fktablename="EEINV"
                    fkfieldname="EINZBELEG"/>
      <pkfield name="VERTRAG" fktablename="EEINV"
                    fkfieldname="VERTRAG"/>
      <fieldtoread name="ANLAGE"/>
      <fieldtoread name="EINZDAT"/>
    </table>
    <table name="EANLH">
      <pkfield name="ANLAGE" fktablename="EEINV"
                    fkfieldname="ANLAGE"/>
      <pkfield name="VON" fktablename="EEINV"
           fkfieldname="EINZDAT" logicaloperator="leq"/>
      <pkfield name="BIS" fktablename="EEINV"
           fkfieldname="EINZDAT" logicaloperator="geq"/>
      <fieldtoread name="VON"/>
      <fieldtoread name="BIS"/>
      <fieldtoread name="TARIFTYP"/>
    </table>
  </configuration>
</xmlextractor_tableconfiguration>
```

Ein System-Event, das mit dieser Konfiguration ausgelesen wird, sieht bspw. folgendermaßen aus:

```
<event>
  <attribute type="EEINV-EINZBELEG">000000800002</attribute>
  <attribute type="EEINV-VERTRAG">00012532</attribute>
  <attribute type="EEINV-ANLAGE">00045678</attribute>
  <attribute type="EEINV-EINZDAT">20041015</attribute>
  <attribute type="EANLH-VON">20041001</attribute>
  <attribute type="EANLH-BIS">20050930</attribute>
  <attribute type="EANLH-TARIFTYP">YELLOW-PRIVAT</attribute>
</event>
```

6.3.7 Auslesen mit Bedingungen

Das Auslesen der Tabellen kann durch Bedingungen eingeschränkt werden. Sie können mit Bedingungen z. B. steuern, dass beim Auslesen einer Tabelle nur Zeilen gelesen werden, die einem bestimmten Datumsfeld entsprechen. Die Bedingungen werden in der Konfiguration der Tabellenspezifikation direkt im XML-Element **booleancondition** bzw. **condition** angegeben und gelten nur für die zugehörige Tabelle.

Eine Bedingung enthält den Namen des Datenfeldes, den Vergleichsoperator und einen konkreten Wert. Bedingungen lassen sich beliebig komplex verknüpfen (XML-Element **booleancondition**).

XML-Element	XML-Attribut	Beschreibung
booleancondition (optional)	logicaloperator	Logische Operatoren: **AND**, **OR**, **NOT** Vorgabewert: **AND**
condition (optional)	logicaloperator	Vergleichsoperatoren: **eq**, **neq**, **in**, **notin**, **num_gt**, **num_geq**, **num_lt**, **num_leq**, **like** Operatoren zum Einschränken des auszulesenden Datenbereichs: **creationtimestamp**, **valueconstraint** Vorgabewert: **eq**
	fieldname	Tabellenfeldname

Die XML-Elemente **docbooleancondition** und **doccondition** ermöglichen das bedingte Auslesen einer Datentabelle in Abhängigkeit von bereits gelesenen Tabellenfeldern. Sie werden ähnlich wie **booleancondition** und **condition** konfiguriert. Der Tabellenname und der Name der Spalte, die die bereits gelesenen Datenfelder enthält, werden in den XML-Attributen **tablename** und **fieldname** angegeben.

XML-Element	XML-Attribut	Beschreibung
docbooleancondition (optional)	logicaloperator	Logischer Operator: **AND**, **OR**, **NOT** Vorgabewert: **AND**
Doccondition (optional)	logicaloperator	Vergleichsoperator: **eq**, **neq**, **in**, **notin**, **exists**, **notexists** Vorgabewert: **eq**
	tablename	Name der Tabelle, die die bereits gelesenen Tabellenfelder enthält
	fieldname	Spaltenname

6.3.7.1 Bedingungsoperatoren

Gemeinsame Operatoren

Folgende Vergleichsoperatoren werden von den beiden XML-Elementen **condition** und **doccondition** unterstützt:

Operator	Beschreibung
eq	Feldinhalt ist gleich dem angegebenen Wert. (Java-String-Vergleich unter Berücksichtigung von Groß-/Kleinschreibung)
neq	Feldinhalt ist ungleich dem angegebenen Wert. (Java-String-Vergleich unter Berücksichtigung von Groß-/Kleinschreibung)
in	Feldinhalt ist gleich einem angegebenen Wert aus einer Wertemenge. (Java-String-Vergleich unter Berücksichtigung von Groß-/Kleinschreibung)
notin	Feldinhalt ist ungleich einem angegebenen Wert aus einer Wertemenge. (Java-String-Vergleich unter Berücksichtigung von Groß-/Kleinschreibung)

condition-Operatoren

Operator	Beschreibung
num_gt	Feldinhalt ist größer als angegebener Wert.
num_geq	Feldinhalt ist größer als oder gleich dem angegebenen Wert.
num_lt	Feldinhalt ist kleiner als angegebener Wert.
num_leq	Feldinhalt ist kleiner als oder gleich dem angegebenen Wert.

Operator	Beschreibung
like (nur für alphanumerische R/3-Datentypen, d. h. Dictionary-Typen ACCP, CHAR, CLNT, CUKY, LCHR, NUMC, UNIT, VARC, TIMS oder DATS)	Vergleich von Feldwerten mit einer variablen Zeichenkette Folgende Platzhalter sind erlaubt: ***** Kein bzw. beliebig viele Zeichen **?** Genau ein beliebiges Zeichen **** Maskierungszeichen für die Suche nach Platzhaltern bzw. Maskierungszeichen in der Form: **** bzw. ***** bzw. **\?** Beispiel: `<condition fieldname="OBJECTID" logicaloperator="like"> <value>*10?0\\20?0*</value> </condition>` Gesucht werden Werte wie z. B. 5551050\20106667 oder 1080\204044, aber bspw. nicht 34510550\2030*
creationtimestamp	Aus R/3-Feldern mit Zeitangaben werden Zeitstempel (Datum und Uhrzeit) gelesen. Deren Werte bilden die Grundlage für die Einschränkung des auszulesenden Datenbereichs mit den Kommandozeilenparametern **-begindate** (**-begintime**) bzw. **-enddate** (**-endtime**) [s. Kap. **Quellsystemspezifische Argumente** (Seite 62)]. Mehrere Felder werden durch die Zeichenkombination **#-#** voneinander getrennt. Für jedes Feld werden in XML-Elementen **value** die Formatbeschreibungen der Quellsystemfelder angegeben. Beispiel (für Tabelle **VBAP**): `<condition fieldname="ERDAT#-#ERZET" logicaloperator="creationtimestamp"> <value>dd.MM.yyyy</value> <value>HH:mm:ss</value> </condition>`
valueconstraint	Angabe eines R/3-Feldes mit Integer-Werten, die für die Einschränkung des auszulesenden Datenbereichs mit dem Kommandozeilenparameter

Operator	Beschreibung

	-valueconstraint
	verwendet werden [s. Kap. **Quellsystemspezifische Argumente** (Seite 62)].
	Das auszulesende Feld muss vom ABAP-Datentyp NUMC bzw. INT4 sein.
	Beispiel (für Tabelle **VBAP**):
	`<condition fieldname="POSNR"`
	`logicaloperator="valueconstraint"/>`

doccondition-Operatoren

Operator	Beschreibung
exists	Feld existiert.
notexists	Feld existiert nicht.

6.3.7.2 Komplexe Bedingungen

Der folgende Dateiauszug veranschaulicht verschachtelte Bedingungen am Beispiel der Belegkopftabelle:

```
...
  <docreftable tablename="VBAK" >
    <booleancondition logicaloperator="AND">
      <condition fieldname="ERDAT#-#ERZET"
                 logicaloperator="creationtimestamp">
        <value>yyyyMMdd</value>
        <value>HHmmss</value>
      </condition>
      <condition fieldname="VBTYP" logicaloperator="in">
        <value>C</value>
        <value>K</value>
      </condition>
      <booleancondition logicaloperator="OR">
        <booleancondition logicaloperator="AND">
          <condition fieldname="VKORG" logicaloperator="eq">
            <value>1000</value>
          </condition>
          <condition fieldname="VKBUR" logicaloperator="eq">
            <value>0041</value>
          </condition>
        </booleancondition >
        <booleancondition logicaloperator="AND">
          <condition fieldname="VKORG" logicaloperator="eq">
            <value>2000</value>
          </condition>
          <condition fieldname="VKBUR" logicaloperator="neq">
            <value>0060</value>
          </condition>
```

```
      </booleancondition >
    </booleancondition >
  </booleancondition >
  ...
</docreftable>
...
```

Aussagenlogisch entspricht der gezeigte Dateiauszug der folgenden Bedingung:

ERDAT und ERZET enthalten den Erstellzeitpunkt
und
VBTYP ist C oder V
und
((VKORG ist gleich 1000 **und** VKBUR ist gleich 0041) **oder** (VKORG ist gleich 2000 **und** VKBUR ist ungleich 0060))

6.3.8 Datenbankfelder auslesen

Die Datenbankfelder der Belegtabelle und der verknüpften Datentabellen, aus denen Werte ausgelesen werden sollen, werden in den XML-Elementen **fieldtoread** angegeben. Für jedes XML-Element **fieldtoread** wird eine Zeile der Form

```
<attribute type="Type">Wert</attribute>
```

in die XML- Ausgabedatei geschrieben.

Type setzt sich aus dem Tabellennamen, dem Datenfeldnamen und einem optionalen Textfeldnamen zusammen.
Wert ist der aus dem entsprechenden Datenfeld gelesene Wert. Alle Werte werden in Textform geschrieben.

Optional kann anstelle des direkten Datenfeldwertes der Wert der referenzierten Tabelle (XML-Element **textref**) ausgelesen werden. Die optionale Angabe **langfieldname** liest den Text des Datenfeldes sprachabhängig aus.

Aus den Datentabellen werden über Primärschlüsselbeziehungen ergänzende Informationen ausgelesen.

XML-Element	XML-Attribut	Beschreibung
fieldtoread	name	Name der Tabellenspalte, die das auszulesende Datenfeld enthält
textref	tablename	Tabellenname der referenzierten Datentabelle
	reffieldname	Name des Fremdschlüssels
	textfieldname	Name der Tabellenspalte, die das auszulesende Datenfeld enthält

XML-Element	XML-Attribut	Beschreibung
	langfieldname (optional)	Name der Tabellenspalte, die das sprachabhängig auszulesende Datenfeld enthält
fkpart	readfrom (optional)	Richtung der Teilzeichenkette zur Bildung des Fremdschlüssels Mögliche Werte: **left**, **right** Vorgabewert: **left**
	startposition	Position, ab der die Teilzeichenkette gebildet wird
	length (optional)	Länge der Teilzeichenkette Vorgabewert: Anfang bzw. Ende der Zeichenkette (XML-Attribut **readfrom**)

Bevor tatsächlich Daten ausgelesen werden, wird überprüft, ob die konfigurierten Tabellen und Datenfelder existieren und ob der angegebene Systembenutzer eine ausreichende Zugriffsberechtigung hat.

Beispiel

Die gelesenen Feldwerte werden als Attribute eines System-Event's in die XML-Ausgabedatei geschrieben.

Nicht lokalisierte Tabellenfelder:

```
...
  <attribute type="VBAP-WERKS">SB</attribute>
...
```

Lokalisierte Tabellenfelder:

```
...
  <attribute type="VBAP-WERKS-NAME">Saarbrücken</attribute>
...
```

6.3.9 Beispiel: XML-Konfiguration und -Ausgabedatei

Dieses Kapitel veranschaulicht anhand eines Praxisbeispiels die Tabellenkonfiguration zum Auslesen von Daten aus einem SAP-System und zeigt einen Auszug aus der erzeugten XML-Ausgabedatei.

6.3.9.1 Tabellenkonfiguration

Folgender Dateiauszug zeigt die Tabellenkonfiguration **SD_C** zum Auslesen von Daten aus einem SAP-SD-System:

```
...
<xmlextractor_tableconfiguration>
  <configuration name="SD_C">
```

```
<docspec>
  <docreftable name="VBAK">
    <booleancondition>
      <condition fieldname="ERDAT#-#ERZET"
           logicaloperator="creationtimestamp">
        <value>yyyyMMdd</value>
        <value>HHmmss</value>
      </condition>
      <condition fieldname="VBTYP"
                       logicaloperator="eq">
        <value>C</value>
      </condition>
    </booleancondition>
    <pkfield name="VBELN"/>
  </docreftable>
  <doctable name="Auftraege" tablename="VBAP">
    <pkfield name="VBELN" fktablename="VBAK"
                       fkfieldname="VBELN"/>
    <pkfield name="POSNR"/>
    <fieldtoread name="ERDAT"/>
    <fieldtoread name="ERZET"/>
    <fieldtoread name="MATNR">
      <textref tablename="MAKT" reffieldname="MATNR"
        textfieldname="MAKTX" langfieldname="SPRAS"/>
    </fieldtoread>
    <fieldtoread name="KONDM">
      <textref tablename="T178T" reffieldname="KONDM"
        textfieldname="VTEXT" langfieldname="SPRAS"/>
    </fieldtoread>
    <fieldtoread name="SPART">
      <textref tablename="TSPAT" reffieldname="SPART"
        textfieldname="VTEXT" langfieldname="SPRAS"/>
    </fieldtoread>
    <fieldtoread name="WERKS">
      <textref tablename="T001W" reffieldname="WERKS"
                          textfieldname="NAME1"/>
    </fieldtoread>
    <fieldtoread name="CHARG"/>
    <fieldtoread name="PSTYV"/>
    <fieldtoread name="ERNAM"/>
    <fieldtoread name="NETWR"/>
  </doctable>
</docspec>
<table name="VBAK">
  <pkfield name="VBELN" fktablename="Auftraege"
                        fkfieldname="VBELN"/>
  <fieldtoread name="VTWEG">
    <textref tablename="TVTWT" reffieldname="VTWEG"
       textfieldname="VTEXT" langfieldname="SPRAS"/>
  </fieldtoread>
  <fieldtoread name="VKORG">
    <textref tablename="TVKOT" reffieldname="VKORG"
       textfieldname="VTEXT" langfieldname="SPRAS"/>
  </fieldtoread>
  <fieldtoread name="VDATU"/>
  <fieldtoread name="VKBUR">
    <textref tablename="TVKBT" reffieldname="VKBUR"
       textfieldname="BEZEI" langfieldname="SPRAS"/>
  </fieldtoread>
  <fieldtoread name="VKGRP">
    <textref tablename="TVGRT" reffieldname="VKGRP"
```

```
          textfieldname="BEZEI" langfieldname="SPRAS"/>
      </fieldtoread>
      <fieldtoread name="VBTYP"/>
      <fieldtoread name="AUART"/>
    </table>
    <table name="MARA">
      <pkfield name="MATNR" fktablename="Auftraege"
                              fkfieldname="MATNR"/>
      <fieldtoread name="MATNR"/>
      <fieldtoread name="MTART">
        <textref tablename="T134T" reffieldname="MTART"
          textfieldname="MTBEZ" langfieldname="SPRAS"/>
      </fieldtoread>
    </table>
  </configuration>
</xmlextractor_tableconfiguration>
```

Die Tabellenkonfiguration **SD_C** enthält folgende Tabellen:

BELEGKOPFTABELLE

```
...
   <docreftable name="VBAK">
     <booleancondition>
       <condition fieldname="ERDAT#-#ERZET"
           logicaloperator="creationtimestamp">
         <value>yyyyMMdd</value>
         <value>HHmmss</value>
       </condition>
       <condition fieldname="VBTYP"
                        logicaloperator="eq">
         <value>C</value>
       </condition>
     </booleancondition>
     <pkfield name="VBELN"/>
   </docreftable>
...
```

Identifizierer und Name der Belegkopftabelle des R/3-Systems ist jeweils **VBAK**. Es werden alle Auftragsbelegköpfe (**VBTYP=C**) des Zeitraums gelesen, der in der Kommandozeile angegeben wurde. Die **value**-Elemente der Bedingungsoperatoren **creationtimestamp** legen das Format der angegebenen Zeitstempel fest.

Name der Primärschlüsseltabellenspalte ist **VBELN**. Für jeden unterschiedlichen Feldwert **VBELN** werden aus der verknüpften Belegtabelle Datensätze gelesen, für die **VBELN** denselben Wert wie in der Belegkopftabelle hat.

BELEGTABELLE

```
...
   <doctable name="Auftraege" tablename="VBAP">
     <pkfield name="VBELN" fktablename="VBAK"
                         fkfieldname="VBELN"/>
     <pkfield name="POSNR"/>
     <fieldtoread name="ERDAT"/>
     <fieldtoread name="ERZET"/>
     <fieldtoread name="MATNR">
       <textref tablename="MAKT" reffieldname="MATNR"
          textfieldname="MAKTX" langfieldname="SPRAS"/>
     </fieldtoread>
```

```
    <fieldtoread name="KONDM">
      <textref tablename="T178T" reffieldname="KONDM"
        textfieldname="VTEXT" langfieldname="SPRAS"/>
    </fieldtoread>
    <fieldtoread name="SPART">
      <textref tablename="TSPAT" reffieldname="SPART"
        textfieldname="VTEXT" langfieldname="SPRAS"/>
    </fieldtoread>
    <fieldtoread name="WERKS">
      <textref tablename="T001W" reffieldname="WERKS"
                        textfieldname="NAME1"/>
    </fieldtoread>
    <fieldtoread name="CHARG"/>
    <fieldtoread name="PSTYV"/>
    <fieldtoread name="ERNAM"/>
    <fieldtoread name="NETWR"/>
  </doctable>
...
```

Für die Belegtabelle **VBAP** wird der Identifizierer **Auftraege** vergeben. Durch die in der Zeile

```
<pkfield name="VBELN" fktablename="VBAK" fkfieldname="VBELN"/>
```

angegebene Fremdschlüsselbeziehung ist die Tabelle mit der Belegkopftabelle verknüpft. Wenn nicht anders angegeben, werden Fremdschlüsselbeziehungen auf Basis gleicher Feldwerte hergestellt (siehe Kap. **Vergleich von Fremdschlüsselbeziehungen** (Seite 24)). Im vorliegenden Beispiel werden nur Tabellenzeilen gelesen, für die der Vergleich der Feldwerte **VBAP-VBELN** und **VBAK-VBELN** gleiche Werte liefert. Der Primärschlüssel wird aus den Spalten **VBELN** und **POSNR** zusammengesetzt.

Für jede gelesene Tabellenzeile wird in der XML-Ausgabedatei ein System-Event (XML-Element **event**) erzeugt. Für jedes XML-Element **fieldtoread** wird eine Zeile der Form

```
<attribute type="...">...</attribute>
```

in die Ausgabedatei geschrieben.

Die in der **doctable**-Definition angegebenen Primärschlüsselfelder (**pkfield**) werden automatisch ausgelesen. Sie müssen für sie keine **fieldtoread**-Elemente angeben.

Bei einigen **fieldtoread**-Elementen wird zusätzlich zum gelesenen Datenfeldwert der aus der referenzierten Tabelle (XML-Element **textref**) gelesene Wert geschrieben. Die optionale Angabe **langfieldname** liest den Text des Datenfeldes sprachabhängig aus.
Der vollständige Quellsystemattributtyp setzt sich zusammen aus dem Identifizierer der Belegkopftabelle (**doctable name**), dem Feldnamen (**fieldtoread name**) und dem Namen des referenzierten Textfeldes (**textref ... textfieldname**). Der Typ des Quellsystemattributes des ersten gelesenen **fieltoread**-Elements mit referenzierter Tabelle sieht folgendermaßen aus:

```
<attribute type="Auftraege-MATNR-MAKTX">...</attribute>
```

DATENTABELLEN

```
...
   <table name="VBAK">
      <pkfield name="VBELN" fktablename="Auftraege"
                            fkfieldname="VBELN"/>
      <fieldtoread name="VTWEG">
        <textref tablename="TVTWT" reffieldname="VTWEG"
            textfieldname="VTEXT" langfieldname="SPRAS"/>
      </fieldtoread>
      <fieldtoread name="VKORG">
        <textref tablename="TVKOT" reffieldname="VKORG"
            textfieldname="VTEXT" langfieldname="SPRAS"/>
      </fieldtoread>
      <fieldtoread name="VDATU"/>
      <fieldtoread name="VKBUR">
        <textref tablename="TVKBT" reffieldname="VKBUR"
            textfieldname="BEZEI" langfieldname="SPRAS"/>
      </fieldtoread>
      <fieldtoread name="VKGRP">
        <textref tablename="TVGRT" reffieldname="VKGRP"
            textfieldname="BEZEI" langfieldname="SPRAS"/>
      </fieldtoread>
      <fieldtoread name="VBTYP"/>
      <fieldtoread name="AUART"/>
   </table>

   <table name="MARA">
      <pkfield name="MATNR" fktablename="Auftraege"
                              fkfieldname="MATNR"/>
      <fieldtoread name="MATNR"/>
      <fieldtoread name="MTART">
        <textref tablename="T134T" reffieldname="MTART"
            textfieldname="MTBEZ" langfieldname="SPRAS"/>
      </fieldtoread>
   </table>
...
```

Über die Fremdschlüsselbeziehungen auf die Primärschlüsselfelder **VBELN** und **MATNR** der Tabelle **Auftraege** werden aus den Datentabellen **VBAK** und **MARA** ergänzende Informationen ausgelesen.

6.3.9.2 XML-Ausgabedatei (PPM-System-Event-Format)

Der folgende Dateiauszug zeigt einige System-Events der XML- Ausgabedatei, die unter Verwendung der im Kapitel **Tabellenkonfiguration** (Seite 31) angegebenen Konfigurationsdatei erzeugt wurde:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
<event>
  <attribute type="Auftraege-CHARG"></attribute>
  <attribute type="Auftraege-ERDAT">20000214</attribute>
  <attribute type="Auftraege-ERNAM">HDM</attribute>
  <attribute type="Auftraege-ERZET">143616</attribute>
  <attribute type="Auftraege-KONDM"></attribute>
  <attribute type="Auftraege-MATNR">P-100</attribute>
  <attribute type="Auftraege-MATNR-MAKTX">
```

```
    Gewindestab M'8 DIN '8895' ohne Toleranz
  </attribute>
  <attribute type="Auftraege-NETWR">28.70</attribute>
  <attribute type="Auftraege-POSNR">000010</attribute>
  <attribute type="Auftraege-PSTYV">TAN</attribute>
  <attribute type="Auftraege-SPART">01</attribute>
  <attribute type="Auftraege-SPART-VTEXT">Produktsparte 01
  </attribute>
  <attribute type="Auftraege-VBELN">0000000001</attribute>
  <attribute type="Auftraege-WERKS">1000</attribute>
  <attribute type="Auftraege-WERKS-NAME1">Werk 1000 (Hamburg)
  </attribute>
  <attribute type="MARA-MATNR">P-100</attribute>
  <attribute type="MARA-MTART">HAWA</attribute>
  <attribute type="MARA-MTART-MTBEZ">Handelsware
  </attribute>
  <attribute type="VBAK-AUART">TA</attribute>
  <attribute type="VBAK-VBTYP">C</attribute>
  <attribute type="VBAK-VDATU">20000214</attribute>
  <attribute type="VBAK-VKBUR"></attribute>
  <attribute type="VBAK-VKGRP"></attribute>
  <attribute type="VBAK-VKORG">1000</attribute>
  <attribute type="VBAK-VKORG-VTEXT">Deutschl. Frankfurt
  </attribute>
  <attribute type="VBAK-VTWEG">10</attribute>
  <attribute type="VBAK-VTWEG-VTEXT">Endkundenverkauf
  </attribute>
</event>
<event>
  <attribute type="VBAK-VKORG-VTEXT">Deutschl. Frankfurt
  </attribute>
  <attribute type="VBAK-VTWEG-VTEXT">Endkundenverkauf
  </attribute>
  <attribute type="Auftraege-SPART">01</attribute>
  <attribute type="Auftraege-PSTYV">TAD</attribute>
  <attribute type="Auftraege-MATNR">SERVICE</attribute>
  <attribute type="Auftraege-ERDAT">20000214</attribute>
  <attribute type="VBAK-VKBUR"></attribute>
  <attribute type="VBAK-VTWEG">10</attribute>
  <attribute type="Auftraege-SPART-VTEXT">Produktsparte 01
  </attribute>
  <attribute type="Auftraege-ERZET">143616</attribute>
  <attribute type="VBAK-VKORG">1000</attribute>
  <attribute type="Auftraege-POSNR">000020</attribute>
  <attribute type="MARA-MTART-MTBEZ">Dienstleistung
  </attribute>
  <attribute type="Auftraege-WERKS-NAME1">Werk 1000 (Hamburg)
  </attribute>
  <attribute type="Auftraege-CHARG"></attribute>
  <attribute type="Auftraege-WERKS">1000</attribute>
  <attribute type="VBAK-VDATU">20000214</attribute>
  <attribute type="Auftraege-MATNR-MAKTX">Reparatur
  </attribute>
  <attribute type="Auftraege-VBELN">0000000001</attribute>
  <attribute type="VBAK-AUART">TA</attribute>
  <attribute type="VBAK-VKGRP"></attribute>
  <attribute type="Auftraege-NETWR">179.00</attribute>
  <attribute type="MARA-MTART">DIEN</attribute>
  <attribute type="VBAK-VBTYP">C</attribute>
  <attribute type="Auftraege-ERNAM">HDM</attribute>
  <attribute type="MARA-MATNR">SERVICE</attribute>
```

```
  <attribute type="Auftraege-KONDM"></attribute>
</event>
<event>
  <attribute type="VBAK-VKGRP-BEZEI">GR. F2 Hr. Mayer
  </attribute>
  <attribute type="VBAK-VKORG-VTEXT">Deutschl. Frankfurt
  </attribute>
  <attribute type="VBAK-VTWEG-VTEXT">Endkundenverkauf
  </attribute>
  <attribute type="Auftraege-SPART">01</attribute>
  <attribute type="Auftraege-PSTYV">TAN</attribute>
  <attribute type="Auftraege-MATNR">P-100</attribute>
  <attribute type="Auftraege-ERDAT">20000214</attribute>
  <attribute type="VBAK-VKBUR">1000</attribute>
  <attribute type="VBAK-VTWEG">10</attribute>
  <attribute type="Auftraege-SPART-VTEXT">Produktsparte 01
  </attribute>
  <attribute type="Auftraege-ERZET">131257</attribute>
  <attribute type="VBAK-VKBUR-BEZEI">Büro Frankfurt
  </attribute>
  <attribute type="VBAK-VKORG">1000</attribute>
  <attribute type="Auftraege-POSNR">000010</attribute>
  <attribute type="MARA-MTART-MTBEZ">Handelsware</attribute>
  <attribute type="Auftraege-WERKS-NAME1">Werk 1000 (Hamburg)
  </attribute>
  <attribute type="Auftraege-CHARG"></attribute>
  <attribute type="Auftraege-WERKS">1000</attribute>
  <attribute type="VBAK-VDATU">20000214</attribute>
  <attribute type="Auftraege-MATNR-MAKTX">
     Gewindestab M'8 DIN '8895' ohne Toleranz
  </attribute>
  <attribute type="Auftraege-VBELN">0000000002</attribute>
  <attribute type="VBAK-AUART">TA</attribute>
  <attribute type="VBAK-VKGRP">101</attribute>
  <attribute type="Auftraege-NETWR">28.70</attribute>
  <attribute type="MARA-MTART">HAWA</attribute>
  <attribute type="VBAK-VBTYP">C</attribute>
  <attribute type="Auftraege-ERNAM">HDM</attribute>
  <attribute type="MARA-MATNR">P-100</attribute>
  <attribute type="Auftraege-KONDM"></attribute>
</event>
...
</eventlist>
```

6.3.10 Blockweises Auslesen

Sie können das Verhältnis zwischen Ausführ- und Speichereffizienz des Auslesevorgangs beeinflussen.
Der Wert des Kommandozeilenarguments **-cpd** bestimmt die Anzahl der gleichzeitig im Systemspeicher gehaltenen System-Events. Die maximal mögliche Anzahl ist von der Größe des zur Verfügung stehenden Systemspeichers abhängig.

Der tatsächlich belegte Systemspeicher hängt von folgenden Faktoren ab:

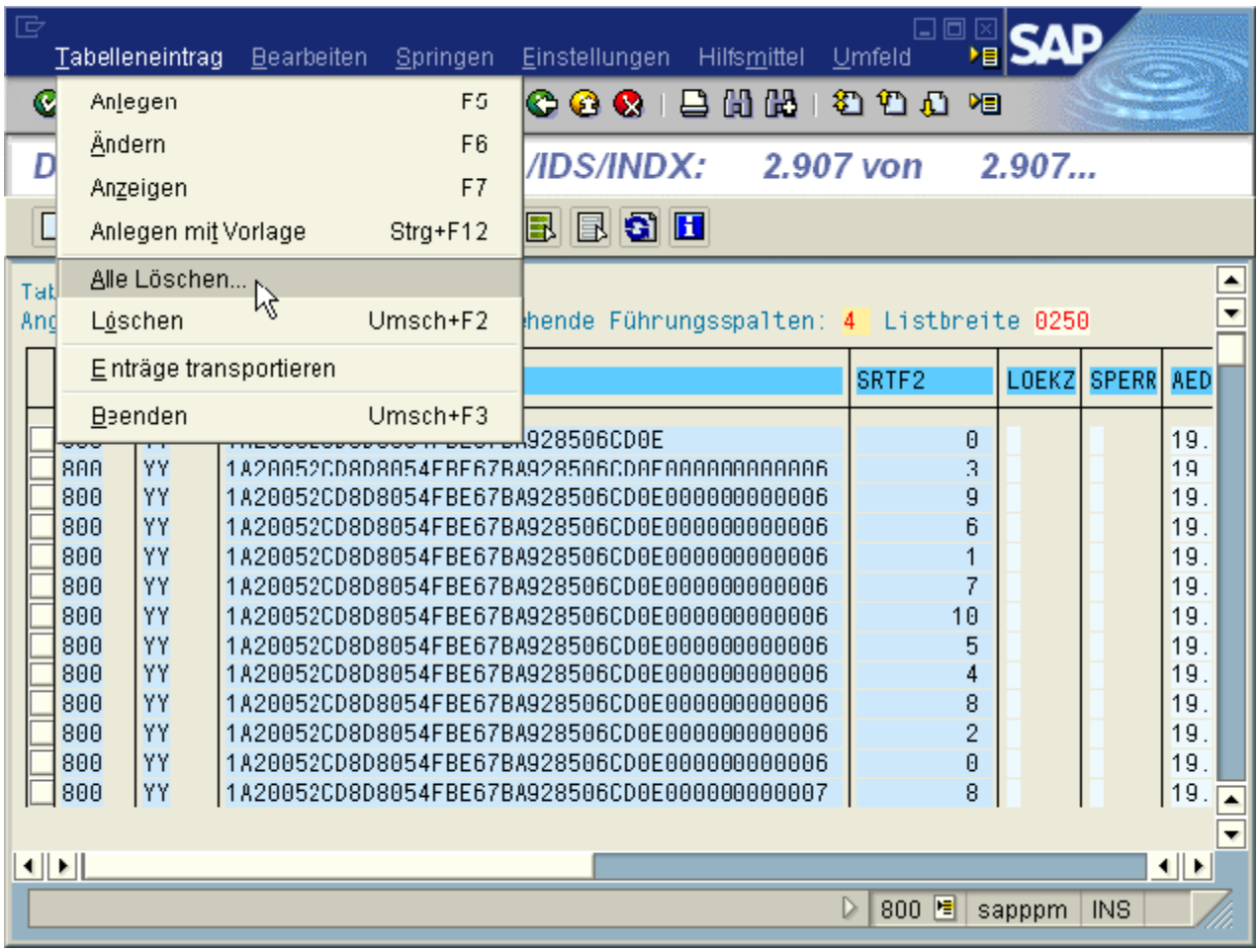
- Anzahl der gesamten im definierten Analysezeitraum zu lesenden System-Events
- Anzahl der parallel im System gehaltenen System-Events (concurrently processed documents)
- Anzahl der zu lesenden Datenfelder (XML-Elemente **fieldtoread** und **reftext**)
- Speicherbedarf der gelesenen Datenfelder

6.3.10.1 Blockweises Auslesen großer Datenmengen

Im Auslesevorgang werden alle Daten eines RFC-Aufrufs auf einmal mittels SAP JCo auf den Java-Client übertragen. Dadurch kann es bei großen Datenmengen zu Speicherüberläufen (OutOfMemoryError) kommen. Dies ist z. B. der Fall, wenn die Primärschlüssel der Beleg- und Belegkopftabellen ermittelt werden. In diesem Fall müssen die Daten zunächst im R/3-System zwischengespeichert und dann blockweise auf den Java-Client übertragen werden. Die Blockgröße wird mit dem Parameter **-cpd** angegeben. Es gibt drei verschiedene Blockmodi, die sich in der Art und Weise, wie die Daten im R/3-System zwischengespeichert werden, unterscheiden.

BLOCKMODUS D

Zum blockweisen Auslesen großer Datenbestände aus dem R/3-System wird Modus **D** empfohlen. Die selektierten Daten werden in der Datenbanktabelle **/IDS/INDX** abgelegt. Bei einem Verbindungsabbruch zum Extraktor oder bei manuellem Abbruch des Extraktionsvorgangs werden die Daten nicht automatisch aus der Datenbanktabelle gelöscht. Die zwischengespeicherten Daten müssen Sie in diesem Fall manuell löschen. Dazu lassen Sie sich mit der Transaktion **SE16** den Inhalt der Tabelle **/IDS/INDX** anzeigen. Markieren Sie dann alle gewünschten Einträge und löschen Sie sie über den Menüeintrag **Alle löschen...** im Hauptmenü **Tabelleneintrag**.



Beispiel

Aus der Tabelle **CDPOS** (Änderungsbelegpositionen) sollen über eine Million Einträge ausgelesen werden. Ohne Verwendung des blockweisen Auslesevorgangs kommt es zum Überlauf des Java-Speichers und Abbruch des Auslesevorgangs.

Durch Ausführen der folgenden Kommandozeile werden die Daten blockweise ausgelesen (Blockparameter in Fettschrift):

```
runsap2ppm -systemconfig Systemconfig.xml -tableconfig TableconfigCDPOS.xml
-outfile eventsCDPOS -nozip -begindate 19700101 -rfc_blockmode D -cpd 10000
```

Durch Angabe des Parameters **-rfc_blockmode** mit dem Argument **D** starten Sie einen blockweisen Auslesevorgang, mit dem alle Daten aus der Datenbanktabelle **CDPOS** in die

Ausgabedatei geschrieben werden. Mit dem Parameter **-cpd** legen Sie fest, wie groß die abzuarbeitenden Blöcke sein sollen. Im Beispiel werden **10000** Dokumente gleichzeitig bearbeitet. Der Vorgabewert ist **5000**.

Der Fortschritt des blockweisen Auslesevorgangs wird vorgabemäßig protokolliert, z. B.:

```
I: 09.05.06 11:30:38: [XML] Lese Daten aus...
...
...
I: 09.05.06 11:32:30: [XML] Tabelle "CDPOS":
       10.000 von 1.036.995 Zeilen gelesen.
I: 09.05.06 11:32:30: [XML] Lese Attribute für
                        10000 Quellsystemevents...
I: 09.05.06 11:32:39: [XML] Tabelle "CDPOS":
       20.000 von 1.036.995 Zeilen gelesen.
I: 09.05.06 11:32:39: [XML] Lese Attribute für
                        10000 Quellsystemevents...
...
```

Durch Angabe von **-progress no** können Sie die zusätzlichen Protokollmeldungen der Blockmodi über den Fortschritt des blockweisen Auslesevorgangs unterbinden.

BLOCKMODUS G

Die selektierten Daten werden in einer internen Tabelle der Funktionsgruppe abgelegt. Die Daten werden im Speicher des R/3-Servers gehalten. Bei einem Verbindungsabbruch zum Extraktor oder bei manuellem Abbruch des Extraktionsvorgangs werden die Daten automatisch im R/3-System gelöscht. Die Angaben gelten ab SAP Release 4.7, d. h. ab SAP Kernel Release 6.10.

BLOCKMODUS M

Die selektierten Daten werden mit **export to memory id** im ABAP-Memory abgelegt. Die Daten werden im Speicher des R/3-Servers gehalten. Bei einem Verbindungsabbruch zum Extraktor oder bei manuellem Abbruch des Extraktionsvorgangs werden die Daten automatisch im R/3-System gelöscht. Die Angaben gelten ab SAP Release 4.7, d. h. ab SAP Kernel Release 6.10.

6.3.11 Änderungsbelege auslesen

Änderungsbelege stellen eine besondere Belegart im SAP R/3-System dar. Diese Belege sind in den Tabellen **CDHDR** (Änderungskopf) und **CDPOS** (Änderungsposition) gespeichert. Da diese Tabellen analog zu den anderen Belegkopf- und Belegtabellen des R/3-Systems aufgebaut sind, kann zum Auslesen der Änderungsbelegtabellen die Konfigurationsstruktur unverändert übernommen werden.

Die Konfiguration zum Lesen der Änderungsbelegtabellen muss so ausgelegt werden, dass die Änderungskopftabelle **CDHDR** vor allen anderen Tabellen gelesen wird.

Änderungsbelege können auf zwei unterschiedliche Arten ausgelesen werden.

6.3.11.1 System-Events aus einzelnen Änderungen erzeugen

Für jeden Änderungsbeleg wird ein System-Event erzeugt.

Beispiel

Aus der Änderungsbelegtabelle sollen für Januar 2003 die Änderungen gelesen werden, die folgende Bedingungen erfüllen:

- Objektklasse ist **VERKBELEG**.
- Geänderte Tabelle ist **VBAK**.
- Geändertes Feld ist **VKGRP**.
- Neuer Feldwert ist **4711** oder **4712**.

Die Bedingungen können in folgender XML-Konfigurationsdatei formuliert werden:

```
...
<configuration name="OneEventForEveryChangeDoc">
  <docspec>
    <docreftable name="CDHDR">
      <booleancondition logicaloperator="AND">
        <condition fieldname="OBJECTCLAS"
                              logicaloperator="eq">
          <value>VERKBELEG</value>
        </condition>
        <condition fieldname="UDATE#-#UTIME"
                  logicaloperator="creationtimestamp">
          <value>yyyyMMdd</value>
          <value>HHmmss</value>
        </condition>
      </booleancondition>
      <pkfield name="OBJECTCLAS"/>
      <pkfield name="OBJECTID"/>
      <pkfield name="CHANGENR"/>
    </docreftable>

    <doctable name="CDPOS">
      <booleancondition logicaloperator="AND">
        <condition fieldname="TABNAME"
                              logicaloperator="eq">
          <value>VBAK</value>
        </condition>
        <condition fieldname="FNAME"
                              logicaloperator="eq">
          <value>VKGRP</value>
        </condition>
        <condition fieldname="VALUE_NEW"
                              logicaloperator="in">
          <value>4711</value>
          <value>4712</value>
        </condition>
      </booleancondition>
      <pkfield name="OBJECTCLAS" fktablename="CDHDR"
                           fkfieldname="OBJECTCLAS"/>
      <pkfield name="OBJECTID" fktablename="CDHDR"
                            fkfieldname="OBJECTID"/>
      <pkfield name="CHANGENR" fktablename="CDHDR"
                            fkfieldname="CHANGENR"/>
      <pkfield name="TABNAME"/>
      <pkfield name="TABKEY"/>
      <pkfield name="FNAME"/>
      <pkfield name="CHNGIND"/>
      <fieldtoread name="TABNAME"/>
      <fieldtoread name="CHNGIND"/>
```

```
    <fieldtoread name="VALUE_NEW"/>
    <fieldtoread name="VALUE_OLD"/>
  </doctable>
 </docspec>

 <table name="CDHDR">
    <pkfield name="OBJECTCLAS" fktablename="CDPOS"
                        fkfieldname="OBJECTCLAS"/>
    <pkfield name="OBJECTID" fktablename="CDPOS"
                          fkfieldname="OBJECTID"/>
    <pkfield name="CHANGENR" fktablename="CDPOS"
                          fkfieldname="CHANGENR"/>
    <fieldtoread name="UDATE"/>
    <fieldtoread name="UTIME"/>
    <fieldtoread name="CHANGENR"/>
    <fieldtoread name="USERNAME"/>
  </table>
</configuration>
...
```

6.3.11.2 System-Events inkl. Änderungen auslesen

Informationen zu Änderungsbelegen aus den SAP-Tabellen **CDPOS** und **CDHDR** werden in speziellen Quellsystemattributen von System-Events unter Verwendung einer besonderen Klasse gespeichert.

Änderungsbelege werden durch eine geeignete Konfiguration einer Datentabelle ausgelesen. Zum Lesen der Änderungsbelege wird die Klasse **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZChangeDocTable_sap2ppm** verwendet.

Wenn es zu einem gelesenen System-Event keine Änderungsbelege gibt, werden nur die konfigurierten Tabellenfelder als Quellsystemattribute übernommen.

In der Konfiguration der Datentabelle, aus der die Änderungsbelege gelesen werden sollen, müssen folgende Bedingungen angegeben werden:

```
<table name="..." classtouse="com.idsscheer.ppm.
          xmlextractortools.extractor.sap2ppm.
                    ZChangeDocTable_sap2ppm">
  <booleancondition>
    <condition fieldname="OBJECTCLAS" ...>
      ...
    </condition>
    <condition fieldname="OBJECTID" ...>
      ...
    </condition>
    <condition fieldname="TABNAME" ...>
      ...
    </condition>
    <booleancondition logicaloperator="OR">
      <condition fieldname="TABKEY" ...>
        ...
      </condition>
      <condition fieldname="TABKEY" ...>
        ...
      </condition>
      ...
    </booleancondition>
  </booleancondition>
```

```
  ...
</table>
```

Andere Bedingungen dürfen nicht angegeben werden.

Die folgende Tabelle zeigt die Quellsystemattribute, die aus gelesenen Änderungsbeleginformationen erzeugt werden. Als Identifizierer der Datentabelle wurde in der Konfiguration **CH_DOCS** angegeben.

Quellsystemattribut	Beschreibung
CH_DOCS-SUM_OF_CHANGES	Gesamtanzahl der Änderungen des gelesenen Belegs (Anzahl der Speichervorgänge).
CH_DOCS-FIRST_CHANGE_DATE CH_DOCS-FIRST_CHANGE_TIME	Datum der ersten Änderung Uhrzeit der ersten Änderung
CH_DOCS-LAST_CHANGE_DATE CH_DOCS-LAST_CHANGE_TIME	Datum der letzten Änderung Uhrzeit der letzten Änderung
CH_DOCS-CHANGE_USER_<x>	Name des Bearbeiters, der Änderungsbelege erzeugt hat. Das Attribut wird für jeden Bearbeiter erzeugt. **<x>** ist die laufende Nummer des Bearbeiters.
CH_DOCS-NUM_OF_CHANGES_USER_<x>	Gesamtanzahl der Änderungen, die der Bearbeiter durchgeführt hat. **<x>** ist die laufende Nummer des Bearbeiters.

Beispiel

Der gezeigte Dateiauszug veranschaulicht die Konfiguration zum Auslesen der Änderungen der Tabellen **VBAK** und **VBAP**:

```
<configuration name="SD_CHANGE_DOC_SUM">
  <docspec>
    <docreftable name="VBAK">
      <condition fieldname="ERDAT#-#ERZET"
                 logicaloperator="creationtimestamp">
        <value>yyyyMMdd</value>
        <value>HHmmss</value>
      </condition>
      <pkfield name="VBELN"/>
    </docreftable>
    <doctable name="VBAP">
      <pkfield name="VBELN" fktablename="VBAK"
                            fkfieldname="VBELN"/>
      <pkfield name="POSNR"/>
      <fieldtoread name="VBELN"/>
      <fieldtoread name="POSNR"/>
      <fieldtoread name="ERDAT"/>
```

```
    <fieldtoread name="ERZET"/>
    <fieldtoread name="ERNAM"/>
   </doctable>
  </docspec>
  <table name="CH_DOCS" classtouse="com.idsscheer.ppm.
                        xmlextractortools.extractor.
                    sap2ppm.ZChangeDocTable_sap2ppm">
   <booleancondition>
    <condition fieldname="OBJECTCLAS">
     <value>VERKBELEG</value>
    </condition>
    <condition fieldname="OBJECTID"
           logicaloperator="eq_concatEventAttrValues">
     <value>VBAP-VBELN</value>
    </condition>
    <condition fieldname="TABNAME" logicaloperator="in">
     <value>VBAK</value>
     <value>VBAP</value>
    </condition>
    <booleancondition logicaloperator="OR">
     <condition fieldname="TABKEY"
           logicaloperator="eq_concatEventAttrValues">
      <value>MANDANT#-#</value>
      <value>VBAP-VBELN#-#</value>
      <value>VBAP-POSNR</value>
     </condition>
     <condition fieldname="TABKEY"
           logicaloperator="eq_concatEventAttrValues">
      <value>MANDANT#-#</value>
      <value>VBAP-VBELN</value>
     </condition>
    </booleancondition>
   </booleancondition>
   <pkfield name="VBELN" fktablename="VBAP"
                              fkfieldname="VBELN"/>
   <pkfield name="POSNR" fktablename="VBAP"
                              fkfieldname="POSNR"/>
  </table>
</configuration>
```

Der Operator **eq_concatEventAttrValues** vergleicht den Inhalt des angegebenen Feldes mit zusammengesetzten Attributwerten der angegebenen Felder. Dabei werden die in den XML-Elementen **value** angegebenen Werte als Identifizierer bereits gelesener Quellsystemattribute interpretiert.

Hinweis zu den Wertzusammensetzungen der Felder OBJECTID und TABKEY

Die Konkatenation der einzelnen Feldwerte (<value>...</value>) erfolgt mittels der angegebenen Trennzeichenkette **#-#** zwischen den einzelnen Feldwerten. Die Zusammensetzung der Wertangaben der Felder **OBJECTID** und **TABKEY** ist im Beispiel wie folgt:

OBJECTID

Wertangabe	Beschreibung	Beispielwert
VBAP-VBELN	Die Übereinstimmung mit dem bereits gelesenen Wert des Feldes **VBELN** der Tabelle **VBAP** wird als Bedingung für das Feld **OBJECTID** festgelegt.	0000004972

TABKEY

Wertangabe	Beschreibung	Beispielwert
MANDANT	Parameter, der durch die Nummer des SAP-Mandanten, aus welchem ausgelesen wird, ersetzt wird	800
VBAP-VBELN	Wert des bereits gelesenen Feldes **VBELN** der Tabelle **VBAP**	0000004972
VBAP-POSNR	Wert des bereits gelesenen Feldes **POSNR** der Tabelle **VBAP**	000020

Für das Feld **TABKEY** (Key der geänderten Tabellenzeile) ergibt sich auf Basis der Beispielwerte gemäß obiger Konfiguration zum einen der konkatenierte Wert 800#-#0000004972#-#000020 und zum anderen 800#-#0000004972.

Der folgende Auszug aus der Ausgabedatei zeigt ein mögliches System-Event:

```
<event>
  <attribute type="CH_DOCS-CHANGE_USER_1">
      sapuser
  </attribute>
  <attribute type="CH_DOCS-CHANGE_USER_2">
      ANFEL
  </attribute>
  <attribute type="CH_DOCS-FIRST_CHANGE_DATE">
      20030919
  </attribute>
  <attribute type="CH_DOCS-FIRST_CHANGE_TIME">
      181020
  </attribute>
  <attribute type="CH_DOCS-LAST_CHANGE_DATE">
      20030930
  </attribute>
  <attribute type="CH_DOCS-NUM_OF_CHANGES_USER_1">
      1
  </attribute>
```

```
  <attribute type="CH_DOCS-NUM_OF_CHANGES_USER_2">
      3
  </attribute>
  <attribute type="CH_DOCS-SUM_OF_CHANGES">
      4
  </attribute>
  <attribute type="VBAP-ERDAT">20030919</attribute>
  <attribute type="VBAP-ERNAM">ANFEL</attribute>
  <attribute type="VBAP-ERZET">174938</attribute>
  <attribute type="VBAP-POSNR">000010</attribute>
  <attribute type="VBAP-VBELN">0000007500</attribute>
</event>
```

6.3.11.3 Auslesen des ersten Änderungsbeleges eines Tabellenfeldes

Mit der Klasse **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZFirstFieldChange_sap2ppm** lesen Sie Informationen zur ersten Änderung eines Tabellenfeldes aus einem SAP-System aus. Sie geben die Klasse in der Tabellenkonfiguration im XML-Element **table** mit dem XML-Attribut **classtouse** an.

BERECHNUNG

Für einige Belege im SAP-System gibt es nur ein Feld für das letzte Änderungsdatum. Der initiale Wert dieses Feldes im ersten Änderungsbeleg wird als Erstelldatum des Beleges interpretiert. Dieser Wert wird mit der zu verwendenden Klasse aus den Änderungsbelegtabellen **CDHDR** und **CDPOS** gelesen.

DIE BEDINGUNGEN FÜR DIE VERWENDUNG DER KLASSE MÜSSEN FOLGENDERMAßEN AUSSEHEN:

```
<table name="name" classtouse="com.idsscheer.ppm.
                          xmlextractortools.
                          extractor.sap2ppm.
                          ZFirstFieldChange_sap2ppm">
  <booleancondition>
    <condition fieldname="OBJECTCLAS" ...>
      ...
    </condition>
    <condition fieldname="OBJECTID" ...>
      ...
    </condition>
    <condition fieldname="TABNAME" ...>
      ...
    </condition>
    <condition fieldname="TABKEY" ...>
      ...
    </condition>
    <condition fieldname="FNAME" ...>
      ...
    </condition>
  </booleancondition>

  <pkfield name="OBJECTCLAS"/>
  <pkfield name="OBJECTID"/>
```

```
  <pkfield name="CHANGENR"/>
  <pkfield name="TABNAME"/>
  <pkfield name="TABKEY"/>
  <pkfield name="FNAME"/>
  <pkfield name="CHNGIND"/>

  <pkfield name="<Feld, das im Tabkey verwendet wird>" ↵
    fktablename="<Fremdschlüsseltabelle, ↵
                aus der das Feld gelesen wird>" ↵
    fkfieldname="<Fremdschlüsseltabellenfeld>"/>
  ... Weitere Felder, die im Tabkey verwendet werden
  <fieldtoread name="CDPOS-VALUE_OLD"/>
  ... Sonstige Felder aus der CDPOS-Tabelle
  <fieldtoread name="CDHDR-UDATE"/>
  ... Sonstige Felder aus der CDHDR-Tabelle
</table>
```

Die Zusammensetzung der Wertangaben der Felder **OBJECTID** und **TABKEY** ist die Gleiche wie bei der Klasse **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm. ↵ ZChangeDocTable_sap2ppm** (siehe Kapitel **System-Events inkl. Änderungen auslesen** (Seite 42)).

Beispiel (für SAP-Konfiguration)

Mit dieser Konfiguration soll ausgelesen werden, wann und wer zuletzt das Feld mit dem Freigabestatus von Bestellanforderungen geändert hat:

```
<table name="ORIGINAL_ERDAT" classtouse="com.idsscheer.ppm. ↵
                  xmlextractortools.extractor.sap2ppm. ↵
                         ZFirstFieldChange_sap2ppm">
  <booleancondition>
    <condition fieldname="OBJECTCLAS" logicaloperator="eq">
      <value>EINKBELEG</value>
    </condition>
    <condition fieldname="OBJECTID" ↵
             logicaloperator="eq_concatEventAttrValues">
      <value>EKPO-EBELN</value>
    </condition>
    <condition fieldname="TABNAME" logicaloperator="eq">
      <value>EKPO</value>
    </condition>
    <condition fieldname="TABKEY" ↵
            logicaloperator="eq_concatEventAttrValues">
      <value>MANDANT#-#EKPO-EBELN#-#EKPO-EBELP</value>
    </condition>
    <condition fieldname="FNAME" logicaloperator="eq">
      <value>AEDAT</value>
    </condition>
  </booleancondition>

  <pkfield name="OBJECTCLAS"/>
  <pkfield name="OBJECTID"/>
  <pkfield name="CHANGENR"/>
  <pkfield name="TABNAME"/>
  <pkfield name="TABKEY"/>
  <pkfield name="FNAME"/>
  <pkfield name="CHNGIND"/>

  <pkfield name="EBELN" fktablename="EKPO" ↵
                   fkfieldname="EBELN"/>
```

```
  <pkfield name="EBELP" fktablename="EKPO" ↵
                   fkfieldname="EBELP"/>

  <fieldtoread name="CDPOS-VALUE_OLD"/>
</table>
```

Das Ergebnis der Wertextraktion mit dieser Tabellenkonfiguration könnte in einem erzeugten System-Event bspw. folgendermaßen aussehen:

```
<event>
  ...
  <attribute type="ORIGINAL_ERDAT-VALUE_OLD">
    20010402
  </attribute>
  ...
</event>
```

6.3.11.4 Auslesen des letzten Änderungsbeleges eines Tabellenfeldes

Mit der Klasse **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm. ↵ ZLastFieldChange_sap2ppm** lesen Sie Informationen zu letzten Änderungen eines Beleges aus einem SAP-System aus. Konfiguration und Berechnung der Klasse erfolgen genau so wie bei der Klasse **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm. ↵ ZFirstFieldChange_sap2ppm** (siehe Kapitel **Auslesen des ersten Änderungsbeleges eines Tabellenfeldes** (Seite 46)) mit dem Unterschied, dass der letzte Änderungsbeleg statt des Ersten zur Ermittlung der Daten verwendet wird.

6.3.11.5 Auslesen des letzten Änderungsbeleges mehrerer Tabellenfelder

Informationen zu Änderungsbelegen werden in den Änderungsbelegtabellen **CDHDR** (Änderungsbelegkopf) und **CDPOS** (Änderungsbelegpositionen) gespeichert. Das Auslesen solcher Informationen ist mittels einer geeigneten Konfiguration der jeweiligen Datentabelle (XML-Element **table**) möglich. Dazu ist die Klasse **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm. ZChangeDocTableNew_sap2ppm** zu verwenden.

Beispiel (Auszug aus der Tabellenkonfiguration, allgemeines Dateigerüst)

Eine Konfiguration mit Bedingungen, die in einem **table**-Element angegeben werden müssen, könnte bspw. folgendermaßen aussehen:

```
...
<table name="tablename" classtouse="com.idsscheer.ppm.
               xmlextractortools.extractor.sap2ppm.
                           ZChangeDocTableNew_sap2ppm">
  <booleancondition>
    <condition fieldname="OBJECTCLAS" logicaloperator="...">
      ...
    </condition>
    <condition fieldname="OBJECTID" logicaloperator="...">
      ...
    </condition>
```

```
    <condition fieldname="TABNAME" logicaloperator="...">
      ...
    </condition>
    <condition fieldname="TABKEY" logicaloperator="...">
      ...
    </condition>
    <condition fieldname="FNAME" logicaloperator="in">
      ...
    </condition>
</booleancondition>

<pkfield name="OBJECTCLAS"/>
<pkfield name="OBJECTID"/>
<pkfield name="CHANGENR"/>
<pkfield name="TABNAME"/>
<pkfield name="TABKEY"/>
<pkfield name="FNAME"/>
<pkfield name="CHNGIND"/>

<pkfield name="<Feld, das in TABKEY verwendet wird>"
        fktablename="<Fremdschlüsseltabelle, aus
                  der das Feld gelesen wird>"
        fkfieldname="<Fremdschlüsseltabellenfeld>"/>
... weitere Felder, die in TABKEY verwendet werden
<fieldtoread name="CDPOS-VALUE_OLD"/>
... sonstige Felder aus der CDPOS-Tabelle
<fieldtoread name="CDHDR-UDATE"/>
... sonstige Felder aus der CDHDR-Tabelle
</table>
...
```

Die Zusammensetzung der Wertangaben der Felder **OBJECTID** und **TABKEY** ist die Gleiche wie bei der Klasse **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZChangeDocTable_sap2ppm**. Nachfolgend sehen Sie ein konkreteres Beispiel.

Beispiel (Auszug aus einer Tabellenkonfiguration aus Configuration Package PPM4MM)

Mit der beispielhaften Konfiguration soll ausgelesen werden, wann zuletzt eines der Sperrgrundfelder der logistischen Rechnungen geändert wurde:

```
...
  <table name="CDPOS" classtouse="com.idsscheer.ppm.
              xmlextractortools.extractor.sap2ppm.
                           ZChangeDocTableNew_sap2ppm">
    <booleancondition>
      <condition fieldname="OBJECTCLAS" logicaloperator="eq">
        <value>INCOMINGINVOICE</value>
      </condition>
      <condition fieldname="OBJECTID"
                logicaloperator="eq_concatEventAttrValues">
        <value>RSEG-BELNR#-#RSEG-GJAHR</value>
      </condition>
      <condition fieldname="TABNAME" logicaloperator="eq">
        <value>RSEG</value>
      </condition>
      <condition fieldname="TABKEY"
                logicaloperator="eq_concatEventAttrValues">
        <value>   #-#RSEG-BELNR#-#
                          RSEG-GJAHR#-#RSEG-BUZEI</value>
      </condition>
      <condition fieldname="FNAME" logicaloperator="in">
        <value>SPGRP</value>
```

```
        <value>SPGRM</value>
        <value>SPGRT</value>
        <value>SPGRG</value>
        <value>SPGRV</value>
        <value>SPGRQ</value>
        <value>SPGRS</value>
        <value>SPGRC</value>
      </condition>
    </booleancondition>
    <pkfield name="OBJECTCLAS"/>
    <pkfield name="OBJECTID"/>
    <pkfield name="CHANGENR"/>
    <pkfield name="TABNAME"/>
    <pkfield name="TABKEY"/>
    <pkfield name="FNAME"/>
    <pkfield name="CHNGIND"/>
    <pkfield name="BELNR" fktablename="RSEG"
                                fkfieldname="BELNR"/>
    <pkfield name="GJAHR" fktablename="RSEG"
                                fkfieldname="GJAHR"/>
    <pkfield name="BUZEI" fktablename="RSEG"
                                fkfieldname="BUZEI"/>
  </table>
...
```

Ein mit dieser Beispielkonfiguration erzeugtes System-Event könnte bspw. folgendermaßen aussehen:

```
...
<event>
  <attribute type="CDPOS-CHANGE_USER_1">HUETT</attribute>
  <attribute type="CDPOS-FIRST_CHANGE_DATE">20010220
  </attribute>
  <attribute type="CDPOS-FIRST_CHANGE_TIME">154943
  </attribute>
  <attribute type="CDPOS-LAST_CHANGE_DATE">20010220
  </attribute>
  <attribute type="CDPOS-LAST_CHANGE_TIME">154943
  </attribute>
  <attribute type="CDPOS-NUM_OF_CHANGES_USER_1">1
  </attribute>
  <attribute type="CDPOS-SUM_OF_CHANGES">1</attribute>
  ...
<event>
...
```

6.3.12 Auslesen der ersten bzw. letzten Zeile einer Sortierung

Die zu verwendende Java-Klasse **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZSortWithInteger_sap2ppm** liest Datensätze aus Tabellen aus, sortiert diese mithilfe der angegebenen Sortierkriterien und schreibt entweder den ersten oder letzten Datensatz in die zu erzeugenden System-Events. Die Klasse ist im Attribut **classtouse** des XML-Elements **table** anzugeben.

Beispiel

Statushistorienbelege stellen eine besondere Belegart im SAP R/3-System dar. Diese Belege sind in der Tabelle **JCDS** gespeichert. Mit der Klasse soll die letzte Änderung hinsichtlich eines bestimmten Statustyps ermittelt und in die entsprechenden System-Events geschrieben werden. Dazu sollen aus der Tabelle **AUFK** alle Auftragsnummern gelesen werden. Zusätzlich sollen zu jeder Auftragsnummer, zu der es Statusinformationen gibt, aus der Tabelle **JCDS** alle Einträge zu dieser Auftragsnummer mit dem Prefix **OR** und dem Status **I0043** (Feld **STAT**) gelesen werden. Diese sollen nach dem Erfassungszeitpunkt (**UDATE**, **UTIME**) sortiert werden und es soll derjenige Datensatz ausgelesen werden, der zuletzt erfasst wurde. Von diesem Datensatz sollen alle Feldwerte in das jeweilige System-Event geschrieben werden.

Der folgende Dateiauszug veranschaulicht die Konfiguration zum Auslesen von Statusinformationen aus der Tabelle **JCDS** unter den o. g. Bedingungen:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
             'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
  <configuration name="AUFK_JCDS">
    <docspec>
      <doctable name="AUFK">
        <pkfield name="AUFNR"/>
      </doctable>
    </docspec>
    <table name="JCDS_I0043" tablename="JCDS"
             classtouse="com.idsscheer.ppm.
               xmlextractortools.extractor.
           sap2ppm.ZSortWithInteger_sap2ppm">
      <parameter name="SORTCRITERION">
        <value>UDATE</value>
        <value>UTIME</value>
      </parameter>
      <parameter name="USE">
        <value>MAX</value>
      </parameter>
      <condition fieldname="STAT"
                          logicaloperator="eq">
        <value>I0043</value>
      </condition>
      <pkfield name="OBJNR" fktablename="AUFK"
                           fkfieldname="AUFNR">
        <prefix>
          <value>OR</value>
        </prefix>
      </pkfield>
```

```
     <fieldtoread name="CDTCODE"/>
     <fieldtoread name="CHGNR"/>
     <fieldtoread name="CHIND"/>
     <fieldtoread name="INACT"/>
     <fieldtoread name="OBJNR"/>
     <fieldtoread name="STAT"/>
     <fieldtoread name="TCODE"/>
     <fieldtoread name="USNAM"/>
    </table>
  </configuration>
</xmlextractor_tableconfiguration>
```

Ein System-Event, das mit dieser Konfiguration ausgelesen wird, sieht bspw. folgendermaßen aus:

```
<event>
  <attribute type="AUFK-AUFNR">000000800002</attribute>
  <attribute type="JCDS_I0043-CDTCODE"></attribute>
  <attribute type="JCDS_I0043-CHGNR">002</attribute>
  <attribute type="JCDS_I0043-CHIND">U</attribute>
  <attribute type="JCDS_I0043-INACT">X</attribute>
  <attribute type="JCDS_I0043-OBJNR">
                  OR000000800002
  </attribute>
  <attribute type="JCDS_I0043-STAT">I0043</attribute>
  <attribute type="JCDS_I0043-TCODE">KOK2</attribute>
  <attribute type="JCDS_I0043-UDATE">19961217</attribute>
  <attribute type="JCDS_I0043-USNAM">MUELLERJ</attribute>
  <attribute type="JCDS_I0043-UTIME">192122</attribute>
</event>
```

Die Sortierkriterien (hier: **UDATE**, **UTIME**) sind automatisch Bestandteil der System-Event-Spezifikation.

Die Vorlage für die Erstellung der gezeigten Konfiguration sieht folgendermaßen aus (welche XML-Elemente bzw. -Attribute optional sind, können Sie der nachfolgenden Tabelle entnehmen):

```
<table name="..." tablename="..." classtouse=
                    "com.idsscheer.ppm.
               xmlextractortools.extractor.
          sap2ppm.ZSortWithInteger_sap2ppm">
  <parameter name="SORTCRITERION">
    <value>...</value>
    <value>...</value>
    ...
  </parameter>
  <parameter name="USE">
    <value>...</value>
  </parameter>
  ...
  Siehe Kap.
  R/3-Tabellenkonfiguration (System-Event-Spezifikation) (Seite 13)
  ...
</table>
```

Folgende Tabelle zeigt noch einmal die wichtigsten Konfigurationseinträge der obigen **table**-Definition:

XML-Element /-Attribut	**Wert**: **Beschreibung**
name	Angegebener Name wird den gelesenen Quellsystemattributen als Präfix vorangestellt
tablename (optional)	Tabelle, aus der Informationen gelesen werden sollen. Vorgabewert: Wert aus XML-Attribut **name**
classtouse	**com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZSortWithInteger_sap2ppm**: Zu verwendende Java-Klasse
parameter name	
SORTCRITERION	Erster anzugebender Parameter. Mindestens ein XML-Element **value** muss angegeben werden (Feldname des Sortierkriteriums). Die angegebenen Felder dürfen nur Integer-Werte enthalten. Sind mehrere Sortierkriterien angegeben, erfolgt eine Priorisierung von oben nach unten, d. h. die Datensätze werden zunächst anhand des ersten (obersten) Kriteriums verglichen, konnte keine vollständige Sortierung erstellt werden, dann anhand des zweiten Kriteriums usw.
USE	Zweiter anzugebender Parameter. Genau ein XML-Element **value** muss angegeben werden. Mögliche Werte: **MIN** (es wird der Datensatz mit dem kleinsten Integer-Wert des Sortierkriteriums gewählt) **MAX** (es wird der Datensatz mit dem größten Integer-Wert des Sortierkriteriums gewählt) Aus der auf Basis der angegebenen Sortierkriterien ermittelten Liste von Datensätzen wird entweder der erste oder letzte Datensatz zum Auslesen gewählt.

Setzen Sie die Klasse zum Auslesen der ersten bzw. letzten Zeile aus einer Sortierung nur begrenzt ein, da es aufgrund der Sortier- und Selektionsvorgänge zu Leistungs- und Speichereinbußen kommt.

6.3.13 Attribute mit invertiertem Datum erzeugen

Die Java-Klasse **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZTableInvertDates_sap2ppm** kann auch verwendet werden, um während des Auslesevorgangs aus einem SAP-System neue System-Event-Attribute mit invertiertem Datum zu erzeugen. Dazu werden die Werte bereits ausgelesener Datumsattribute invertiert. Ein invertiertes Datum wird berechnet, indem von 99999999 der Wert des Datums im SAP-Format, z. B. 20110529 für den 29. Juli 2011, abgezogen wird. Die neu erzeugten System-Event-Attribute können in weiteren Datentabellenkonfigurationen (XML-Element **table**) referenziert werden.

Beispiel

```
<docspec>
    <doctable name="VBAP">
      <pkfield name="VBELN"/>
      <pkfield name="POSNR"/>
      <pkfield name="ERDAT"/>
      <pkfield name="AEDAT"/>
      <pkfield name="ABDAT"/>
    </doctable>
</docspec>
<table name="VBAP_DATES_INVERTED" tablename="VBAP_DATES_INVERTED"
classtouse="com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZTableInvertDa
tes_sap2ppm">
    <pkfield name="ERDAT" fktablename="VBAP" fkfieldname="ERDAT" />
    <pkfield name="AEDAT" fktablename="VBAP" fkfieldname="AEDAT" />
</table>
```

Wenn Sie die Klasse **ZTableInvertDates_sap2ppm** verwenden, müssen Sie kein fieldtoread-Element angeben. Eventuell angegebene fieldtoread-Elemente werden ignoriert. Die neuen System-Event-Attribute werden aus dem Wert des Attributs **name** des Elements **table** und dem Wert des Attributs **name** des Elements **pkfield** zusammengesetzt. Im Beispiel werden zwei neue System-Event-Attribute mit dem Namen **VBAP_DATES_INVERTED-ERDAT** und **VBAP_DATES_INVERTED-AEDAT** erzeugt.

Bei den pkfield-Elementen können die üblichen Operationen **fkpart**, **prefix** und **postfix** verwendet werden.

Beispiel

Ein mit obiger Konfiguration erzeugtes System-Event könnte beispielsweise wie folgt aussehen:

```
...
<event>
    <attribute type="VBAP-ABDAT">00000000</attribute>
    <attribute type="VBAP-AEDAT">19970127</attribute>
    <attribute type="VBAP-ERDAT">19970121</attribute>
    <attribute type="VBAP-POSNR">000010</attribute>
    <attribute type="VBAP-VBELN">0000004974</attribute>
    <attribute type="VBAP_DATES_INVERTED-AEDAT">80029872</attribute>
    <attribute type="VBAP_DATES_INVERTED-ERDAT">80029878</attribute>
</event>
...
```

Die Klasse **ZTableInvertDates_sap2ppm** kann verwendet werden, um ein invertiertes Datum in ein normales Datum im SAP-Format zurück zu wandeln.

Beispiel

```
<docspec>
    <doctable name="VBAP">
      <pkfield name="VBELN"/>
      <pkfield name="POSNR"/>
      <pkfield name="ERDAT"/>
      <pkfield name="AEDAT"/>
      <pkfield name="ABDAT"/>
      </doctable>
</docspec>
<table name="VBAP_DATES_INVERTED" tablename="VBAP_DATES_INVERTED"
classtouse="com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZTableInvertDa
tes_sap2ppm">
    <pkfield name="ERDAT" fktablename="VBAP" fkfieldname="ERDAT" />
    <pkfield name="AEDAT" fktablename="VBAP" fkfieldname="AEDAT" />
</table>
<table name="VBAP_DATES_INVERT_AGAIN" tablename="VBAP_DATES_INVERTED"
classtouse="com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZTableInvertDa
tes_sap2ppm">
    <pkfield name="ERDAT" fktablename="VBAP_DATES_INVERTED" fkfieldname="ERDAT" />
    <pkfield name="AEDAT" fktablename="VBAP_DATES_INVERTED" fkfieldname="AEDAT" />
    <pkfield name="ABDAT" fktablename="VBAP_DATES_INVERTED" fkfieldname="ABDAT" />
</table>
```

Liefert z. B. folgendes Ergebnis:

```
...
<event>
    <attribute type="VBAP-ABDAT">00000000</attribute>
    <attribute type="VBAP-AEDAT">19970127</attribute>
    <attribute type="VBAP-ERDAT">19970121</attribute>
    <attribute type="VBAP-POSNR">000010</attribute>
    <attribute type="VBAP-VBELN">0000004974</attribute>
    <attribute type="VBAP_DATES_INVERTED-AEDAT">80029872</attribute>
    <attribute type="VBAP_DATES_INVERTED-ERDAT">80029878</attribute>
    <attribute type="VBAP_DATES_INVERT_AGAIN-AEDAT">19970127</attribute>
```

```
    <attribute type="VBAP_DATES_INVERT_AGAIN-ERDAT">19970121</attribute>
</event>
...
```

6.3.14 Vervielfältigung von System Events in Tabellen

Beim Auslesen aus einem SAP-System gibt es die Möglichkeit, basierend auf einer System-Event-Tabelle weitere System-Events zu erzeugen.

Auf Quellsystemebene kann es vorkommen, dass es zwischen zwei Tabellen eine 1:n-Beziehung gibt, die beim Ermitteln der System-Events nicht aufgelöst werden kann. Mit Hilfe der Klasse **com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZTableMultiplyEvents_sap2ppm** können aus einem System Event mehrere System Events erzeugt und somit die 1:n-Beziehung aufgelöst werden.

Beispiel

```
<configuration name="MultiplyEvents">
    <docspec>
      <doctable name="VBAK" tablename="VBAK">
       <condition fieldname="VBELN" logicaloperator="in">
         <value>0000006662</value>
         <value>0000006741</value>
       </condition>

       <pkfield name="VBELN"/>
      </doctable>
    </docspec>

    <table name="VBAP"
    classtouse="com.idsscheer.ppm.xmlextractortools.extractor.sap2ppm.ZTableMult
    iplyEvents_sap2ppm">
      <pkfield name="VBELN" fktablename="VBAK" fkfieldname="VBELN"/>
      <fieldtoread name="POSNR"/>
      <fieldtoread name="MATNR"/>
    </table>

    <table name="MAKT">
       <condition fieldname="SPRAS" logicaloperator="eq">
        <value>de</value>
       </condition>

    <pkfield name="MATNR" fktablename="VBAP" fkfieldname="MATNR"/>
      <fieldtoread name="MAKTX"/>
    </table>
</configuration>
```

Folgend die zugehörige System-Event-Ausgabedatei.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
<event>
  <attribute type="VBAK-VBELN">0000006662</attribute>
</event>
<event>
```

```
  <attribute type="MAKT-MAKTX">Flatscreen MS 1460 P</attribute>
  <attribute type="VBAK-VBELN">0000006741</attribute>
  <attribute type="VBAP-MATNR">M-06</attribute>
  <attribute type="VBAP-POSNR">000010</attribute>
</event>
<event>
  <attribute type="MAKT-MAKTX">Flatscreen MS 1775P</attribute>
  <attribute type="VBAK-VBELN">0000006741</attribute>
  <attribute type="VBAP-MATNR">M-10</attribute>
  <attribute type="VBAP-POSNR">000020</attribute>
</event>
<event>
  <attribute type="MAKT-MAKTX">MAG PA/DX 175</attribute>
  <attribute type="VBAK-VBELN">0000006741</attribute>
  <attribute type="VBAP-MATNR">M-14</attribute>
  <attribute type="VBAP-POSNR">000030</attribute>
</event>
<event>
  <attribute type="MAKT-MAKTX">Jotachi SN4500</attribute>
  <attribute type="VBAK-VBELN">0000006741</attribute>
  <attribute type="VBAP-MATNR">M-18</attribute>
  <attribute type="VBAP-POSNR">000040</attribute>
</event>
</eventlist>
```

Folgendes Beispiel, ohne die beschriebene Klasse, verdeutlicht den Vorgang bei der Extraktion.

```
<configuration name="MultiplyEvents_Doctable_Only">
    <docspec>
      <doctable name="VBAK" tablename="VBAK">
       <condition fieldname="VBELN" logicaloperator="in">
        <value>0000006662</value>
        <value>0000006741</value>
       </condition>

       <pkfield name="VBELN"/>
      </doctable>
    </docspec>

    <table name="VBAP" >
      <pkfield name="VBELN" fktablename="VBAK" fkfieldname="VBELN"/>
      <fieldtoread name="POSNR"/>
      <fieldtoread name="MATNR"/>
    </table>

    <table name="MAKT">
       <condition fieldname="SPRAS" logicaloperator="eq">
        <value>de</value>
       </condition>

      <pkfield name="MATNR" fktablename="VBAP" fkfieldname="MATNR"/>
      <fieldtoread name="MAKTX"/>
    </table>
</configuration>
```

Da einem Eintrag in der Tabelle **VBAK** mehrere Einträge in der Tabelle **VBAP** zugeordnet sind, wird aus der Tabelle **VBAP** nur eine einzige, zufällige Zeile gelesen.

Folgend die zugehörige System-Event-Ausgabedatei.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
<event>
  <attribute type="VBAK-VBELN">0000006662</attribute>
</event>
<event>
  <attribute type="MAKT-MAKTX">Flatscreen MS 1775P</attribute>
  <attribute type="VBAK-VBELN">0000006741</attribute>
  <attribute type="VBAP-MATNR">M-10</attribute>
  <attribute type="VBAP-POSNR">000020</attribute>
</event>
</eventlist>
```

6.3.15 Felder einzelner Tabellen für Data Analytics extrahieren

Um für Data Analytics eine einfache Datenextraktion zu ermöglichen, gibt es die Möglichkeit den kompletten Inhalt einer Quellsystemtabelle in eine Datei im Event-Format zu extrahieren, die dann in einen Analyseraum von Data Analytics importiert werden kann.

Sie können auch die auszulesenden Daten einschränken (Seite 60), indem Sie Bedingungen zur Datenextraktion definieren.

Die auszulesende Tabelle wird nicht über die Tabellenkonfiguration konfiguriert, sondern in der Datenquellendatei selbst. Dazu gibt es in der Datei **datasource.dtd** folgende Einträge:

XML-Element/-Attribut	Beschreibung
analysistype	XML-Attribut: Muss den Wert **DATA_ANALYTICS** haben.
realmtable	Das umfassende XML-Element für die Konfiguration der Data-Analytics-Datenquelle
tablename	XML-Attribut des Elements **realmtable**: Name der Tabelle im Quellsystem
sourcetable	Das umfassende XML-Element für die Konfiguration der Data-Analytics-Quellsystemtabelle. Muss mindestens ein Element **sourcefield** enthalten.
tablename	XML-Attribut des Elements **sourcetable**: Name der Tabelle im Quellsystem
sourcefield	Enthält den Namen des Feldes der Quellsystemtabelle

Das Attribut **analysistype** des XML-Elements **datasource** muss den Wert **DATA_ANALYTICS** haben, wenn durch das Element **<realmtable>** eine Analyseraum-Tabelle angegeben ist (PROCESS der Standardwert).

Das Attribut **tablename** des Elements **<realmtable>** gibt den Tabellennamen der Zieltabelle in der Analyseraum-Konfiguration an. Dieser Tabellenname hat auf die Extraktion selbst keinen Einfluss, sondern wird nur vom PPM-Import ausgewertet.

Weitere Informationen zum Datenimport für Data Analytics erhalten Sie im Benutzerhandbuch PPM Data Analytics.

Das XML-Element **<realmtable>** enthält das optionale Element **<sourcetable>**, das die zu extrahierende Tabelle angibt. Die auszulesenden Spalten dieser Tabelle müssen im Element **<sourcefield>** angegeben werden. Das Element **<sourcetable>** ist optional. Beim JDBC- oder SAP-Extraktor müssen genau eine Quelltabelle und mindestens eine Quellspalte angegeben werden, andernfalls wird während des Parsen der Datenquellendatei eine Fehlermeldung ausgegeben.

In einer Datenquellendefinition darf maximal eine Tabelle stehen. Es ist nicht möglich die Anzahl der Zeilen einzuschränken, sondern es werden immer alle Zeilen ausgelesen, einschließlich aller Zeilen mit gleichen Werten an den auszulesenden Spalten. Sollen beispielsweise die Spalten **Vorname** und **Nachname** ausgelesen werden und die Tabelle enthält zehn Einträge mit **Peter** und **Schmidt**, dann werden dafür auch zehn Events mit gleichen Attributwerten erzeugt.

Das folgende Beispiel verdeutlicht die Konfiguration:

```
<realmtable tablename="COMPANY_EMPLOYEE">
    <sourcetable tablename="EMPLOYEE">
      <sourcefield>EMPLOYEE_ID</sourcefield>
      <sourcefield>NAME</sourcefield>
    </sourcetable>
    ...
</realmtable>
<dataextraction>
    <outputfilename>..\custom\testclient\data\employee.xml</outputfilename>
</dataextraction>
...
<systemconfig>..\custom\testclient\SourceSystemConfig.xml</systemconfig>
```

Im Unterschied zum herkömmlichen JDBC- oder SAP-Extraktor erhalten die Attribute in der Event-Ausgabedatei keinen Tabellennamen als Präfix. Wird beispielsweise die Tabelle **EMPLOYEE** ausgelesen, erzeugte der Extraktor normalerweise Events der Art **<tabellenname>-<spaltenname>**:

```
<event>
    <attribute type="EMPLOYEE-EMPLOYEE_ID">4711</attribute>
    <attribute type="EMPLOYEE-NAME">Schmidt</attribute>
</event>
```

Das Auslesen einer Tabelle per Element **<realmtable>** erzeugt hingegen nur Events ohne Tabellennamen:

```
<event>
    <attribute type="EMPLOYEE_ID">4711</attribute>
    <attribute type="NAME">Schmidt</attribute>
</event>
```

NULL-WERTE IM MODUS DATA ANALYTICS

Ist beim Auslesen einer Analyseraum-Tabelle der Wert einer Spalte **null**, wird dieser nicht in das Event geschrieben. Gibt es zum obigen Beispiel noch beispielsweise eine Zeile **EMPLOYEE_ID = 4712** ohne Nachnamen, erzeugt der Extraktor folgende Events.

```
<event>
    <attribute type="EMPLOYEE_ID">4711</attribute>
    <attribute type="NAME">Schmidt</attribute>
</event>
<event>
    <attribute type="EMPLOYEE_ID">4712</attribute>
</event>
```

Sind hingegen alle auszulesenden Spaltenwerte **null**, wird das Event mit leeren Attributen rausgeschrieben:

```
<event>
    <attribute type="EMPLOYEE_ID">4711</attribute>
    <attribute type="NAME">Schmidt</attribute>
</event>
<event>
    <attribute type="EMPLOYEE_ID"></attribute>
    <attribute type="NAME"></attribute>
</event>
```

Gibt es mehrere solcher Zeilen, werden diese – im Unterschied zum herkömmlichen Auslesevorgang (analysistype=PROCESS) – entsprechend oft übernommen, so dass in der Event-Datei immer genauso viele <event>-Elemente stehen, wie es Zeilen in der Datentabelle des Quellsystems gibt.

6.3.15.1 Datenextraktion einschränken

Wenn Sie nur einen bestimmten Teil des Tabelleninhaltes auslesen möchten, können Sie Bedingungen zur Datenextraktion definieren. Sie können die auszulesende Datenmenge mittels Zeitstempel oder Integerwert (z. B einer Sequenz) beschränken.

Geben Sie dazu in der Datenquellen-Konfiguration die gewünschte Bedingung mittels des Elements **condition** und des Attributs **dataextractiontype** wie folgt an.

```
<datasource name="VBAP" type="JDBC" analysistype="DATA_ANALYTICS"
dataextractiontype="TIME_BASED">
      <realmtable tablename="VBAP">
            <sourcetable tablename="VBAP2">
                  <condition logicaloperator="char_creationtimestamp"
fieldname="AEDAT">
                        <value>yyyy-MM-dd</value>
                  </condition>
                  <sourcefield>AEDAT</sourcefield>
                  <sourcefield>ERDAT</sourcefield>
                  <sourcefield>ERNAM</sourcefield>
                  <sourcefield>ERZET</sourcefield>
                  <sourcefield>MATKL</sourcefield>
                  <sourcefield>MATNR</sourcefield>
                  <sourcefield>WERKS</sourcefield>
            </sourcetable>
      </realmtable>
...
</datasource>
```

Das Attribut **dataextractiontype** kann bei Data-Analytics-Datenquellen folgende Werte haben:

- COMPLETE: Es wird bei einem Zeitkriterium ab dem Datum **01.01.1990 00:00:00** bzw. bei einem Integer-Kriterium ab dem Wert **0** ausgelesen. Dies ist auch der Standard, wenn das Attribut nicht vorhanden ist.
- TIME_BASED: Es soll eine zeitstempelbasierte Datenextraktion durchgeführt werden.
- VALUE_BASED: Es soll eine wertbasierte Datenextraktion durchgeführt werden.

Es wird keine Überprüfung vorgenommen, ob die verwendete Datenextraktionsbedingung zu dem konfigurierten Datenextraktionstyp (dataextractiontype=) passt. Wird die Konfiguration mittels CTK erstellt, so wird die Korrektheit vom CTK sichergestellt. Wird die Datei von Hand konfiguriert und es wird eine nicht passende Bedingung verwendet, so kann es zu einem Fehler bei der Datenextraktion kommen.

6.4 Kommandozeilenprogramm

Die XML-Ausgabedatei(en) erzeugen Sie mit dem Kommandozeilenprogramm **runsap2ppm.bat**.

Der Aufruf des Programmes ohne Parameter oder mit **-h** oder **-?** gibt die Hilfe auf der Konsole aus, in der alle verfügbaren Optionen beschrieben sind.

6.4.1 Argumente des Kommandozeilenprogramms

6.4.1.1 Allgemeine Argumente

-VERSION

Gibt die Versionsnummer des verwendeten PPM Prozessextraktors aus. Andere angegebene Argumente werden ignoriert.

-INFORMATION YES|NO

Hier legen Sie fest, ob Informationen während des Imports ausgegeben werden sollen (yes) oder nicht (no). Voreingestellt ist **yes**.

-WARNING YES|NO

Hier legen Sie fest, ob Warnmeldungen während des Imports ausgegeben werden sollen (yes) oder nicht (no). Voreingestellt ist **yes**.

-ERROR YES|NO

Hier legen Sie fest, ob Fehlermeldungen während des Imports ausgegeben werden sollen (yes) oder nicht (no).Voreingestellt ist **yes**.

-PROTOCOLFILE <FILENAME>

Hier können Sie eine Protokolldatei festlegen, in die alle Meldungen während des Imports geschrieben werden. Wenn Sie eine Datei festlegen, werden am Bildschirm nur noch kritische Fehlermeldungen ausgegeben, die zum Programmabbruch führen.

-LANGUAGE <ISO-CODE>

Hier geben Sie die Sprache an, in der die Protokollinformationen ausgegeben werden sollen.

6.4.1.2 Quellsystemspezifische Argumente

-DATASOURCE <FILENAME> (OPTIONAL)

Hier geben Sie die Datenquelle an, die für den Auslesevorgang verwendet werden soll. Die XML-Datei enthält Angaben zu allen für den Extraktionsvorgang und die XML-Ausgabe zu verwendenden Dateien. Verwenden Sie diesen Parameter nicht zusammen mit einem der Parameter **-datasourcelist**, **-systemconfig**, **-tableconfig**, **-calcconfig**, **-outfile** bzw. **-nozip**, sondern anstelle dieser Parameter.

-DATASOURCELIST <FILENAME>

Mit Hilfe des Arguments -**datasourcelist** können mehrere Datenquellen auf einmal extrahiert werden. Dies entspricht dem mehrfachen, nacheinander Ausführen der Extraktion mittels des Arguments -**datasource**. Es werden nur SAP-Datenquellen extrahiert.

Sie auch Kapitel Extraktion mehrere Datenquellen (Seite 67).

-SYSTEMCONFIG <FILENAME> <CONFIGNAME>

Hier geben Sie den Namen der XML-Datei an, die die Systemkonfiguration(en) enthält. Der Inhalt der Datei ist quellsystemspezifisch und kann mehrere Konfigurationen enthalten. Im zweiten Argument wird der Name der zu verwendenden Konfiguration angegeben. Enthält die XML-Datei nur eine Konfiguration, wird diese automatisch verwendet. In diesem Fall müssen Sie keinen Konfigurationsnamen angeben.

-TABLECONFIG <FILENAME> <CONFIGNAME>

Hier geben Sie den Namen der XML-Datei an, die die Tabellen-konfiguration(en) enthält. Der Inhalt der Datei ist quellsystemspezifisch und kann mehrere Konfigurationen enthalten. Im zweiten Argument wird der Name der Tabellenkonfiguration angegeben. Enthält die XML-Datei nur eine Konfiguration, wird diese automatisch verwendet. In diesem Fall müssen Sie keinen Konfigurationsnamen angeben.

-CALCCONFIG <FILENAME> (OPTIONAL)

Hier geben Sie den Namen der XML-Datei an, mit der Sie Attribute der XML-Ausgabedatei ändern bzw. Attribute einschließlich Attributtransformationen hinzufügen können.

-BEGINDATE <DD.MM.YYYY> (OPTIONAL)

Hier geben Sie das Datum an, ab dem Daten aus dem Quellsystem gelesen werden sollen. Wenn Sie den Parameter nicht verwenden, wird der Wert des XML-Attributs **lastreaddate** aus der angegebenen Systemkonfiguration bzw. Datenquelle verwendet. Das SAP-Format **yyyymmdd** wird unterstützt.

-BEGINTIME <HH:MM:SS> (OPTIONAL)

Hier geben Sie die Uhrzeit an, ab der Daten aus dem Quellsystem gelesen werden sollen. Wenn Sie den Parameter und **-begindate** nicht verwenden, wird der Wert des XML-Attributs **lastreadtime** aus der angegebenen Systemkonfiguration bzw. Datenquelle verwendet. Wenn Sie den Parameter nicht verwenden, aber **-begindate** verwenden, wird der Vorgabewert **000000** gesetzt.

-ENDDATE <DD.MM.YYYY> (OPTIONAL)

Hier geben Sie das Datum an, bis zu dem Daten aus dem Quellsystem gelesen werden sollen. Wenn Sie den Parameter nicht verwenden, wird das aktuelle Datum verwendet. Das SAP-Format **yyyymmdd** wird unterstützt.

-ENDTIME <HH:MM:SS> (OPTIONAL)

Hier geben Sie die Uhrzeit an, bis zu der Daten aus dem Quellsystem gelesen werden sollen. Wenn Sie diesen Parameter bei Verwendung von
-enddate nicht verwenden, wird der Vorgabewert **235959** gesetzt. Wenn Sie den Parameter und **-enddate** nicht verwenden, wird die aktuelle Uhrzeit verwendet.

Wird PPM Process Extractor SAP-2-PPM ohne Angabe eines Zeitraumes gestartet, wird der Beginnzeitpunkt aus der angegebenen Konfigurationsdatei gelesen. Als Endzeitpunkt werden das aktuelle Datum und die aktuelle Uhrzeit verwendet.

-VALUECONSTRAINT <PARAMETER1> ... <PARAMETER4> (OPTIONAL)

Mit diesem Parameter schränken Sie die auszulesende Datenmenge anhand eines Integer-Kriteriums ein. Erlaubt sind die Vergleichsoperatoren "<", "<=", ">" bzw. ">=". Sie können den entsprechenden Feldwert mit einem Wert oder mit zwei Werten (Intervallangabe) vergleichen. Wurde bereits ein zuletzt gelesener Wert (**lastreadvalue**) in der Systemkonfiguration bzw. Datenquelle gespeichert, können Sie auch diesen Wert für den Vergleich verwenden (siehe Kap. **Fortlaufendes automatisiertes Auslesen** (Seite 65)).

Beispiele

```
-valueconstraint 25000020 "<="
```

Es werden nur die Datensätze ausgelesen, deren Integer-Wert des in der Event-Spezifikation angegebenen Feldes **kleiner** bzw. **gleich** dem Wert **25000020** ist.

```
-valueconstraint 15000020 ">" 25000020 "<"
```

Intervallangabe: Es werden nur die Datensätze ausgelesen, deren Integer-Wert des in der System-Event-Spezifikation angegebenen Feldes **kleiner** als der Wert **25000020** und **größer** als der Wert **15000020** ist.

-SAVE_VALUE_MINIMUM (OPTIONAL)

Mit diesem Parameter geben Sie an, dass beim einschränkenden Auslesen mit **-valueconstraint** der kleinste zuletzt gelesene Wert in der verwendeten Konfigurationsdatei (Systemkonfiguration bzw. Datenquelle) als **lastreadvalue** gespeichert wird. Vorgabewert: Speichern des größten Wertes

-CPD <INT> (OPTIONAL)

Mit diesem Parameter spezifizieren Sie die Anzahl der parallel auszulesenden System-Events (concurrently processed documents). Mit dem angegebenen Wert werden die Primärschlüsselfelder aus der mit dem XML-Element **doctable** in der Tabellenkonfiguration referenzierten Tabelle schrittweise ausgelesen und im Java-Speicher des Extraktors gehalten. Eine höhere Wertangabe führt zu einer besseren Performance des Auslesevorganges und zu einer effizienteren Speichernutzung.

Wenn Sie den Parameter nicht angeben, wird der empfohlene Standardwert 1000 verwendet.

-PING (OPTIONAL)

Mit diesem Parameter testen Sie die Verbindung zum angegebenen Quellsystem. Es werden keine Daten ausgelesen.

-RFC_BLOCKMODE {G|D|M} (OPTIONAL)

Mit diesem Parameter verwenden Sie die blockweise Datenextraktion (siehe Kapitel **Blockweises Auslesen großer Datenmengen** (Seite 38)). Den entsprechenden Modus geben Sie als Argument an. Verfügbare Blockmodi:

G (Interne Tabelle der Funktionsgruppe)
M (ABAP-Memory)
D (Datenbanktabelle)

-PROGRESS {YES|NO} (OPTIONAL)

Mit diesem Parameter geben Sie an, ob Sie zusätzliche Protokollmeldungen über den Fortschritt der blockweisen Datenextraktion ausgeben lassen wollen. Vorgabewert: **yes**

-RFC_CALLRETRYS <ANZAHL> (OPTIONAL)

Mit diesem Parameter geben Sie die Anzahl der erneuten Anfragen an das R/3-System beim Auftreten eines Fehlers oder Verbindungsabbruchs an. Bei Verwendung der Blockmodi **G** bzw. **M** muss die Anzahl **0** angegeben werden. Der Parameter ist global wirksam, nicht nur beim blockweisen Auslesen. Vorgabewert: **0**

-RFC_DELIMITER <ZEICHEN> (OPTIONAL)

Mit diesem Parameter geben Sie das Zeichen an, mit dem die ausgelesenen Werte getrennt werden sollen. Das Zeichen darf in den ausgelesenen Werten selbst nicht enthalten sein. Vorgabewert: **;**

-REMOVEEMPTY (OPTIONAL)

Mit diesem Parameter entfernen Sie ausgelesene Attribute ohne Werte vor der Attributtransformation.

-REMOVEZERODATES (OPTIONAL)

Mit diesem Parameter entfernen Sie Werte von SAP-Datumsfeldern, d. h. SAP-Tabellenfeldern des Typs **DATS** bzw. **D**, die nur Nullen enthalten. Die Werte werden direkt nach der Datenextraktion aus den Events entfernt.

Beispiel

```
<attribute type="VBAP-ABDAT">00000000</attribute>
```

6.4.1.3 Ausgabedateispezifische Argumente

-OUTFILE <FILENAME>

Mit diesem Parameter geben Sie den Namen der XML-Ausgabedatei an. Die Dateinamenerweiterung wird automatisch ergänzt. Standardmäßig wird die Datei als ZIP-Datei ausgegeben.

-OUTFILEENCODING <ENCODING> (OPTIONAL)

Mit diesem Parameter geben Sie das Encoding der XML-Ausgabedatei an. Vorgabewert: UTF-8

-NOZIP (OPTIONAL)

Mit diesem Parameter geben Sie an, dass die Ausgabedatei nicht als ZIP-Datei, sondern als XML-Datei ausgegeben wird.

-PIKIDATAMAPPING <FILENAME> <PCNAME> (OPTIONAL)

Mit diesem Parameter geben Sie im ersten Argument den Namen der XML-Datei an, die das Mapping enthält. Im zweiten Argument geben Sie den Namen der Kennzahlenreihe an, in der die prozessinstanzunabhängigen Kennzahlenreihen gespeichert werden sollen.

-DIMDATAMAPPING <FILENAME> <DIMNAME> (OPTIONAL)

Mit diesem Parameter geben Sie im ersten Argument den Namen der XML-Datei an, die das Mapping enthält. Im zweiten Argument geben Sie den Namen der Dimension an, für die die extrahierten Werte eingelesen werden sollen.

-SORTEVENTATTRIBUTES (OPTIONAL)

Mit diesem Parameter sortieren Sie die Quellsystemattribute in den System-Events anhand des Attributtyps in alphanumerischer Reihenfolge. Bei großen Datenmengen kann sich dieser Parameter negativ auf die Geschwindigkeit des Auslesevorgangs auswirken.

6.4.1.4 Fortlaufendes automatisiertes Auslesen

Sie können Daten lückenlos automatisiert auslesen, indem Sie die Parameter **-begindate**, **-begintime**, **-enddate**, **-endtime** auf der Kommandozeile weglassen bzw. **-valueconstraint** ohne Wert für den ersten Vergleichsoperator verwenden. Bei Einschränkungen des auszulesenden Datenbereichs mittels Integer-Werten wird entweder der kleinste oder größte

zuletzt gelesene Wert in der verwendeten Konfigurationsdatei gespeichert bzw. aktualisiert. Bei zeitlicher Einschränkung des auszulesenden Datenbereichs wird der Zeitpunkt der letzten Datenextraktion in der Konfigurationsdatei gespeichert bzw. aktualisiert.

Folgende Voraussetzungen müssen gegeben sein:

- In der System-Event-Spezifikation ist eine korrekte Konfiguration zu **creationtimestamp** bzw. **valueconstraint** für die auszulesenden Daten angegeben (siehe Kap. **Bedingungsoperatoren** (Seite 27))
- Sie verwenden für das fortlaufende automatisierte Auslesen immer den gleichen Kommandozeilenaufruf, z. B.:

```
runsap2ppm -datasource datasource.xml -valueconstraint ">"
```

- Achten Sie bei der Verwendung des Parameters **-valueconstraint** darauf, in allen Auslesevorgängen entweder nur den größten zuletzt gelesenen Wert zu speichern (vorgabemäßig) oder mittels Angabe von **-save_value_minimum** den kleinsten zuletzt gelesenen Wert. Der Wert für **lastreadvalue** in der Konfigurationsdatei wird nur aktualisiert, wenn bei Ausführen der Kommandozeile ohne Parameter **-save_value_minimum** der größte zuletzt gelesene Wert größer als der bisher gespeicherte Wert ist, bzw. bei Verwendung von **-save_value_minimum**, wenn der kleinste zuletzt gelesene Wert kleiner als der bisher gespeicherte Wert ist.

Beispiel 1 (lastreadvalue mit Vorgabewert "0")

`-valueconstraint ">="`

Es werden nur die Datensätze ausgelesen, deren Integer-Wert zum Einschränken des auszulesenden Datenbereichs **größer** bzw. **gleich** dem Wert ist, der in der Systemkonfiguration bzw. Datenquelle als zuletzt gelesener Wert (**lastreadvalue**) gespeichert ist. Ist kein Wert gespeichert, wird der Vorgabewert **0** verwendet (im Beispiel würden alle Datensätze mit Werten des Integer-Kriteriums **>=0** gelesen werden). Als **lastreadvalue** wird entsprechend der Vorgabe (siehe auch Parameter **-save_value_minimum**) der größte zuletzt gelesene Wert gespeichert, hier z. B. **300**.

Beispiel 2 (lastreadvalue="40")

`-valueconstraint ">=" 270 "<="`

Es werden alle Datensätze ausgelesen, deren Integer-Wert zum Einschränken des auszulesenden Datenbereichs **>=40** und **<=270** ist. Nach dem Extraktionsvorgang wird der Wert für **lastreadvalue** z. B. auf **270** aktualisiert, wenn dies tatsächlich der größte zuletzt gelesene Wert ist.

Beispiel 3 (lastreaddate="19971231" lastreadtime="155959" lastreadvalue="270")

Durch erneuten Aufruf des Befehls

```
runsap2ppm -datasource datasource.xml -valueconstraint ">"
```

werden alle Datensätze gelesen, deren Integer-Wert zum Einschränken des auszulesenden Datenbereichs größer als **270** ist und deren Zeitstempel gleichalt oder jünger als der Startzeitpunkt der Datenextraktion (**31.12.1997 15:59:59 Uhr**) ist.

Als **lastreaddate**/**lastreadtime** wird der Startzeitpunkt der Datenextraktion in der Konfigurationsdatei eingetragen.

Durch das Weglassen der Parameter **-begindate**/**-begintime** lesen Sie stets die Datensätze aus, deren Zeitstempel jünger (größer) sind als die mittels **lastreaddate**/**lastreadtime** festgelegten zuletzt gelesenen Zeitstempelwerte. Als **lastreaddate**/**lastreadtime** werden aktuelles Datum bzw. aktuelle Uhrzeit eingetragen, wenn Sie die Parameter **-enddate** und **-endtime** nicht verwenden.

6.5 Extraktion mehrerer Datenquellen

Sie können mit Hilfe einer Datenquellenliste mehrere Datenquellen auf einmal extrahieren. Die Datenquellenliste kann in einer eigenen Konfigurationsdatei angegeben werden.

Jede in CTK neu angelegte SAP-Datenquelle wird automatisch am Ende der Datenquellenliste des aktuellen Mandanten hinzugefügt. Bei der Extraktion mit der Kommandozeilenoption **-datasourcelist <filename>** werden die Daten dieser Datenquellen nacheinander extrahiert, so als wäre die Datenextraktion mehrmals mit der Option **-datasource <filename>** aufgerufen worden. Die Reihenfolge, in der die Extraktion der Datenquellen erfolgt, ist in der Konfigurationsdatei angegeben. In der Liste vorhandene Datenquellen eines anderen Typs als MYSAP werden bei der Extraktion übersprungen.

Für die Fehlerbehandlung gilt Folgendes:

- Wird die Extraktion mit einer gültigen Datenquellenliste aufgerufen, die aber keine passenden Datenquellen enthält, so beendet sie sich ohne eine Fehlermeldung.
- Wird die Extraktion mit einer Datenquellenliste aufgerufen, die mehrere passende Datenquellen enthält und tritt bei der Extraktion einer dieser Datenquellen ein Fehler auf, der zum Abbruch dieses Vorgangs führt, so wird die Verarbeitung mit der nächsten Datenquelle fortgeführt, d. h. der Abbruch bei einer einzelnen Datenquelle führt nicht zu einem Komplettabbruch.
- Tritt bei mindestens einer Datenquelle ein Fehler auf, der bei der Einzel-Extraktion dieser Datenquelle zu einem Exit-Fehlerstatus geführt hätte, so liefert die Gesamtverarbeitung der Datenquellenliste den letzten dieser Exit-Fehlerstatus zurück.

6.6 SAP Secure Network Communication

SAP-2-PPM unterstützt SAP Secure Network Communication (SNC).) SAP SNC ist ein proprietäres Protokoll der SAP zur Absicherung von RFC-Verbindungen.

Mit Hilfe von CTK können Sie die erforderlichen Einstellungen für den Datenimport von Datenquellen mit SNC konfigurieren. Weitere Informationen dazu erhalten Sie in der CTK-Hilfe.

6.6.1 Voraussetzungen

Damit SAP-2-PPM SAP-Datenquellen mit SNC auslesen kann, müssen folgende Voraussetzungen erfüllt sein.

- Konfigurieren Sie den SAP-Anwendungsserver für SNC, inklusive der Einrichtung geeigneter SNC-Benutzer.
- Installieren Sie eine geeignete, SAP-zertifizierte und mit GSS API v2 kompatible Verschlüsselungssoftware. Wir empfehlen den Einsatz der SAP Common Cryptolib, mit der PPM getestet wurde.
- Konfigurieren Sie die Sicherheitsumgebung gemäß den Vorgaben des Herstellers der installierten Verschlüsselungssoftware. Standardmäßig werden dabei Sicherheitszertifikate mit dem SAP-Anwendungsserver ausgetauscht.
- Verwenden Sie SAP JCo 3.0.14 oder höher. Wir empfehlen, die aktuelle Version von JCo zu verwenden. Definieren Sie die Umgebungsvariablen **SNC_LIB** und **SECUDIR** in der Systemumgebung des SAP-2PPM-Extraktors. Die im JCo eingebettete RFC-Bibliothek benötigt diese Variablen zum Auffinden der SNC-Schicht und der benötigten Credentials.

Detaillierte Informationen zur SAP Common Cryptolib und SAP JCo entnehmen Sie bitte der entsprechenden SAP-Dokumentation.

6.6.2 XML-Konfiguration von SAP-Datenquellen

Die Definition der Systemkonfiguration (R3Config.dtd) einer SAP-Datenquelle mit SNC wird um die folgenden Attribute erweitert:

```
<!ATTLIST r3system
…
snc_mode (true|false) "false"
      snc_sso (true|false) "true"
      snc_qop (1|2|3|8|9) "9"
      snc_partnername CDATA #IMPLIED
snc_myname CDATA #IMPLIED
…
>
```

Attribut	Beschreibung
snc_mode	SNC aktivieren/deaktivieren Default: nicht aktiviert
snc_sso	SNC-Logon mit Benutzernamen und Kennwort (ohne SingleSign-On). Default: SSO verwenden
snc_qop	SNC-Dienstqualität 1: Nur Authentifizierung 2: Schutz der Integrität 3: Schutz der Vertraulichkeit 8: Vom Anwendungsserver vorgegebener Default 9: Höchste verfügbare Sicherheitseinstellung (Default)
snc_partnername	SNC-Name des Anwendungsservers (z. B. p:CN=R3, O=XYZ-INC, C=US)
snc_myname	SNC-Name des RFC-Clients (z. B. p:CN=Smith, O=XYZ-INC, C=EN)

Das bisher erforderliche Attribut **user** ist jetzt optional. Die Angabe von Benutzernamen und Kennwort ist bei SNC nicht erforderlich, es sei denn, dies wird explizit mit dem Attribut **snc_sso=false** so eingestellt.

Die SNC-Konfiguration ist bei Datenquellen des PPM-2-SAP Extraktors und bei SAP-Datenquellen für PPM Data Analytics identisch.

Hier ein Beispiel für eine vollständige Konfiguration unter Verwendung von SNC:

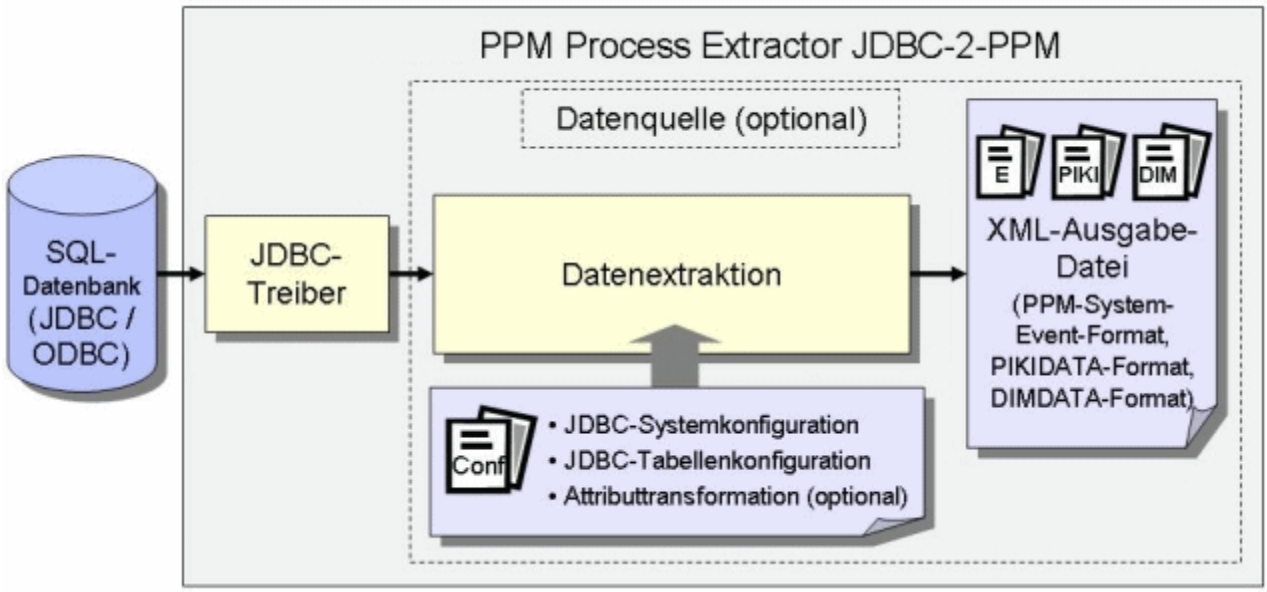
```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE r3systemconf SYSTEM "R3Config.dtd">
<r3systemconf>
  <r3system configname="snc_connection"
  appserver="xyz.com" systemnumber="00"
  client="777" user="" snc_mode="true" snc_sso="true" snc_qop="9"
  snc_partnername="p:CN=R3, O=XYZ-INC, C=US"
  snc_myname="p:CN=Smith, O=XYZ-INC, C=DE"
  language="" lastreaddate="19700101" lastreadtime="000000"
  encryptpw="" />
</r3systemconf>
```

7 PPM Process Extractor JDBC-2-PPM

Dieses Kapitel gibt einen Überblick über die Architektur, Funktionsweise und Konfiguration von PPM Process Extractor JDBC-2-PPM.

7.1 Architektur

Die folgende Abbildung veranschaulicht die Funktionalität von PPM Process Extractor JDBC-2-PPM:



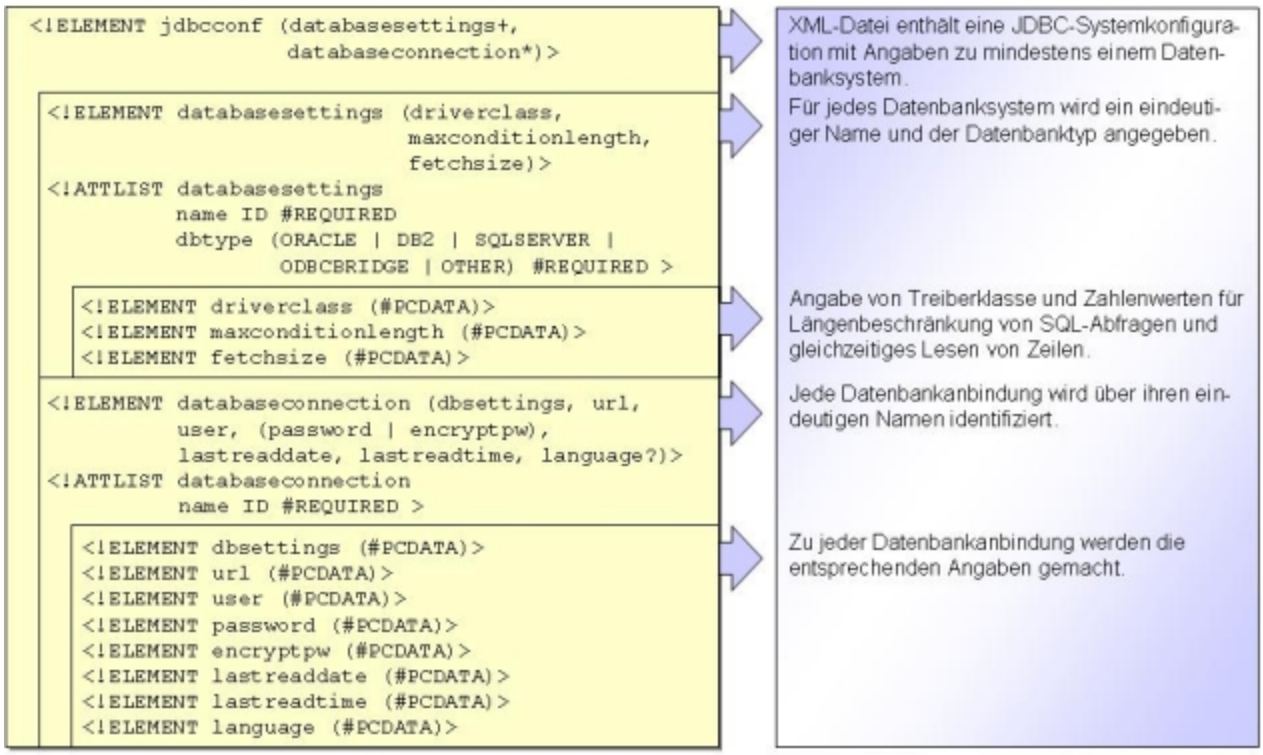
Mit Hilfe des JDBC-Treibers wird über die Zugangsdaten der JDBC-Systemkonfiguration eine Verbindung zur SQL-Quelldatenbank hergestellt. Anschließend werden Daten aus den Tabellen des Quelldatenbanksystems mit der JDBC-Tabellenkonfiguration und einer optionalen Attributtransformation über das Modul **Datenextraktion** (Seite 6) ausgelesen und in eine XML-Datei eines PPM-konformen Ausgabeformates geschrieben.

Achten Sie darauf, dass der angegebene Quellsystembenutzer eine ausreichende Zugriffsberechtigung besitzt, um die gewünschten Datenfelder auszulesen.

7.2 JDBC-Systemkonfiguration

ie Zugangsdaten zum SQL-Datenbanksystem werden in einer XML-Datei angegeben. Der Name dieser XML-Datei wird dem Kommandozeilenprogramm als Argument übergeben.

Das Format der XML-Datei ist durch folgende DTD vorgegeben:



XML-Element bzw. -Attribut	Beschreibung	Beispiel
jdbcconf	JDBC-Systemkonfiguration mit Angaben zu den zu verwendenden Datenbanksystemen und Datenbankanbindungen	-
databasesettings	Allgemeine Einstellungen des verwendeten Datenbanksystems	-
name	Eindeutiger Name der allgemeinen Einstellungen des Datenbanksystems	setting_oracle

XML-Element bzw. -Attribut	Beschreibung	Beispiel
dbtype	Verwendeter Datenbanktyp, mögliche Werte: **ORACLE** | **DB2** | **SQLSERVER** | **OTHER** | **ADABAS**	ORACLE
driverclass	Für das Datenbanksystem zu verwendende Treiberklasse	oracle.jdbc.driverOracleDriver
maxconditionlength	Der angegebene Zahlenwert beschränkt die Länge des WHERE-Teils einer SQL-Abfrage	1000
fetchsize	Anzahl der Datensätze, die gleichzeitig vom DB-Server auf den Java-Client übertragen werden. Standardwert ist bei den meisten Datenbanksystemen **10**. Empfohlener Wert: **1000**	1000
databaseconnection	Angaben zur Datenbankanbindung	-
name	Eindeutiger Name der Datenbankanbindung	oracle_connection1
dbsettings	Eindeutiger Name der allgemeinen Einstellungen des Datenbanksystems	setting_oracle
url	URL der Datenbankanbindung	s. Dateiauszug weiter unten
user	Name des Datenbankbenutzers	ppmoracle
password	Kennwort des Datenbankbenutzers	ppmoracle
encryptpw	Verschlüsseltes Kennwort als Zahlenkombination nach erfolgter Auswertung des <password>-Elements	62 -62 -56
lastreaddate	Datum, ab dem Daten ausgelesen werden. Entfällt bei Verwendung einer **Datenquelle** (Seite 159), Format: **yyyymmdd**	20050131
lastreadtime	Uhrzeit, ab der Daten	152917

XML-Element bzw. -Attribut	Beschreibung	Beispiel
	ausgelesen werden. Entfällt bei Verwendung einer **Datenquelle** (Seite 159), Format: **hhmmss**	
language	Sprache der ausgelesenen Textfelder	EN

Der Datenbanktyp kann einer der vier vorgegebenen Werte für die Datenbanksysteme Oracle, IBM DB2, MS SQL Server, MS Access oder ADABAS sein. Der Wert **OTHER** ist zu verwenden, wenn Sie Daten aus einer Datenbank eines anderen Typs als den vier Vorgenannten auslesen möchten.

Der Datenbanktyp **ADABAS** ist für die Verwendung mit dem Adabas Process Extractor vorgesehen. Eine detaillierte Beschreibung des Adabas Process Extractor finden Sie in der Dokumentation **Process Intelligence for Natural Applications How-To Guide**.

Mit Hilfe des Parameters **maxconditionlength** kann eine Zahl angegeben werden, welche die Länge des WHERE-Teils einer SQL-Abfrage auf den angegebenen Wert beschränkt. Dies ist notwendig, um eventuelle Längenbeschränkungen der verschiedenen Datenbanksysteme einhalten zu können. Ist die SQL-Abfrage länger als die angegebene Maximallänge, wird sie in mehrere, kleinere Abfragen aufgeteilt, die getrennt ausgeführt werden.

Mit Hilfe des Parameters **fetchsize** kann angegeben werden, wie viele Datenzeilen gleichzeitig pro Lesevorgang von der Datenbank an den JDBC-Extraktor übertragen werden sollen.

Das Kennwort des Datenbankbenutzers muss im XML-Element **password** unverschlüsselt in Klarschrift angegeben werden. Nach erfolgter Auswertung im nächsten Auslesevorgang wird das Kennwort schließlich im XML-Element **encryptpw** verschlüsselt in die JDBC-Systemkonfiguration zurückgeschrieben. Der **password**-Eintrag wird dann gelöscht. Es darf immer nur einer der beiden Einträge in der Systemkonfiguration vorhanden sein.

Wenn Sie ein bestehendes Kennwort ändern möchten, fügen Sie einen neuen **password**-Eintrag in die aktuelle Konfigurationsdatei ein und löschen den **encryptpw**-Eintrag. Im nächsten Auslesevorgang wird dann das neue Kennwort verschlüsselt.

Der Parameter **language** ist optional. Wenn Sie einen Wert dazu angeben, wird dieser mit dem Feldnamen aus der Tabellenkonfiguration (**langfieldname**) verglichen (siehe Kapitel **JDBC-Tabellenkonfiguration** (Seite 74)). Ist beispielsweise in einer Tabelle **langfieldname="LANG"** und in der JDBC-Systemkonfiguration **<language>EN</language>** angegeben, werden nur die Datensätze ausgelesen, bei denen **LANG** den Wert **EN** hat.

Folgender Dateiauszug zeigt eine beispielhafte JDBC-Systemkonfiguration:

```
<jdbcconf>
  <databasesettings name="setting_sqls" dbtype="SQLSERVER">
    <driverclass>com.microsoft.jdbc.sqlserver.SQLServerDriver
    </driverclass>
    <maxconditionlength>1000</maxconditionlength>
    <fetchsize>1000</fetchsize>
  </databasesettings>
  <databasesettings name="setting_oracle" dbtype="ORACLE">
    <driverclass>oracle.jdbc.driver.OracleDriver</driverclass>
    <maxconditionlength>1000</maxconditionlength>
    <fetchsize>1000</fetchsize>
  </databasesettings>
  <databaseconnection name="sqls_connection">
    <dbsettings>setting_sqls</dbsettings>
    <url>jdbc:microsoft:sqlserver:
      //PC3:1433;SelectMethod=Cursor;
                DatabaseName=ppmdb
    </url>
    <user>ppmuser</user>
    <password>ppmuser</password>
    <lastreaddate>20050228</lastreaddate>
    <lastreadtime>000000</lastreadtime>
    <language />
  </databaseconnection>
  <databaseconnection name="oracle_connection1">
    <dbsettings>setting_oracle</dbsettings>
    <url>jdbc:oracle:thin:@pcppm:1521:orappm_test
    </url>
    <user>ppmoracle</user>
    <password>ppmoracle</password>
    <lastreaddate>20051231</lastreaddate>
    <lastreadtime>235959</lastreadtime>
  </databaseconnection>
  <databaseconnection name="oracle_connection2">
    <dbsettings>setting_oracle</dbsettings>
    <url>jdbc:oracle:thin:@pcppm:1521:orappm_produktiv</url>
    <user>ppmoracle2</user>
    <encryptpw>48 -62 -76 -60 -108 -57 -92</encryptpw>
    <lastreaddate>20050228</lastreaddate>
    <lastreadtime>000000</lastreadtime>
  </databaseconnection>
</jdbcconf>
```

Damit sich PPM Process Extractor JDBC-2-PPM anhand der Konfigurationsdaten mit der Datenbank verbinden kann, müssen Sie die JDBC-Treiber (JAR- und/oder ZIP-Dateien) in folgendes Verzeichnis Ihrer Installation kopieren.

<PPM-Installationsverzeichnis>\ppm\server\bin\work\data_ppm\drivers

7.3 JDBC-Tabellenkonfiguration

Die JDBC-Tabellenkonfiguration (System-Event-Spezifikation) legt fest, welche Tabellenfelder aus dem SQL-Datenbanksystem ausgelesen und als Quellsystemattribute in die System-Events geschrieben werden. Sie können in der XML-Datei mehrere Tabellenkonfigurationen mit eindeutigen Namen speichern.

Die JDBC-Tabellenkonfiguration setzt sich aus folgenden Teilen zusammen:

GLOBALE TABELLEN

Aus globalen Tabellen werden Informationen gelesen, die für alle System-Events geschrieben werden.

FREMDSCHLÜSSELTABELLEN

Das XML-Element **docreftable** enthält den Namen der Fremdschlüsseltabelle. Es legt fest, wie der aus der System-Event-Tabelle auszulesende Datenbereich eingeschränkt wird. Über die im XML-Element **pkfield** angegebenen Primärschlüsselfelder ist die Fremdschlüsseltabelle mit der System-Event-Tabelle und anderen Fremdschlüsseltabellen verknüpft.

SYSTEM-EVENT-TABELLE

Das XML-Element **doctable** enthält den Namen der System-Event-Tabelle. Es bestimmt die zu einem Belegfluss auszulesenden Belege. Jeder aus der System-Event-Tabelle gelesene Datensatz erzeugt in der Ausgabedatei ein System-Event (XML-Element **event**).

DATENTABELLEN

Die Informationen der System-Event-Tabelle können durch Lesen weiterer Datenfelder aus beliebigen Datentabellen ergänzt werden (z. B. wird aus der System-Event-Tabelle die Materialnummer gelesen und aus einer Datentabelle der zu dieser Nummer gehörende Beschreibungstext).

Einige Datenbanksysteme erlauben Tabellennamen abweichend vom SQL-Standard. Das Auslesen solcher Tabellen mit PPM Process Extractor JDBC-2-PPM fürht zu einer Fehlermeldung. Sie können die Inhalte auslesen, in dem Sie für jede auszulesende Tabelle mit nicht SQL-konformen Namen eine View erstellen und diese zum Auslesen mit PPM Process Extractor JDBC-2-PPM verwenden.

Folgendes XML-Dateigerüst veranschaulicht die Konfiguration der zu lesenden Tabellen. Welche XML-Elemente bzw. -Attribute optional sind, können Sie der tabellarischen Erläuterung in Kap. **Konfiguration des Tabellenzugriffs** (Seite 77) entnehmen.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE r3systemconffields SYSTEM
           'jdbc_tableconfiguration.dtd'>
<jdbc_tableconfiguration>
  <configuration name="..." printname="..." classtouse="...">
    <globaltable name="..." tablename="..." classtouse="...">
      <fieldtoread name="..."/>
        <textref tablename="..." reffieldname="..."
```

```
            textfieldname="..." langfieldname="..."/>
  </fieldtoread>
  ...
</globaltable>
<docspec>
  <docreftable name="..." tablename="..."
                                  classtouse="...">
    <condition fieldname="..." logicaloperator="...">
      <value>...</value>
    </condition>
    <pkfield name="..." fktablename="..."
                                  fkfieldname="...">
      <fkpart readfrom="..."
                  startposition="..." length="..."/>
      <prefix>
        <value>...</value>
      </prefix>
      <postfix>
        <value>...</value>
      </postfix>
    </pkfield>
    ...
  </docreftable>
  ...
  <doctable name="..." tablename="..."
                                  classtouse="...">
    <condition fieldname="..." logicaloperator="...">
      <value>...</value>
    </condition>
    <pkfield name="..." fktablename="..."
                                  fkfieldname="...">
      <fkpart readfrom="..." startposition="..."
                                     length="..."/>
      <prefix>
        <value>...</value>
      </prefix>
      <postfix>
        <value>...</value>
      </postfix>
    </pkfield>
    ...
    <fieldtoread name="...">
      <textref tablename="..." reffieldname="..."
              textfieldname="..." langfieldname="..."/>
    </fieldtoread>
    ...
  </doctable>
</docspec>
<table name="..." tablename="..." classtouse="...">
  <condition fieldname="..." logicaloperator="...">
     <value>...</value>
  </condition>
  <pkfield name="..." fktablename="..."
                                  fkfieldname="...">
     <fkpart readfrom="..." startposition="..."
                                     length="..."/>
     <prefix>
       <value>...</value>
     </prefix>
     <postfix>
       <value>...</value>
```

```
        </postfix>
      </pkfield>
      ...
      <fieldtoread name="...">
        <textref tablename="..." reffieldname="..."
                 textfieldname="..." langfieldname="..."/>
      </fieldtoread>
      ...
    </table>
    ...
  </configuration>
  ...
</jdbc_tableconfiguration>
```

7.3.1 Konfiguration des Tabellenzugriffs

XML-Element	XML-Attribut	Beschreibung
jdbc_table-configuration		Liste der Tabellenkonfigurationen
configuration	name	Eindeutiger Identifizierer der Tabellenkonfigurationen
	printname (optional)	Schalter. **yes** bewirkt, dass der Konfigurationsname als Präfix der Feldnamen ausgegeben wird. Vorgabewert: **no**
	classtouse (optional)	Name der Java-Klasse, die für die Bearbeitung der Konfiguration verwendet wird Vorgabewert: Standardimplementierung
globaltable (optional)	name	Identifizierer der Tabelle, aus der globale System-Event-Attribute gelesen werden
	tablename (optional)	Name der Tabelle im Quellsystem Vorgabewert: Angabe im XML-Attribut **name**
	classtouse (optional)	Name der Java-Klasse, die zum Auslesen der Tabelle verwendet wird Vorgabewert: Standardimplementierung
docspec	-	Gruppierung der Spezifikationen aller Fremdschlüsseltabellen und der System-Event-Tabelle

XML-Element	XML-Attribut	Beschreibung
docreftable	name	Identifizierer einer Fremdschlüsseltabelle
	tablename (optional)	Name der Tabelle im Quellsystem Vorgabewert: Angabe im XML-Attribut **name**
	classtouse (optional)	Name der Java-Klasse, die zum Auslesen der Tabelle verwendet wird Vorgabewert: Standardimplementierung
doctable	name	Identifizierer der System-Event-Tabelle. Jede ausgelesene Zeile erzeugt ein System-Event in der XML-Datei.
	tablename (optional)	Name der Tabelle im Quellsystem Vorgabewert: Angabe im XML-Attribut **name**
	classtouse (optional)	Name der Java-Klasse, die zum Auslesen der Tabelle verwendet wird Vorgabewert: Standardimplementierung
table (optional)	name	Identifizierer der Datentabelle, aus der ergänzende Informationen gelesen werden
	tablename (optional)	Name der Tabelle im Quellsystem Vorgabewert: Angabe im XML-Attribut **name**
	classtouse (optional)	Name der Java-Klasse, die zum Auslesen der Tabelle verwendet wird Vorgabewert: Standardimplementierung

Jede Tabelle darf in der Konfiguration nur einmal referenziert werden. Um eine Tabelle mehrmals auslesen zu können, referenzieren Sie dieselbe Tabelle (XML-Attribut **tablename**) über verschiedene Identifizierer.

Beispiel

Aus der Tabelle **PRODH** sollen die Felder **VBAP-MATNR_1** und **VBAP-MATNR_2** ausgelesen werden. Der folgende Dateiauszug veranschaulicht die Konfiguration:

```
<table name="PROD_HIER_1" tablename="PRODH"
                              classtouse="ZTable">
   <pkfield name="HNUM" fktablename="VBAP"
                            fkfieldname="MATNR_1"/>
   <fieldtoread name="HVAL"/>
</table>

<table name="PROD_HIER_2" tablename="PRODH"
                              classtouse="ZTable">
   <pkfield name="HNUM" fktablename="VBAP"
                            fkfieldname="MATNR_2"/>
   <fieldtoread name="HVAL"/>
</table>
```

7.3.2 Globale Metadaten

Mit dem XML-Element **globaltable** können Metadaten aus den PPM Prozessextraktoren SAP-2-PPM und JDBC-2-PPM als globale System-Event-Attribute in die Ausgabedatei übernommen werden. Typische Daten für globale System-Event-Attribute sind z. B. die Namen der Konfigurationen, die zum Auslesen verwendet wurden. Die Daten selbst werden aus den in der Kommandozeile angegebenen Argumentwerten ermittelt. Für nicht in der Kommandozeile angegebene Argumente wird der Vorgabewert ermittelt.

Die Groß-/Kleinschreibung der Schlüsselwörter wird berücksichtigt.

Globale System-Event-Attribute sind für alle System-Events der aktuellen Ausgabedatei gültig.

Schlüsselwort	Wert aus Kommandozeilenargument
SYSTEM_CONFIG_FILE_NAME	-systemconfig <filename> ...
SYSTEM_CONFIG_NAME	-systemconfig ... <configname>
TABLE_CONFIG_FILE_NAME	-tableconfig <filename> ...
TABLE_CONFIG_NAME	-tableconfig ... <configname>
BEGIN_DATE,	-begindate <dd.MM.yyyy>
BEGIN_TIME	-begintime <hh:mm:ss>
END_DATE	-enddate <dd.MM.yyyy>
END_TIME	-endtime <hh:mm:ss>
CPD	-cpd <int>
OUT_FILE_NAME	-outfile <filename>
PIKI_DATA_MAPPING_FILE_NAME	-pikidatamapping <filename>...
PIKI_DATA_MAPPING_NAME	-pikidatamapping ... <pcname>

Schlüsselwort	Wert aus Kommandozeilenargument
DIM_DATA_MAPPING_FILE_NAME	-dimdatamapping <filename> ...
DIM_DATA_MAPPING_NAME	-dimdatamapping ... <dimname>

Kann für ein Schlüsselwort kein Wert ermittelt werden, wird das Schlüsselwort selbst in die Ausgabedatei geschrieben. Auf diese Weise können Sie Konstanten in die Ausgabedatei übernehmen.

Beispiel

Der folgende Dateiauszug zeigt die Konfiguration, die zum Erzeugen von globalen System-Event-Attributen verwendet wird:

```
<globaltable name="INFO">
  <fieldtoread name="SYSTEM_CONFIG_NAME"/>
</globaltable>
```

Der über die Kommandozeile mit -systemconfig **<configname>** angegebene Konfigurationsname, z. B. **sapppm** , wird als globales System-Event-Attribut **INFO-SYSTEM_CONFIG_NAME** in die Ausgabedatei geschrieben:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <attribute type="INFO-SYSTEM_CONFIG_NAME ">
    sapppm
  </attribute>
  ...
  <event>
    ...
  </event>
  ...
</eventlist>
```

Durch Angabe einer geeigneten Java-Klasse im XML-Attribut **classtouse** können auch beliebige andere Quellsystemtabellen ausgelesen werden.

7.3.3 System-Events sortieren

Sie können die System-Events der XML-Ausgabe gemäß dem Inhalt von Feldern, die Sie als Sortierkriterien festlegen, in alphanumerisch aufsteigender Reihenfolge sortieren. Die Angabe mehrerer Felder als Sortierkriterien ist möglich. Im Sortiervorgang findet dabei eine Priorisierung vom erstgenannten bis zum letztgenannten Feld statt, d. h. es wird zunächst gemäß den Inhalten des erstgenannten Feldes sortiert, dann nach den Inhalten des zweitgenannten Feldes usw.

Tipp

Nutzen Sie die Sortiermöglichkeit für einen effizienteren XML-Import der extrahierten Daten ins PPM-System. Dadurch, dass System-Events, die zu einer Prozessinstanz gehören, aufgrund einer Sortierung in den XML-Ausgabedateien unmittelbar aufeinander folgen, ist es möglich, diese bereits beim XML-Import über den Prozessschlüssel in eine EPK zu importieren.

In der Tabellenkonfiguration werden dazu im XML-Element **parameter** Felder als Sortierkriterien für die System-Event-Tabelle (**doctable**) und die referenzierten Fremdschlüsseltabellen (**docreftable**) angegeben.

Beispiel (Auszug aus der Tabellenkonfiguration)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
                'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
  <configuration name="BANF">
    <docspec>
      <doctable name="EBAN">
        <parameter name="ORDER_BY">
          <value>BNFPO</value>
          <value>BANFN</value>
        </parameter>
        <condition fieldname="ERDAT"
                   logicaloperator="creationtimestamp">
          <value>yyyyMMdd</value>
        </condition>
        <pkfield name="BANFN"/>
        <pkfield name="BNFPO"/>
        <pkfield name="ERDAT"/>
        <fieldtoread name="BSART"/>
        <fieldtoread name="KONNR"/>
        <fieldtoread name="KTPNR"/>
        <fieldtoread name="LIFNR"/>
        <fieldtoread name="MFRNR"/>
      </doctable>
    </docspec>
  </configuration>
</xmlextractor_tableconfiguration>
```

Gemäß den Angaben der Konfiguration **BANF** (XML-Element **parameter**) werden die System-Events nach der Positionsnummer der Bestellanforderungsnummer (Feld **BNFPO**) und in zweiter Priorität nach der Bestellanforderungsnummer (Feld **BANFN**) aufsteigend sortiert in die Ausgabedatei(en) geschrieben.

Die folgende Tabelle zeigt alle Konfigurationsmöglichkeiten:

XML-Element/-Attribut	Beschreibung
parameter	Angabe eines Sortierparameters für eine System-Event-Tabelle (**doctable**) bzw. eine Fremdschlüsseltabelle (**docreftable**)
name	Parametername. Es muss **ORDER_BY** angegeben werden.
value	Angabe eines Sortierfeldes. Das Feld muss als Primärschlüsselfeld (**pkfield**) angegeben sein.

Sie können den Sortierparameter in PPM Customizing Toolkit einstellen. Aktivieren Sie dazu die Registerkarte **System-Event** im Modul **Datenextraktion** der Modulgruppe **Prozess-Merge**. Wechseln Sie in den Bearbeitungsmodus und wählen Sie **System-Event bearbeiten** aus dem Kontextmenü auf der System-Event-Tabelle. Im Schritt **Parameter festlegen** können Sie die gewünschten Einstellungen vornehmen.

7.3.4 Aus gelesenen Attributwerten neue Attributwerte konkatenieren

Die Klasse **com.idsscheer.ppm.xmlextractortools.extractor.jdbc2ppm.ZTableConcatFKs_jdbc2ppm** kann verwendet werden, um während des Auslesevorgangs aus einem JDBC-System neue System-Event-Attribute zu erzeugen, indem die Werte bereits ausgelesener Attribute konkateniert werden. Die neu erzeugten System-Event-Attribute können in weiteren Datentabellenkonfigurationen (XML-Element **table**) referenziert werden.

Beispiel

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
                'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
  <configuration name="SD">
    <docspec>
      <doctable name="RBKP">
        <condition logicaloperator="eq" fieldname="GJAHR">
          <value>2005</value>
        </condition>
        <pkfield name="BELNR"/>
        <pkfield name="GJAHR"/>
      </doctable>
    </docspec>
    <table name="RBKP_NEW" classtouse="com.idsscheer.ppm.
                        xmlextractortools.extractor.
                    jdbc2ppm.ZTableConcatFKs_jdbc2ppm">
      <pkfield name="BELNR" fktablename="RBKP"
                              fkfieldname="BELNR"/>
      <pkfield name="GJAHR" fktablename="RBKP"
                              fkfieldname="GJAHR"/>
      <fieldtoread name="AWKEY_GENERATED"/>
    </table>
  </configuration>
</xmlextractor_tableconfiguration>
```

Wenn Sie die Klasse **ZTableConcatFKs_jdbc2ppm** verwenden, müssen Sie genau ein **fieldtoread**-Element angeben. Das Element enthält den Namen des zu erzeugenden System-Event-Attributs. Im Beispiel wird ein neues System-Event-Attribut mit dem Namen **RBKP_NEW-AWKEY_GENERATED** erzeugt. Der Wert des Attributs setzt sich aus den Werten der angegebenen **pkfield**-Elemente zusammen.
Bei den **pkfield**-Elementen können die üblichen Operationen **fkpart**, **prefix** und **postfix** verwendet werden.

Ein mit obiger Konfiguration erzeugtes System-Event könnte beispielsweise wie folgt aussehen:

```
Ermittle die neuen bzw. geänderten MM-Belege ...
<event>
  <attribute type="RBKP-BELNR">5105601204</attribute>
  <attribute type="RBKP-GJAHR">2005</attribute>
  <attribute type="RBKP_NEW-AWKEY_GENERATED">51056012042005
  </attribute>
</event>
Ermittle die neuen bzw. geänderten MM-Belege ...
```

Das neu erzeugte Attribut **RBKP_NEWER-AWKEY_GENERATED** kann anschließend wie jedes andere JDBC-Tabellenfeld in weiteren **table**-Konfigurationen referenziert werden.

Beispiel

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xmlextractor_tableconfiguration SYSTEM
                'xmlextractor_tableconfiguration.dtd'>
<xmlextractor_tableconfiguration>
  <configuration name="SD">
    <docspec>
      <doctable name="RBKP">
        <condition logicaloperator="eq" fieldname="GJAHR">
          <value>2005</value>
        </condition>
        <pkfield name="BELNR"/>
        <pkfield name="GJAHR"/>
      </doctable>
    </docspec>
    <table name="RBKP_NEW" classtouse="com.idsscheer.
                ppm.xmlextractortools.extractor.
                   jdbc2ppm.ZTableConcatFKs_jdbc2ppm">
      <pkfield name="BELNR" fktablename="RBKP"
                              fkfieldname="BELNR"/>
      <pkfield name="GJAHR" fktablename="RBKP"
                              fkfieldname="GJAHR"/>
      <fieldtoread name="AWKEY_GENERATED"/>
    </table>
    <table name="BKPF">
      <pkfield name="AWKEY" fktablename="RBKP_NEW"
                     fkfieldname="AWKEY_GENERATED"/>
      <fieldtoread name="BELNR"/>
    </table>
  </configuration>
</xmlextractor_tableconfiguration>
```

Ein mit dieser Konfiguration erzeugtes System-Event könnte bspw. wie folgt aussehen:

```
Ermittle die neuen bzw. geänderten MM-Belege ...
<event>
  <attribute type="RBKP-BELNR">5105601204</attribute>
  <attribute type="RBKP-GJAHR">2005</attribute>
  <attribute type="RBKP_NEW-AWKEY_GENERATED"
                     >51056012042005</attribute>
  <attribute type="BKPF-BELNR">5100004691</attribute>
</event>
Ermittle die neuen bzw. geänderten MM-Belege ...
```

7.3.5 Auslesen mit Bedingungen

Das Auslesen der Tabellen kann durch Bedingungen eingeschränkt werden. Sie können mit Bedingungen z. B. steuern, dass beim Auslesen von Tabellenfeldern nur Belegflüsse eines Belegtyps in einem bestimmten Zeitraum gelesen werden.

Die Bedingungen werden in der Tabellenkonfiguration direkt im XML-Element **booleancondition** bzw. **condition** angegeben und gelten nur für die zugehörige Tabelle.

Eine Bedingung enthält den Namen des Tabellenfeldes, den Vergleichsoperator und einen konkreten Wert. Bedingungen lassen sich beliebig komplex verknüpfen (XML-Element **booleancondition**).

XML-Element	XML-Attribut	Beschreibung
booleancondition (optional)	logicaloperator	Logische Operatoren: **AND**, **OR**, **NOT** Vorgabewert: **AND**
condition (optional)	logicaloperator	Vergleichsoperatoren: **eq**, **neq**, **in**, **notin**, **num_gt**, **num_geq**, **num_lt**, **num_leq**, **is_null**, **is_not_null**, **timestamp_eq**, **timestamp_geq**, **timestamp_gt**, **timestamp_leq**, **timestamp_lt**, **time_eq**, **time_geq**, **time_gt**, **time_leq**, **time_lt**, **date_eq**, **date_geq**, **date_gt**, **date_leq**, **date_lt**, **num_eq**, **num_neq**, **num_in**, **num_notin** Operatoren zum Einschränken des auszulesenden Datenbereichs: **char_creationtimestamp**, **date_creationtimestamp**, **valueconstraint** Vorgabewert: **eq**
	fieldname	Tabellenfeldname

Die XML-Elemente **docbooleancondition** und **doccondition** ermöglichen das bedingte Auslesen einer Datentabelle in Abhängigkeit von bereits gelesenen Tabellenfeldern. Sie werden ähnlich wie **booleancondition** und **condition** konfiguriert. Der Tabellenname und der Name der Spalte, die die bereits gelesenen Datenfelder enthält, werden in den XML-Attributen **tablename** und **fieldname** angegeben.

XML-Element	XML-Attribut	Beschreibung
docbooleancondition (optional)	logicaloperator	Logische Operatoren: **AND**, **OR**, **NOT** Vorgabewert: **AND**
doccondition (optional)	logicaloperator	Vergleichsoperatoren: **eq**, **neq**, **in**, **notin**, **exists**, **notexists**, **is_null**, **is_not_null**, Vorgabewert: **eq**
	tablename	Name der Tabelle, die die bereits gelesenen Tabellenfelder enthält
	fieldname	Spaltenname

7.3.5.1 Bedingungsoperatoren

Gemeinsame Operatoren

Folgende Vergleichsoperatoren werden von den beiden XML-Elementen **condition** und **doccondition** unterstützt:

Operator	Beschreibung
eq	Feldinhalt ist gleich dem angegebenen Wert.
neq	Feldinhalt ist ungleich dem angegebenen Wert.
in	Feldinhalt ist gleich einem angegebenen Wert aus einer Wertemenge.
notin	Feldinhalt ist ungleich einem angegebenen Wert aus einer Wertemenge.
is_null	Prüft, ob der Feldinhalt NULL ist
is_not_null	Prüft, ob der Feldinhalt ungleich NULL ist

condition-Operatoren

Operator	Beschreibung
num_gt	Feldinhalt ist größer als angegebener Wert.
num_geq	Feldinhalt ist größer als oder gleich dem angegebenen Wert.
num_lt	Feldinhalt ist kleiner als angegebener Wert.
num_leq	Feldinhalt ist kleiner als oder gleich dem angegebenen Wert.
num_eq	Feldinhalt ist gleich dem angegebenen Wert.

Operator	Beschreibung
num_neq	Feldinhalt ist ungleich dem angegebenen Wert.
num_in	Feldinhalt ist gleich einem angegebenen Wert aus einer Wertemenge.
num_notin	Feldinhalt ist ungleich einem angegebenen Wert aus einer Wertemenge.
timestamp_eq	Prüft, ob der Zeitstempel des Feldes mit dem angegebenen Vergleichswert (**value**) übereinstimmt. Der Vergleichswert muss im Format **dd.MM.yyyy HH:mm:ss** angegeben werden, dies gilt auch für alle anderen timestamp_*-Operatoren.
timestamp_geq	Prüft, ob der Zeitstempel des Feldes größer/gleich Vergleichswert ist
timestamp_gt	Prüft, ob der Zeitstempel des Feldes größer Vergleichswert ist
timestamp_leq	Prüft, ob der Zeitstempel des Feldes kleiner/gleich Vergleichswert ist
timestamp_lt	Prüft, ob der Zeitstempel des Feldes kleiner Vergleichswert ist
time_eq	Prüft Uhrzeit des Feldes auf Gleichheit. Der Vergleichswert muss im Format **HH:mm:ss** angegeben werden, dies gilt auch für alle anderen time_*-Operatoren.
time_geq	Prüft Uhrzeit des Feldes auf größer/gleich
time_gt	Prüft Uhrzeit des Feldes auf größer
time_leq	Prüft Uhrzeit des Feldes auf kleiner/gleich
time_lt	Prüft Uhrzeit des Feldes auf kleiner
date_eq	Prüft Datum des Feldes auf Gleichheit. Der Vergleichswert muss im Format **dd.MM.yyyy** angegeben werden, dies gilt auch für alle anderen date_*-Operatoren.
date_geq	Prüft Datum des Feldes auf größer/gleich
date_gt	Prüft Datum des Feldes auf größer
date_leq	Prüft Datum des Feldes auf kleiner/gleich
date_lt	Prüft Datum des Feldes auf kleiner

Operator	Beschreibung
like (nur mit Zeichenketten-Datenbankfeldern, bspw. CHAR, VARCHAR)	Vergleich von Feldwerten mit einer variablen Zeichenkette Folgende Platzhalter sind erlaubt: * Kein bzw. beliebig viele Zeichen ? Genau ein beliebiges Zeichen \ Maskierungszeichen für die Suche nach Platzhaltern bzw. Maskierungszeichen in der Form: \\ bzw. * bzw. \? Beispiel: `<condition fieldname="OBJECTID" logicaloperator="like"> <value>*10?0\\20?0*</value> </condition>` Gesucht werden Werte wie z. B. 5551050\20106667 oder 1080\204044, aber bspw. nicht 34510550\2030*
char_ creationtimestamp	Aus Quelldatenbankfeldern vom Datentyp CHAR/VARCHAR, die Zeitangaben enthalten, werden Zeitstempel (Datum und Uhrzeit) gelesen. Deren Werte bilden die Grundlage für die Einschränkung des auszulesenden Datenbereichs mit den Kommandozeilenparametern **-begindate** (**-begintime**) bzw. **-enddate** (**-endtime**) [s. Kap. **Quelldatenbankspezifische Argumente** (Seite 116)]. Mehrere Felder werden durch die Zeichenkombination **#-#** voneinander getrennt. Für jedes Feld wird im XML-Element **value** das Zeitstempelformat angegeben, in welchem die Werte im entsprechenden Quelldatenbankfeld gespeichert sind. Beispiel: `<condition fieldname="VC_DATE#-#VC_TIME" logicaloperator="char_creationtimestamp"> <value>yyyyMMdd</value> <value>HHmmss</value> </condition>` Beachten Sie bei Verwendung des Operators **char_creationtimestamp** mögliche Auswirkungen von speziellen Zeitformaten der Quelldatenbankfelder auf Sortierung und Extraktion der Daten. Die Datenbankfelder in einem Zeitformat werden in alphabetischer Reihenfolge ausgelesen.

Operator	Beschreibung
	Sind die Werte des Datumsfeldes bspw. im Format **ddMMyyyy** in der Datenbank gespeichert, würde beim Auslesen des Zeitraumes **15.01.2000 - 31.12.2000** das Datum **23021999** ausgelesen werden, weil es alphabetisch gesehen zwischen Start- und Enddatum liegt, das Datum **09122000** dagegen nicht, weil es als kleiner als das Startdatum interpretiert würde.
date_ creationtimestamp	Aus Quelldatenbankfeldern mit Zeitangaben, die von einem datenbanksystemabhängigen Zeit-Datentyp sind, werden Zeitstempel (Datum und Uhrzeit) gelesen. Deren Werte bilden die Grundlage für die Einschränkung des auszulesenden Datenbereichs mit den Kommandozeilenparametern **-begindate** (**-begintime**) bzw. **-enddate** (**-endtime**) [s. Kap. **Quelldatenbankspezifische Argumente** (Seite 116)]. Mehrere Felder werden durch die Zeichenkombination **#-#** voneinander getrennt. Für jedes auszulesende Feld werden in den XML-Elementen **value** die Datentypen entsprechend den Quelldatenbankfeldern in einem der drei Zeit-Datentypen angegeben. Es sind nur folgende Datentypen in den angegebenen Kombinationen zulässig: **DATE**/**TIME** oder **TIME**/**DATE** oder **TIMESTAMP** oder **DATE** Beispiel: `<condition fieldname="CHG_DATE#-#CHG_TIME" logicaloperator="date_creationtimestamp"> <value>DATE</value> <value>TIME</value> </condition>`
valueconstraint	Angabe eines Quelldatenbanksystemfeldes mit Integer-Werten, die für die Einschränkung des auszulesenden Datenbereichs mit dem Kommandozeilenparameter **-valueconstraint** verwendet werden [s. Kap. **Quelldatenbankspezifische Argumente** (Seite 116)]. Das auszulesende Feld muss von einem Integer-Datentyp sein.

Operator	Beschreibung
	Beispiel: `<condition fieldname="INTEGERFIELD" logicaloperator="valueconstraint"/>`

Bei Verwendung der **timestamp**_*-, **time**_*- und **date**_*-Operatoren aus obiger Tabelle müssen Sie wissen, welcher Datentyp in welchem Format dem aus dem entsprechenden Datenbanksystem (Oracle, IBM DB2, MS SQL Server) ausgelesenen Wert zugrunde liegt.

Die folgenden Tabellen liefern eine Übersicht über die unterschiedlichen DB-Datentypen und DB-Formate sowie erzeugte Beispielformate in der XML-Ausgabedatei:

Oracle / Datentyp TIMESTAMP

In die Datenbank geschriebenes Wert-format (Beispiel)	Ergebnisformat in der XML-Ausgabedatei (Beispiel)	Anmerkung
07.05.2005 04:02:36	07.05.2005 04:02:36	-
07.05.2005	07.05.2005 12:25:11	Uhrzeit = Erzeugungsuhrzeit des Datenbank-Feldwerts
04:02:36	21.12.2005 04:02:36	Datum = Erzeugungsdatum des Datenbank-Feldwerts

Oracle / Datentyp DATE

In die Datenbank geschriebenes Wert-format (Beispiel)	Ergebnisformat in der XML-Ausgabedatei (Beispiel)	Anmerkung
07.05.2005 04:02:36	07.05.2005 04:02:36	-
07.05.2005	07.05.2005 12:25:11	Uhrzeit = Erzeugungsuhrzeit des Datenbank-Feldwerts
04:02:36	21.12.2005 04:02:36	Datum = Erzeugungsdatum des Datenbank-Feldwerts

Der Oracle-Datentyp **DATE** speichert ausschließlich Zeitstempelwerte im Format **dd.MM.yyyy HH:mm:ss**. Um die gewünschten Daten aus einer Oracle-Datenbank auslesen zu können, müssen die Auslesebedingungen bei Verwendung von **date**_*-Operatoren in geeigneter Weise konfiguriert werden.

Beispiel

Sie möchten alle Datensätze mit dem Datum **16.09.2004** auslesen, die Uhrzeit sei dabei egal. Wenn Sie in der Tabellenkonfiguration die folgende Bedingung angeben:

```
<condition fieldname="TACT_TDATE" logicaloperator="date_eq">
  <value>16.09.2004</value>
</condition>
```

werden nur die Datensätze mit dem Zeitstempel **16.09.2004 00:00:00** ausgelesen. Um sicherzustellen, dass alle Datensätze des angegebenen Datums ausgelesen werden, müssen Sie die Bedingung umformulieren:

```
<booleancondition logicaloperator="AND">
  <condition fieldname="TACT_TDATE"
                     logicaloperator="date_geq">
  <value>16.09.2004</value>
  </condition>
  <condition fieldname="TACT_TDATE"
                      logicaloperator="date_lt">
  <value>17.09.2004</value>
  </condition>
</booleancondition>
```

Mit der angegebenen Bedingung werden alle Datensätze mit dem Datum **16.09.2004** und beliebiger Uhrzeit ausgelesen.

IBM DB2 / Datentyp TIMESTAMP

In die Datenbank geschriebenes Wert-format (Beispiel)	Ergebnisformat in der XML-Ausgabedatei (Beispiel)	Anmerkung
07.05.2005 04:02:36	07.05.2005 04:02:36	-

IBM DB2 / Datentyp DATE

In die Datenbank geschriebenes Wert-format (Beispiel)	Ergebnisformat in der XML-Ausgabedatei (Beispiel)	Anmerkung
07.05.2005	07.05.2005	-

IBM DB2 / Datentyp TIME

In die Datenbank geschriebenes Wert-format (Beispiel)	Ergebnisformat in der XML-Ausgabedatei (Beispiel)	Anmerkung
04:02:36	04:02:36	-

MS SQL Server / Datentyp DATETIME

In die Datenbank geschriebenes Wert-format (Beispiel)	Ergebnisformat in der XML-Ausgabedatei (Beispiel)	Anmerkung
07.05.2005 04:02:36	07.05.2005 04:02:36	-
07.05.2005	07.05.2005 00:00:00	Uhrzeit ist immer 00:00:00
04:02:36	01.01.1900 04:02:36	Datum ist immer 01.01.1900

MS SQL Server / Datentyp SMALLDATETIME

In die Datenbank geschriebenes Wert-format (Beispiel)	Ergebnisformat in der XML-Ausgabedatei (Beispiel)	Anmerkung
07.05.2005 04:02:36	07.05.2005 04:02:00	Uhrzeit nur minutengenau
07.05.2005	07.05.2005 00:00:00	Uhrzeit ist immer 00:00:00
04:02:36	01.01.1900 04:02:00	Datum ist immer 01.01.1900 und die Uhrzeit nur minutengenau

doccondition-Operatoren

Operator	Beschreibung
exists	Feld existiert.
notexists	Feld existiert nicht.

7.3.5.2 Komplexe Bedingungen

Der folgende Dateiauszug veranschaulicht verschachtelte Bedingungen am Beispiel der Fremdschlüsseltabelle:

```
...
  <docreftable tablename="VBAK" classtouse="ZDocRefTable">
    <booleancondition logicaloperator="AND">
      <condition fieldname="ERDAT#-#ERZET"
              logicaloperator="char_creationtimestamp">
        <value>yyyyMMdd</value>
        <value>HHmmss</value>
      </condition>
      <condition fieldname="VBTYP" logicaloperator="in">
        <value>C</value>
        <value>K</value>
      </condition>
      <booleancondition logicaloperator="OR">
        <booleancondition logicaloperator="AND">
          <condition fieldname="VKORG" logicaloperator="eq">
            <value>1000</value>
          </condition>
          <condition fieldname="VKBUR" logicaloperator="eq">
            <value>0041</value>
          </condition>
        </booleancondition >
        <booleancondition logicaloperator="AND">
          <condition fieldname="VKORG" logicaloperator="eq">
            <value>2000</value>
          </condition>
          <condition fieldname="VKBUR" logicaloperator="neq">
            <value>0060</value>
          </condition>
        </booleancondition >
      </booleancondition >
    </booleancondition >
    ...
  </docreftable>
...
```

Aussagenlogisch entspricht der gezeigte Dateiauszug der folgenden Bedingung:

ERDAT und ERZET enthalten den Erstellzeitpunkt

und

VBTYP ist C oder V

und

((VKORG ist gleich 1000 **und** VKBUR ist gleich 0041) **oder** (VKORG ist gleich 2000 **und** VKBUR ist ungleich 0060))

7.3.6 Datenbankfelder auslesen

Die Datenbankfelder der System-Event-Tabelle und der verknüpften Datentabellen, aus denen Werte ausgelesen werden sollen, werden in den XML-Elementen **fieldtoread** angegeben. Für jedes XML-Element **fieldtoread** wird eine Zeile der Form

```
<attribute type="Type">Wert</attribute>
```

in die Ausgabedatei geschrieben.

Type setzt sich aus dem Tabellennamen, dem Datenfeldnamen und einem optionalen Textfeldnamen zusammen.
Wert ist der aus dem entsprechenden Datenfeld gelesene Wert. Alle Werte werden in Textform geschrieben.

Optional kann anstelle des direkten Datenfeldwertes der Wert der referenzierten Tabelle (XML-Element **textref**) ausgelesen werden. Die optionale Angabe **langfieldname** liest den Text des Datenfeldes sprachabhängig aus.

Aus den Datentabellen werden über Primärschlüsselbeziehungen ergänzende Informationen ausgelesen.

XML-Element	XML-Attribut	Beschreibung
fieldtoread	name	Name der Tabellenspalte, die das auszulesende Datenfeld enthält
textref	tablename	Tabellenname der referenzierten Datentabelle
	reffieldname	Name des Fremdschlüssels
	textfieldname	Name der Tabellenspalte, die das auszulesende Datenfeld enthält
	langfieldname (optional)	Name der Tabellenspalte, die das sprachabhängig auszulesende Datenfeld enthält
fkpart	readfrom (optional)	Richtung der Teilzeichenkette zur Bildung des Fremdschlüssels Mögliche Werte: **left**, **right** Vorgabewert: **left**
	startposition	Position, ab der die Teilzeichenkette gebildet wird
	length (optional)	Länge der Teilzeichenkette Vorgabewert: Anfang bzw. Ende der Zeichenkette (XML-Attribut **readfrom**)

Bevor tatsächlich Daten ausgelesen werden, wird überprüft, ob die konfigurierten Tabellen und Datenfelder existieren und ob der angegebene Systembenutzer eine ausreichende Zugriffsberechtigung hat.

Beispiel

Die gelesenen Feldwerte werden als Attribute eines System-Event's in die Ausgabedatei geschrieben.

Nicht lokalisierte Tabellenfelder:

```
...
  <attribute type="VBAP-WERKS">SB</attribute>
...
```

Lokalisierte Tabellenfelder:

```
...
  <attribute type="VBAP-WERKS-NAME">Saarbrücken</attribute>
...
```

7.3.6.1 Behandlung von Null-Werten

Bei der Behandlung von Null-Werten im Auslesevorgang ist zu unterscheiden, ob der Wert NULL oder eine leere Zeichenkette ausgelesen wird. Nicht jedes verwendete Datenbanksystem unterstützt die Unterscheidung zwischen NULL und leerer Zeichenkette:

- Das Datenbanksystem Oracle unterscheidet nicht zwischen keinem Wert (NULL) und einer leeren Zeichenkette. In beiden Fällen wird beim Auslesen von entsprechenden Datenbankfeldern kein System-Event-Attribut erzeugt.
- Die Datenbanksysteme IBM DB2 und SQL-Server unterscheiden zwischen NULL und leerer Zeichenkette. Liefert ein Datenbankfeld beim Auslesen den Wert NULL zurück, wird kein System-Event-Attribut erzeugt. Dagegen wird beim Auslesen eines Datenbankfeldes, das eine leere Zeichenkette enthält, das entsprechende System-Event-Attribut in der XML-Ausgabedatei ohne Wert erzeugt:

```
...
  <attribute type="AT_EXAMPLE"></attribute>
...
```

Wenn Sie anstelle von ausgelesenen Null-Werten Standardwerte, wie z. B. **nicht gepflegt** nach dem Import in das PPM-System verwenden möchten, können Sie dafür Standardwerte (XML-Element **defaultvalue**) in der Berechnung des entsprechenden Attributs definieren (siehe Handbuch **PPM Customizing**).

7.3.6.2 Felder mehrwertig auslesen

Mittels einer geeigneten Konfiguration des jeweiligen **table**-Elements der Tabellenkonfiguration können alle unterschiedlichen Werte eines zu extrahierenden Feldes (**fieldtoread**) in ein System-Event geschrieben werden. Dazu ist eine eigene Klasse zu verwenden.

Beispiel

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE jdbc_tableconfiguration SYSTEM
```

```
                      'jdbc_tableconfiguration.dtd'>
<jdbc_tableconfiguration>
  <configuration name="MULTIPLE_VALUES">
    <docspec>
      <docreftable name="...">
        ...
      </docreftable>
      <doctable name="...">
        ...
      </doctable>
    </docspec>
    <table name="VBEP" classtouse="com.idsscheer.ppm.
              xmlextractortools.extractor.jdbc2ppm.
                       ZTableMultipleValues_jdbc2ppm">
      <pkfield name="VBELN" fktablename="VBAP"
                              fkfieldname="VBELN"/>
      <pkfield name="POSNR" fktablename="VBAP"
                              fkfieldname="POSNR"/>
      <fieldtoread name="WMENG"/>
      <fieldtoread name="EDATU"/>
    </table>
    <table name="VBAK" classtouse="com.idsscheer.ppm.
              xmlextractortools.extractor.jdbc2ppm.
                       ZTableMultipleValues_jdbc2ppm">
      <pkfield name="VBELN" fktablename="VBEP"
                              fkfieldname="VBELE"/>
      <fieldtoread name="VBTYP"/>
    </table>
  </configuration>
</jdbc_tableconfiguration>
```

In der Beispielkonfiguration ist durch Verwendung der Klasse **com.idsscheer.ppm.xmlextractortools.extractor.jdbc2ppm.ZTableMultipleValues_jdbc2ppm** für alle Datentabellen (**table**-Elemente) festgelegt, dass alle auszulesenden Felder (**fieldtoread**-Elemente) mehrwertig ausgelesen werden, wenn in der Quellsystemdatenbank mehrere Werte zu einem Feld existieren. Das mehrwertig ausgelesene Feld erscheint in der XML-Ausgabedatei als gleichnamige Attribute innerhalb eines System-Event's:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="VBAP-POSNR">000010</attribute>
    <attribute type="VBAP-VBELN">0000004969</attribute>
    <attribute type="VBEP-EDATU">19970103</attribute>
    <attribute type="VBEP-EDATU">19970107</attribute>
    <attribute type="VBEP-WMENG">138.000</attribute>
    <attribute type="VBEP-WMENG">317.000</attribute>
    <attribute type="VBEP-WMENG">496.000</attribute>
  </event>
</eventlist>
```

7.3.7 Konfiguration des Tabellenzugriffs

XML-Element	XML-Attribut	Beschreibung
xmlextractor_table-configuration		Liste der R/3-Tabellenkonfigurationen
configuration	name	Eindeutiger Identifizierer der Tabellenkonfigurationen
	printname (optional)	Schalter. **yes** bewirkt, dass der Konfigurationsname als Präfix der Feldnamen ausgegeben wird. Vorgabewert: **no**
	classtouse (optional)	Name der Java-Klasse, die für die Bearbeitung der Konfiguration verwendet wird Vorgabewert: Standardimplementierung
globaltable (optional)	name	Identifizierer der Tabelle, aus der globale System-Event-Attribute gelesen werden
	tablename (optional)	Name der Tabelle im Quellsystem Vorgabewert: Angabe im XML-Attribut **name**
	classtouse (optional)	Name der Java-Klasse, die zum Auslesen der Tabelle verwendet wird Vorgabewert: Standardimplementierung
docspec	-	Gruppierung der Spezifikationen aller Belegkopftabellen und der Belegtabelle
docreftable	name	Identifizierer einer Belegkopftabelle
	tablename (optional)	Name der Tabelle im Quellsystem Vorgabewert: Angabe im XML-Attribut **name**

XML-Element	XML-Attribut	Beschreibung
	classtouse (optional)	Name der Java-Klasse, die zum Auslesen der Tabelle verwendet wird Vorgabewert: Standardimplementierung
doctable	name	Identifizierer der Belegtabelle. Jede ausgelesene Zeile erzeugt ein System-Event in der XML-Ausgabedatei.
	tablename (optional)	Name der Tabelle im Quellsystem Vorgabewert: Angabe im XML-Attribut **name**
	classtouse (optional)	Name der Java-Klasse, die zum Auslesen der Tabelle verwendet wird Vorgabewert: Standardimplementierung
table (optional)	name	Identifizierer der Datentabelle, aus der ergänzende Informationen gelesen werden
	tablename (optional)	Name der Tabelle im Quellsystem Vorgabewert: Angabe im XML-Attribut **name**
	classtouse (optional)	Name der Java-Klasse, die zum Auslesen der Tabelle verwendet wird Vorgabewert: Standardimplementierung

Jede SAP-Tabelle darf in der Konfiguration nur einmal referenziert werden. Um eine Tabelle mehrmals auslesen zu können, referenzieren Sie dieselbe Tabelle (XML-Attribut **tablename**) über verschiedene Identifizierer.

Beispiel

Aus der Tabelle **PRODH** sollen die Felder **VBAP-MATNR_1** und **VBAP-MATNR_2** ausgelesen werden. Der folgende Dateiauszug veranschaulicht die Konfiguration:

```
<table name="PROD_HIER_1" tablename="PRODH" >
  <pkfield name="HNUM" fktablename="VBAP"
                       fkfieldname="MATNR_1"/>
  <fieldtoread name="HVAL"/>
</table>
```

```
<table name="PROD_HIER_2" tablename="PRODH" >
  <pkfield name="HNUM" fktablename="VBAP"
                       fkfieldname="MATNR_2"/>
  <fieldtoread name="HVAL"/>
</table>
```

7.3.8 Beispiel: XML-Konfiguration und -Ausgabedatei

Dieses Kapitel veranschaulicht anhand eines Praxisbeispiels die Tabellenkonfiguration eines SQL-Datenbanksystems und zeigt eine mögliche XML-Ausgabedatei.

7.3.8.1 Tabellenkonfiguration

Folgende Datei zeigt die Konfiguration **SD_C** eines SQL-Datenbanksystems:

```
...
<configuration name="SD_C">
  <docspec>
    <docreftable name="VBAK">
      <booleancondition>
        <condition fieldname="ERDATCHAR#-#ERZETCHAR"
                logicaloperator="char_creationtimestamp">
          <value>yyyyMMdd</value>
          <value>HHmmss</value>
        </condition>
        <condition fieldname="ERDAT#-#ERZET"
                logicaloperator="date_creationtimestamp">
          <value>DATE</value>
          <value>TIME</value>
        </condition>
        <condition fieldname="VBTYP" logicaloperator="eq">
          <value>C</value>
        </condition>
      </booleancondition>
      <pkfield name="VBELN"/>
    </docreftable>
    <doctable name="Auftraege" tablename="VBAP">
      <pkfield name="VBELN" fktablename="VBAK"
                        fkfieldname="VBELN"/>
      <pkfield name="POSNR"/>
      <fieldtoread name="VBELN"/>
      <fieldtoread name="POSNR"/>
      <fieldtoread name="ERDAT"/>
      <fieldtoread name="ERZET"/>
      <fieldtoread name="MATNR">
        <textref tablename="MAKT" reffieldname="MATNR"
            textfieldname="MAKTX" langfieldname="SPRAS"/>
      </fieldtoread>
      <fieldtoread name="KONDM">
        <textref tablename="T178T" reffieldname="KONDM"
            textfieldname="VTEXT" langfieldname="SPRAS"/>
      </fieldtoread>
      <fieldtoread name="SPART">
        <textref tablename="TSPAT" reffieldname="SPART"
            textfieldname="VTEXT" langfieldname="SPRAS"/>
      </fieldtoread>
      <fieldtoread name="WERKS">
```

```
      <textref tablename="T001W" reffieldname="WERKS"
                              textfieldname="NAME1"/>
    </fieldtoread>
    <fieldtoread name="CHARG"/>
    <fieldtoread name="PSTYV"/>
    <fieldtoread name="ERNAM"/>
    <fieldtoread name="NETWR"/>
  </doctable>
 </docspec>
 <table name="VBAK">
  <pkfield name="VBELN" fktablename="Auftraege"
                               fkfieldname="VBELN"/>
  <fieldtoread name="VTWEG">
    <textref tablename="TVTWT" reffieldname="VTWEG"
          textfieldname="VTEXT" langfieldname="SPRAS"/>
  </fieldtoread>
  <fieldtoread name="VKORG">
    <textref tablename="TVKOT" reffieldname="VKORG"
          textfieldname="VTEXT" langfieldname="SPRAS"/>
  </fieldtoread>
  <fieldtoread name="VDATU"/>
  <fieldtoread name="VKBUR">
    <textref tablename="TVKBT" reffieldname="VKBUR"
          textfieldname="BEZEI" langfieldname="SPRAS"/>
  </fieldtoread>
  <fieldtoread name="VKGRP">
    <textref tablename="TVGRT" reffieldname="VKGRP"
          textfieldname="BEZEI" langfieldname="SPRAS"/>
  </fieldtoread>
  <fieldtoread name="VBTYP"/>
  <fieldtoread name="AUART"/>
 </table>
 <table name="MARA">
  <pkfield name="MATNR" fktablename="Auftraege"
                               fkfieldname="MATNR"/>
  <fieldtoread name="MATNR"/>
  <fieldtoread name="MTART">
    <textref tablename="T134T" reffieldname="MTART"
          textfieldname="MTBEZ" langfieldname="SPRAS"/>
  </fieldtoread>
 </table>
</configuration>
```

FREMDSCHLÜSSELTABELLE

```
...
<docreftable name="VBAK">
 <booleancondition>
  <condition fieldname="ERDATCHAR#-#ERZETCHAR"
             logicaloperator="char_creationtimestamp">
    <value>yyyyMMdd</value>
    <value>HHmmss</value>
  </condition>
  <condition fieldname="ERDAT#-#ERZET"
             logicaloperator="date_creationtimestamp">
    <value>DATE</value>
    <value>TIME</value>
  </condition>
  <condition fieldname="ERDAT_DATE"
                         logicaloperator="date_geq">
    <value>05.12.2001</value>
```

```
    </condition>
    <condition fieldname="ERZET_TIME"
                          logicaloperator="time_gt">
      <value>22:27:13</value>
    </condition>
    <condition fieldname="VBTYP"
                               logicaloperator="eq">
      <value>C</value>
    </condition>
  </booleancondition>
  <pkfield name="VBELN"/>
</docreftable>
...
```

Identifizierer und Name der Fremdschlüsseltabelle des SQL-Datenbanksystems ist jeweils **VBAK**. Es werden alle Auftragsbelege des Typs **VBTYP=C** gelesen. Die **value**-Elemente der Bedingungsoperatoren **char_creationtimestamp** legen das Format der aus den entsprechenden Datenbankfeldern zu erzeugenden Zeitstempel fest. Die **value**-Elemente der Bedingungsoperatoren **date_creationtimestamp** legen die Datentypen der ausgelesenen Felder fest. Dabei sind die Werte der Tabellenspalte **ERDAT** vom Typ **DATE** und die der Spalte **ERZET** vom Typ **TIME**.

Es werden nur die Tabellenzeilen ausgelesen, in denen die Werte der Attribute **ERDAT_DATE** und **ERZET_TIME** größer/gleich bzw. größer als die mit **value** angegebenen Werte (**05.12.2001** bzw. **22:27:13**) sind.

Name der Primärschlüsseltabellenspalte ist **VBELN**. Für jeden unterschiedlichen Feldwert **VBELN** werden aus der verknüpften System-Event-Tabelle Datensätze gelesen, für die **VBELN** denselben Wert wie in der Fremdschlüsseltabelle hat.

SYSTEM-EVENT-TABELLE

```
...
<doctable name="Auftraege" tablename="VBAP">
  <pkfield name="VBELN" fktablename="VBAK"
                               fkfieldname="VBELN"/>
  <pkfield name="POSNR"/>
  <fieldtoread name="VBELN"/>
  <fieldtoread name="POSNR"/>
  <fieldtoread name="ERDAT"/>
  <fieldtoread name="ERZET"/>
  <fieldtoread name="MATNR">
    <textref tablename="MAKT" reffieldname="MATNR"
           textfieldname="MAKTX" langfieldname="SPRAS"/>
  </fieldtoread>
  <fieldtoread name="KONDM">
    <textref tablename="T178T" reffieldname="KONDM"
           textfieldname="VTEXT" langfieldname="SPRAS"/>
  </fieldtoread>
  <fieldtoread name="SPART">
    <textref tablename="TSPAT" reffieldname="SPART"
           textfieldname="VTEXT" langfieldname="SPRAS"/>
  </fieldtoread>
  <fieldtoread name="WERKS">
    <textref tablename="T001W"
            reffieldname="WERKS" textfieldname="NAME1"/>
  </fieldtoread>
  <fieldtoread name="CHARG"/>
```

```
  <fieldtoread name="PSTYV"/>
  <fieldtoread name="ERNAM"/>
  <fieldtoread name="NETWR"/>
</doctable>
...
```

Für die System-Event-Tabelle **VBAP** wird der Identifizierer **Auftraege** vergeben. Durch die in der Zeile

```
<pkfield name="VBELN" fktablename="VBAK"
                                fkfieldname="VBELN"/>
```

angegebene Fremdschlüsselbeziehung ist die Tabelle mit der Fremdschlüsseltabelle **VBAK** verknüpft. Der Primärschlüssel wird aus den Spalten **VBELN** und **POSNR** zusammengesetzt.

Für jede gelesene Tabellenzeile wird in der XML-Ausgabedatei ein System-Event (XML-Element **event**) erzeugt. Für jedes XML-Element **fieldtoread** wird eine Zeile der Form

```
<attribute type="...">...</attribute>
```

in die Ausgabedatei geschrieben. Bei einigen **fieldtoread**-Elementen wird zusätzlich zum gelesenen Datenfeldwert der aus der referenzierten Tabelle (XML-Element **textref**) gelesene Wert geschrieben. Die optionale Angabe **langfieldname** liest den Text des Datenfeldes sprachabhängig aus.
Der vollständige Quellsystemattributtyp setzt sich zusammen aus dem Identifizierer der System-Event-Tabelle (**doctable name**), dem Feldnamen (**fieldtoread name**) und dem Namen des referenzierten Textfeldes (**textref ... textfieldname**). Der Typ des Quellsystemattributes des ersten gelesenen **fieltoread**-Elements mit referenzierter Tabelle sieht folgendermaßen aus:

```
<attribute type="Auftraege-MATNR-MAKTX">...</attribute>
```

DATENTABELLE

```
...
<table name="VBAK">
  <pkfield name="VBELN" fktablename="Auftraege"
                                fkfieldname="VBELN"/>
  <fieldtoread name="VTWEG">
    <textref tablename="TVTWT" reffieldname="VTWEG"
           textfieldname="VTEXT" langfieldname="SPRAS"/>
  </fieldtoread>
  <fieldtoread name="VKORG">
    <textref tablename="TVKOT" reffieldname="VKORG"
           textfieldname="VTEXT" langfieldname="SPRAS"/>
  </fieldtoread>
  <fieldtoread name="VDATU"/>
  <fieldtoread name="VKBUR">
    <textref tablename="TVKBT" reffieldname="VKBUR"
           textfieldname="BEZEI" langfieldname="SPRAS"/>
  </fieldtoread>
  <fieldtoread name="VKGRP">
    <textref tablename="TVGRT" reffieldname="VKGRP"
```

```
          textfieldname="BEZEI" langfieldname="SPRAS"/>
  </fieldtoread>
  <fieldtoread name="VBTYP"/>
  <fieldtoread name="AUART"/>
</table>
<table name="MARA">
  <pkfield name="MATNR" fktablename="Auftraege"
                              fkfieldname="MATNR"/>
  <fieldtoread name="MATNR"/>
  <fieldtoread name="MTART">
    <textref tablename="T134T" reffieldname="MTART"
          textfieldname="MTBEZ" langfieldname="SPRAS"/>
  </fieldtoread>
</table>
...
```

Über die Fremdschlüsselbeziehungen auf die Primärschlüsselfelder **VBELN** und **MATNR** der Tabelle **Auftraege** werden aus den Datentabellen **VBAK** und **MARA** ergänzende Informationen ausgelesen.

7.3.8.2 XML-Ausgabedatei (PPM-System-Event-Format)

Der folgende Dateiauszug zeigt einige System-Events der XML-Ausgabedatei, die unter Verwendung der im Kapitel **Tabellenkonfiguration** (Seite 98) angegebenen Konfigurationsdatei erzeugt wurde.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="Auftraege-CHARG"/>
    <attribute type="Auftraege-ERDAT">20000214</attribute>
    <attribute type="Auftraege-ERNAM">HDM</attribute>
    <attribute type="Auftraege-ERZET">143616</attribute>
    <attribute type="Auftraege-KONDM"/>
    <attribute type="Auftraege-MATNR">P-100</attribute>
    <attribute type="Auftraege-MATNR-MAKTX">
           Gewindestab M'8 DIN '8895' ohne Toleranz
    </attribute>
    <attribute type="Auftraege-NETWR">28.70</attribute>
    <attribute type="Auftraege-POSNR">000010</attribute>
    <attribute type="Auftraege-PSTYV">TAN</attribute>
    <attribute type="Auftraege-SPART">01</attribute>
    <attribute type="Auftraege-SPART-VTEXT">Produktsparte 01
    </attribute>
    <attribute type="Auftraege-VBELN">0000000001</attribute>
    <attribute type="Auftraege-WERKS">1000</attribute>
    <attribute type="Auftraege-WERKS-NAME1">
                  Werk 1000 (Hamburg)
    </attribute>
    <attribute type="MARA-MATNR">P-100</attribute>
    <attribute type="MARA-MTART">HAWA</attribute>
    <attribute type="MARA-MTART-MTBEZ">
                  Handelsware
    </attribute>
    <attribute type="VBAK-AUART">TA</attribute>
    <attribute type="VBAK-VBTYP">C</attribute>
    <attribute type="VBAK-VDATU">20000214</attribute>
    <attribute type="VBAK-VKBUR"/>
```

```
  <attribute type="VBAK-VKGRP"/>
  <attribute type="VBAK-VKORG">1000</attribute>
  <attribute type="VBAK-VKORG-VTEXT">Deutschl. Frankfurt
  </attribute>
  <attribute type="VBAK-VTWEG">10</attribute>
  <attribute type="VBAK-VTWEG-VTEXT">Endkundenverkauf
  </attribute>
</event>
<event>
  <attribute type="VBAK-VKORG-VTEXT">Deutschl. Frankfurt
  </attribute>
  <attribute type="VBAK-VTWEG-VTEXT">Endkundenverkauf
  </attribute>
  <attribute type="Auftraege-SPART">01</attribute>
  <attribute type="Auftraege-PSTYV">TAD</attribute>
  <attribute type="Auftraege-MATNR">SERVICE</attribute>
  <attribute type="Auftraege-ERDAT">20000214</attribute>
  <attribute type="VBAK-VKBUR"/>
  <attribute type="VBAK-VTWEG">10</attribute>
  <attribute type="Auftraege-SPART-VTEXT">Produktsparte 01
  </attribute>
  <attribute type="Auftraege-ERZET">143616</attribute>
  <attribute type="VBAK-VKORG">1000</attribute>
  <attribute type="Auftraege-POSNR">000020</attribute>
  <attribute type="MARA-MTART-MTBEZ">Dienstleistung
  </attribute>
  <attribute type="Auftraege-WERKS-NAME1">
                Werk 1000 (Hamburg)
  </attribute>
  <attribute type="Auftraege-CHARG"/>
  <attribute type="Auftraege-WERKS">1000</attribute>
  <attribute type="VBAK-VDATU">20000214</attribute>
  <attribute type="Auftraege-MATNR-MAKTX">
                    Reparatur
  </attribute>
  <attribute type="Auftraege-VBELN">0000000001</attribute>
  <attribute type="VBAK-AUART">TA</attribute>
  <attribute type="VBAK-VKGRP"/>
  <attribute type="Auftraege-NETWR">179.00</attribute>
  <attribute type="MARA-MTART">DIEN</attribute>
  <attribute type="VBAK-VBTYP">C</attribute>
  <attribute type="Auftraege-ERNAM">HDM</attribute>
  <attribute type="MARA-MATNR">SERVICE</attribute>
  <attribute type="Auftraege-KONDM"/>
</event>
<event>
  <attribute type="VBAK-VKGRP-BEZEI">GR. F2 Hr. Mayer
  </attribute>
  <attribute type="VBAK-VKORG-VTEXT">Deutschl. Frankfurt
  </attribute>
  <attribute type="VBAK-VTWEG-VTEXT">Endkundenverkauf
  </attribute>
  <attribute type="Auftraege-SPART">01</attribute>
  <attribute type="Auftraege-PSTYV">TAN</attribute>
  <attribute type="Auftraege-MATNR">P-100</attribute>
  <attribute type="Auftraege-ERDAT">20000214</attribute>
  <attribute type="VBAK-VKBUR">1000</attribute>
  <attribute type="VBAK-VTWEG">10</attribute>
  <attribute type="Auftraege-SPART-VTEXT">Produktsparte 01
  </attribute>
  <attribute type="Auftraege-ERZET">131257</attribute>
```

```
    <attribute type="VBAK-VKBUR-BEZEI">Büro Frankfurt
    </attribute>
    <attribute type="VBAK-VKORG">1000</attribute>
    <attribute type="Auftraege-POSNR">000010</attribute>
    <attribute type="MARA-MTART-MTBEZ">
                    Handelsware
    </attribute>
    <attribute type="Auftraege-WERKS-NAME1">
                  Werk 1000 (Hamburg)
    </attribute>
    <attribute type="Auftraege-CHARG"/>
    <attribute type="Auftraege-WERKS">1000</attribute>
    <attribute type="VBAK-VDATU">20000214</attribute>
    <attribute type="Auftraege-MATNR-MAKTX">
          Gewindestab M'8 DIN '8895' ohne Toleranz
    </attribute>
    <attribute type="Auftraege-VBELN">0000000002
     </attribute>
    <attribute type="VBAK-AUART">TA</attribute>
    <attribute type="VBAK-VKGRP">101</attribute>
    <attribute type="Auftraege-NETWR">28.70</attribute>
    <attribute type="MARA-MTART">HAWA</attribute>
    <attribute type="VBAK-VBTYP">C</attribute>
    <attribute type="Auftraege-ERNAM">HDM</attribute>
    <attribute type="MARA-MATNR">P-100</attribute>
    <attribute type="Auftraege-KONDM"/>
  </event>
  ...
</eventlist>
```

7.3.9 Blockweises Auslesen

Sie können das Verhältnis zwischen Ausführ- und Speichereffizienz des Auslesevorgangs beeinflussen.
Der Wert des Kommandozeilenarguments **-cpd** bestimmt die Anzahl der gleichzeitig im Systemspeicher gehaltenen System-Events. Die maximal mögliche Anzahl ist von der Größe des zur Verfügung stehenden Systemspeichers abhängig.

Der tatsächlich belegte Systemspeicher hängt von folgenden Faktoren ab:

- Anzahl der gesamten im definierten Analysezeitraum zu lesenden System-Events
- Anzahl der parallel im System gehaltenen System-Events (concurrently processed documents)
- Anzahl der zu lesenden Datenfelder (XML-Elemente **fieldtoread** und **reftext**)
- Speicherbedarf der gelesenen Datenfelder

7.3.10 Auslesen der ersten bzw. letzten Zeile einer Sortierung

Die zu verwendende Java-Klasse
com.idsscheer.ppm.xmlextractortools.extractor.jdbc2ppm. ↵
ZSortWithInteger_jdbc2ppm sortiert Datensätze in einer Datentabelle mithilfe des angegebenen Sortierkriteriums und schreibt entweder den ersten oder letzten Datensatz in die zu

erzeugenden System-Events. Die Klasse ist im Attribut **classtouse** des XML-Elements **table** anzugeben.

Beispiel

Mit der Klasse werden über Fremdschlüsselbeziehungen referenzierte Datensätze in der Tabelle **AUFTRAGSPOSITION** mittels eines ganzzahligen Kriteriums (Betrag der Position) sortiert. Als Sortierkriterium wird das Feld **BETRAG** verwendet. Aus der Sortierung wird schließlich einmal der Datensatz mit dem größten Wert des Sortierkriteriums (für **POSITIONSBETRAG_MAX**) und ein andermal mit dem niedrigsten Wert (für **POSITIONSBETRAG_MIN**) – jeweils mit der entsprechenden Positionsnummer – in die erzeugten System-Events geschrieben. Der folgende Dateiauszug veranschaulicht die Konfiguration zum Auslesen unter den o. g. Bedingungen:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE jdbc_tableconfiguration SYSTEM ↵
                  'jdbc_tableconfiguration.dtd'>
<jdbc_tableconfiguration>
  <configuration name="SortWithInteger">
    <docspec>
      <doctable name="AUFTRAGSKOPF">
        <pkfield name="NUMMER"/>
      </doctable>
    </docspec>
    <table name="POSITIONSBETRAG_MAX" ↵
          tablename="AUFTRAGSPOSITION" ↵
          classtouse="com.idsscheer.ppm.xmlextractortools. ↵
                    extractor.jdbc2ppm. ↵
                    ZSortWithInteger_jdbc2ppm">
      <parameter name="SORTCRITERION">
        <value>BETRAG</value>
      </parameter>
      <parameter name="USE">
        <value>MAX</value>
      </parameter>
      <pkfield name="NUMMER" fktablename="AUFTRAGSKOPF" ↵
                              fkfieldname="NUMMER"/>
      <fieldtoread name="POSITION"/>
      <fieldtoread name="BETRAG"/>
    </table>
    <table name="POSITIONSBETRAG_MIN" ↵
          tablename="AUFTRAGSPOSITION" ↵
          classtouse="com.idsscheer.ppm.xmlextractortools. ↵
                    extractor.jdbc2ppm. ↵
                    ZSortWithInteger_jdbc2ppm">
      <parameter name="SORTCRITERION">
        <value>BETRAG</value>
      </parameter>
      <parameter name="USE">
        <value>MIN</value>
      </parameter>
      <pkfield name="NUMMER" fktablename="AUFTRAGSKOPF" ↵
                              fkfieldname="NUMMER"/>
      <fieldtoread name="POSITION"/>
      <fieldtoread name="BETRAG"/>
    </table>
  </configuration>
</jdbc_tableconfiguration>
```

Ein System-Event, das mit dieser Konfiguration ausgelesen wird, sieht bspw. folgendermaßen aus:

```
<event>
  <attribute type="AUFTRAGSKOPF-NUMMER">
    4711
  </attribute>
  <attribute type="POSITIONSBETRAG_MAX-POSITION">
    20
  </attribute>
  <attribute type="POSITIONSBETRAG_MAX-BETRAG">
    100000
  </attribute>
  <attribute type="POSITIONSBETRAG_MIN-POSITION">
    50
  </attribute>
  <attribute type="POSITIONSBETRAG_MIN-BETRAG">
    470
  </attribute>
</event>
```

Das Sortierkriterium (hier: **BETRAG**) ist automatisch Bestandteil der System-Event-Spezifikation.

Die Vorlage für die Erstellung der gezeigten Konfiguration sieht folgendermaßen aus:

```
<table name="..." tablename="..." ↵
      classtouse="com.idsscheer.ppm.xmlextractortools. ↵
                  extractor.jdbc2ppm. ↵
                  ZSortWithInteger_jdbc2ppm">
  <parameter name="SORTCRITERION">
    <value>...</value>
  </parameter>
  <parameter name="USE">
    <value>...</value>
  </parameter>
  ...
  Siehe Kap.
  JDBC-Tabellenkonfiguration (Seite 74)
  ...
</table>
```

Folgende Tabelle zeigt noch einmal die wichtigsten Konfigurationseinträge der obigen **table**-Definition:

XML-Element /-Attribut	Wert: Beschreibung
name	Angegebener Name wird den gelesenen Quellsystemattributen als Präfix vorangestellt
tablename (optional)	Tabelle, aus der Informationen gelesen werden sollen. Vorgabewert: Wert aus XML-Attribut **name**
classtouse	**com.idsscheer.ppm.xmlextractortools.extractor.jdbc2ppm.ZSortWithInteger_jdbc2ppm**: Zu verwendende Java-Klasse
parameter name	
SORTCRITERION	Erster anzugebender Parameter. Genau ein XML-Element **value** muss angegeben werden (Feldname des Sortierkriteriums). Das angegebene Feld darf nur Integer-Werte enthalten.
USE	Zweiter anzugebender Parameter. Genau ein XML-Element **value** muss angegeben werden. Mögliche Werte: **MIN** (es wird der Datensatz mit dem kleinsten Integer-Wert des Sortierkriteriums gewählt) **MAX** (es wird der Datensatz mit dem größten Integer-Wert des Sortierkriteriums gewählt) Aus der auf Basis des angegebenen Sortierkriteriums ermittelten Liste von Datensätzen wird entweder der erste oder letzte Datensatz zum Auslesen gewählt.

Setzen Sie die Klasse zum Auslesen der ersten bzw. letzten Zeile aus einer Sortierung nur begrenzt ein, da es teilweise aufgrund der Sortier- und Selektionsvorgänge zu Performance- und Speichereinbußen kommen kann.

7.3.11 Auslesen der ersten bzw. letzten Zeile mittels Zeitstempel

Mit Hilfe der Java-Klasse **com.idsscheer.ppm.xmlextractortools.extractor.jdbc2ppm.ZSortWithTimestamp_jdbc2ppm** können Daten nach Zeitstempel (z. B. Oracle-Datentyp **TIMESTAMP**) sortiert aus Tabellen ausgelesen. Der erste bzw. letzte gelesene Datensatz wird in ein Event geschrieben. Die Klasse wird im XML-Element **table** des XML-Attributs **classtouse** angegeben. Die beiden erforderlichen Parameter werden in entsprechenden XML-Elementen angegeben:

- Parameter **name="SORTCRITERION"** mit einem XML-Element **value**, dessen Wert den Feldnamen der auszulesenden Tabelle bestimmt.
- Parameter **name="USE"** mit einem XML-Element **value**, das als Wert entweder **MAX** (Datensatz mit dem spätesten Wert wird übernommen) oder **MIN** (Datensatz mit dem frühesten Wert wird übernommen) haben muss.

Das folgende Beispiel zeigt den Auszug aus einer solchen Tabellenkonfiguration.

```
<table ...
classtouse="com.idsscheer.ppm.xmlextractortools.extractor.jdbc2ppm.ZSortWithTimestamp_jdbc2ppm">
  <parameter name="SORTCRITERION">
    <value>...</value>
  </parameter>
  <parameter name="USE">
    <value>...</value>
  </parameter>
  ... conditions ...
  ... pkfields ...
  ... fieldstoread ...
</table>
```

Das angegebene Sortierkriterium wird automatisch zur Event-Ausgabe hinzugefügt.

Beispiel

Aus der Tabelle **DBO.WMPROCESSSTEP** (Statushistorie der Prozessschritte) sollen Felder des letzten Prozessschrittstatus zu jedem Prozessschritt ausgelesen werden. Abhängig vom Wert des Feldes **AUDITTIMESTAMP** (Oracle-Datentyp TIMESTAMP) soll der späteste Eintrag (**MAX**) übenommen werden.

```
<configuration name="AUFK_JCDS_AUFK">
...
  <table name="WMPROCESSSTEP_END" tablename="DBO.WMPROCESSSTEP"
classtouse="com.idsscheer.ppm.xmlextractortools.extractor.jdbc2ppm.ZSortWithTimestamp_jdbc2ppm">
    <parameter name="USE">
      <value>MAX</value>
    </parameter>
    <parameter name="SORTCRITERION">
      <value>AUDITTIMESTAMP</value>
    </parameter>
    <pkfield name="INSTANCEID" fktablename="WMPROCESSSTEP"
fkfieldname="INSTANCEID" />
    <pkfield name="INSTANCEITERATION" fktablename="WMPROCESSSTEP"
fkfieldname="INSTANCEITERATION" />
```

```
    <pkfield name="STEPID" fktablename="WMPROCESSSTEP" fkfieldname="STEPID" />
    <pkfield name="STEPITERATION" fktablename="WMPROCESSSTEP"
fkfieldname="STEPITERATION" />
    <fieldtoread name="AUDITTIMESTAMP" />
    <fieldtoread name="INSERTTIMESTAMP" />
    <fieldtoread name="STATUS" />
 </table>
...
</configuration>
```

Ein Event, das mit dieser Konfiguration ausgelesen wird, sieht bspw. folgendermaßen aus:

```
<event>
...
  <attribute type="WMPROCESSSTEP_END-AUDITTIMESTAMP">24.08.2010
09:51:13.477</attribute>
  <attribute type="WMPROCESSSTEP_END-INSERTTIMESTAMP">24.08.2010
09:51:13.550</attribute>
  <attribute type="WMPROCESSSTEP_END-STATUS">2</attribute>
 ...
</event>
```

AUSLESEN DER DATEN INKL. MILISEKUNDEN

Zeitstempel werden standardmäßig sekundengenau ausgelesen und sortiert. Gibt es mehrere Datensätze mit identischem Zeitstempelwert, wird zufällig einer dieser Datensätze zur Wertermittlung verwendet.

Sind Zeitstempel millisekundengenau in der Datenbank gespeichert, können die Zeitstempel millisekundengenau ausgelesen und sortiert werden. Hierfür können Sie im Attribut **precisionoftime** des XML-Elementes **databasesettings** der JDBC-Konfigurationsdatei den Wert **MILLISECOND** angeben. Der Standardwert ist **SECOND**.

Mit dem Wert **SECOND** werden alle Zeitstempel bzw. Zeiten im Format **dd.MM.yyyy HH:mm:ss** oder **HH:mm:ss** ausgelesen und in die Event-Datei geschrieben. Die Daten werden nach Zeitstempel sekundengenau sortiert.

Mit dem Wert **MILLISECOND** werden Zeitstempel bzw. Zeiten im Format **dd.MM.yyyy HH:mm:ss.SSS** oder **HH:mm:ss.SSS** extrahiert und in die Event-Datei geschrieben. Die Daten werden nach Zeitstempel millisekundengenau sortiert.

Der Standardwert ist **SECOND**, so dass bei einem Upgrade einer bereits existierenden Mandantenkonfiguration die Zeiten in der ursprünglich Genauigkeit ausgelesen werden.

Das folgende Beispiel zeigt einen Auszug aus der XML-JDBC-Konfigurationsdatei:

```
<jdbcconf>
  <databasesettings name="jdbc_oracle" dbtype="ORACLE"
precisionoftime="MILLISECOND">
    <driverclass>oracle.jdbc.driver.OracleDriver</driverclass>
    <maxconditionlength>2000</maxconditionlength>
    <fetchsize>1000</fetchsize>
  </databasesettings>
  <databaseconnection name="jdbc_oracle10_connection">
    <dbsettings>jdbc_oracle</dbsettings>
    <url>...</url>
    <user>...</user>
...
  </databaseconnection>
</jdbcconf>
```

7.3.12 Vervielfältigung von System Events in Tabellen

Beim Auslesen aus einer JDBC-Datenbank gibt es die Möglichkeit, basierend auf einer System-Event-Tabelle weitere System-Events zu erzeugen.

Auf Quellsystemebene kann es vorkommen, dass es zwischen zwei Tabellen eine 1:n-Beziehung gibt, die beim Ermitteln der System-Events nicht aufgelöst werden kann. Mit Hilfe der Klasse **com.idsscheer.ppm.xmlextractortools.extractor.jdbc2ppm.ZTableMultiplyEvents_jdbc2ppm** können aus einem System Event mehrere System Events erzeugt und somit die 1:n-Beziehung aufgelöst werden.

Beispiel

```
<configuration name="MultiplyEvents">
    <docspec>
      <doctable name="VBAK" tablename="VBAK">
       <condition fieldname="VBELN" logicaloperator="in">
         <value>0000006662</value>
         <value>0000006741</value>
       </condition>

       <pkfield name="VBELN"/>
      </doctable>
    </docspec>

    <table name="VBAP"
    classtouse="com.idsscheer.ppm.xmlextractortools.extractor.jdbc2ppm.ZTableMul
    tiplyEvents_jdbc2ppm">
      <pkfield name="VBELN" fktablename="VBAK" fkfieldname="VBELN"/>
      <fieldtoread name="POSNR"/>
      <fieldtoread name="MATNR"/>
    </table>

    <table name="MAKT">
       <condition fieldname="SPRAS" logicaloperator="eq">
        <value>de</value>
       </condition>

    <pkfield name="MATNR" fktablename="VBAP" fkfieldname="MATNR"/>
      <fieldtoread name="MAKTX"/>
    </table>
</configuration>
```

Folgend die zugehörige System-Event-Ausgabedatei.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
<event>
  <attribute type="VBAK-VBELN">0000006662</attribute>
</event>
<event>
  <attribute type="MAKT-MAKTX">Flatscreen MS 1460 P</attribute>
  <attribute type="VBAK-VBELN">0000006741</attribute>
  <attribute type="VBAP-MATNR">M-06</attribute>
  <attribute type="VBAP-POSNR">000010</attribute>
</event>
```

```
<event>
  <attribute type="MAKT-MAKTX">Flatscreen MS 1775P</attribute>
  <attribute type="VBAK-VBELN">0000006741</attribute>
  <attribute type="VBAP-MATNR">M-10</attribute>
  <attribute type="VBAP-POSNR">000020</attribute>
</event>
<event>
  <attribute type="MAKT-MAKTX">MAG PA/DX 175</attribute>
  <attribute type="VBAK-VBELN">0000006741</attribute>
  <attribute type="VBAP-MATNR">M-14</attribute>
  <attribute type="VBAP-POSNR">000030</attribute>
</event>
<event>
  <attribute type="MAKT-MAKTX">Jotachi SN4500</attribute>
  <attribute type="VBAK-VBELN">0000006741</attribute>
  <attribute type="VBAP-MATNR">M-18</attribute>
  <attribute type="VBAP-POSNR">000040</attribute>
</event>
</eventlist>
```

Folgendes Beispiel, ohne die beschriebene Klasse, verdeutlicht den Vorgang bei der Extraktion.

```
<configuration name="MultiplyEvents_Doctable_Only">
    <docspec>
      <doctable name="VBAK" tablename="VBAK">
       <condition fieldname="VBELN" logicaloperator="in">
        <value>0000006662</value>
        <value>0000006741</value>
       </condition>

       <pkfield name="VBELN"/>
      </doctable>
    </docspec>

    <table name="VBAP" >
      <pkfield name="VBELN" fktablename="VBAK" fkfieldname="VBELN"/>
      <fieldtoread name="POSNR"/>
      <fieldtoread name="MATNR"/>
    </table>

    <table name="MAKT">
       <condition fieldname="SPRAS" logicaloperator="eq">
        <value>de</value>
       </condition>

      <pkfield name="MATNR" fktablename="VBAP" fkfieldname="MATNR"/>
      <fieldtoread name="MAKTX"/>
    </table>
</configuration>
```

Da einem Eintrag in der Tabelle **VBAK** mehrere Einträge in der Tabelle **VBAP** zugeordnet sind, wird aus der Tabelle **VBAP** nur eine einzige, zufällige Zeile gelesen.

Folgend die zugehörige System-Event-Ausgabedatei.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
<event>
  <attribute type="VBAK-VBELN">0000006662</attribute>
```

```
</event>
<event>
  <attribute type="MAKT-MAKTX">Flatscreen MS 1775P</attribute>
  <attribute type="VBAK-VBELN">0000006741</attribute>
  <attribute type="VBAP-MATNR">M-10</attribute>
  <attribute type="VBAP-POSNR">000020</attribute>
</event>
</eventlist>
```

7.3.13 Felder einzelner Tabellen für Data Analytics extrahieren

Um für Data Analytics eine einfache Datenextraktion zu ermöglichen, gibt es die Möglichkeit den kompletten Inhalt einer Quellsystemtabelle in eine Datei im Event-Format zu extrahieren, die dann in einen Analyseraum von Data Analytics importiert werden kann.

Sie können auch die auszulesenden Daten einschränken (Seite 60), indem Sie Bedingungen zur Datenextraktion definieren.

Die auszulesende Tabelle wird nicht über die Tabellenkonfiguration konfiguriert, sondern in der Datenquellendatei selbst. Dazu gibt es in der Datei **datasource.dtd** folgende Einträge:

XML-Element/-Attribut	Beschreibung
analysistype	XML-Attribut: Muss den Wert **DATA_ANALYTICS** haben.
realmtable	Das umfassende XML-Element für die Konfiguration der Data-Analytics-Datenquelle
tablename	XML-Attribut des Elements **realmtable**: Name der Tabelle im Quellsystem
sourcetable	Das umfassende XML-Element für die Konfiguration der Data-Analytics-Quellsystemtabelle. Muss mindestens ein Element **sourcefield** enthalten.
tablename	XML-Attribut des Elements **sourcetable**: Name der Tabelle im Quellsystem
sourcefield	Enthält den Namen des Feldes der Quellsystemtabelle

Das Attribut **analysistype** des XML-Elements **datasource** muss den Wert **DATA_ANALYTICS** haben, wenn durch das Element **<realmtable>** eine Analyseraum-Tabelle angegeben ist (PROCESS der Standardwert).

Das Attribut **tablename** des Elements **<realmtable>** gibt den Tabellennamen der Zieltabelle in der Analyseraum-Konfiguration an. Dieser Tabellenname hat auf die Extraktion selbst keinen Einfluss, sondern wird nur vom PPM-Import ausgewertet.

Weitere Informationen zum Datenimport für Data Analytics erhalten Sie im Benutzerhandbuch PPM Data Analytics.

Das XML-Element **<realmtable>** enthält das optionale Element **<sourcetable>**, das die zu extrahierende Tabelle angibt. Die auszulesenden Spalten dieser Tabelle müssen im Element

<sourcefield> angegeben werden. Das Element **<sourcetable>** ist optional. Beim JDBC- oder SAP-Extraktor müssen genau eine Quelltabelle und mindestens eine Quellspalte angegeben werden, andernfalls wird während des Parsen der Datenquellendatei eine Fehlermeldung ausgegeben.

In einer Datenquellendefinition darf maximal eine Tabelle stehen. Es ist nicht möglich die Anzahl der Zeilen einzuschränken, sondern es werden immer alle Zeilen ausgelesen, einschließlich aller Zeilen mit gleichen Werten an den auszulesenden Spalten. Sollen beispielsweise die Spalten **Vorname** und **Nachname** ausgelesen werden und die Tabelle enthält zehn Einträge mit **Peter** und **Schmidt**, dann werden dafür auch zehn Events mit gleichen Attributwerten erzeugt.

Das folgende Beispiel verdeutlicht die Konfiguration:

```
<realmtable tablename="COMPANY_EMPLOYEE">
    <sourcetable tablename="EMPLOYEE">
       <sourcefield>EMPLOYEE_ID</sourcefield>
       <sourcefield>NAME</sourcefield>
    </sourcetable>
    ...
</realmtable>
<dataextraction>
    <outputfilename>..\custom\testclient\data\employee.xml</outputfilename>
</dataextraction>
...
<systemconfig>..\custom\testclient\SourceSystemConfig.xml</systemconfig>
```

Im Unterschied zum herkömmlichen JDBC- oder SAP-Extraktor erhalten die Attribute in der Event-Ausgabedatei keinen Tabellennamen als Präfix. Wird beispielsweise die Tabelle **EMPLOYEE** ausgelesen, erzeugte der Extraktor normalerweise Events der Art **<tabellenname>-<spaltenname>**:

```
<event>
    <attribute type="EMPLOYEE-EMPLOYEE_ID">4711</attribute>
    <attribute type="EMPLOYEE-NAME">Schmidt</attribute>
</event>
```

Das Auslesen einer Tabelle per Element **<realmtable>** erzeugt hingegen nur Events ohne Tabellennamen:

```
<event>
    <attribute type="EMPLOYEE_ID">4711</attribute>
    <attribute type="NAME">Schmidt</attribute>
</event>
```

NULL-WERTE IM MODUS DATA ANALYTICS

Ist beim Auslesen einer Analyseraum-Tabelle der Wert einer Spalte **null**, wird dieser nicht in das Event geschrieben. Gibt es zum obigen Beispiel noch beispielsweise eine Zeile **EMPLOYEE_ID = 4712** ohne Nachnamen, erzeugt der Extraktor folgende Events.

```
<event>
    <attribute type="EMPLOYEE_ID">4711</attribute>
    <attribute type="NAME">Schmidt</attribute>
</event>
<event>
```

```
    <attribute type="EMPLOYEE_ID">4712</attribute>
</event>
```

Sind hingegen alle auszulesenden Spaltenwerte **null**, wird das Event mit leeren Attributen rausgeschrieben:

```
<event>
    <attribute type="EMPLOYEE_ID">4711</attribute>
    <attribute type="NAME">Schmidt</attribute>
</event>
<event>
    <attribute type="EMPLOYEE_ID"></attribute>
    <attribute type="NAME"></attribute>
</event>
```

Gibt es mehrere solcher Zeilen, werden diese – im Unterschied zum herkömmlichen Auslesevorgang (analysistype=PROCESS) – entsprechend oft übernommen, so dass in der Event-Datei immer genauso viele <event>-Elemente stehen, wie es Zeilen in der Datentabelle des Quellsystems gibt.

7.3.13.1 Datenextraktion einschränken

Wenn Sie nur einen bestimmten Teil des Tabelleninhaltes auslesen möchten, können Sie Bedingungen zur Datenextraktion definieren. Sie können die auszulesende Datenmenge mittels Zeitstempel oder Integerwert (z. B einer Sequenz) beschränken.

Geben Sie dazu in der Datenquellen-Konfiguration die gewünschte Bedingung mittels des Elements **condition** und des Attributs **dataextractiontype** wie folgt an.

```
<datasource name="VBAP" type="JDBC" analysistype="DATA_ANALYTICS"
dataextractiontype="TIME_BASED">
      <realmtable tablename="VBAP">
            <sourcetable tablename="VBAP2">
                  <condition logicaloperator="char_creationtimestamp"
fieldname="AEDAT">
                        <value>yyyy-MM-dd</value>
                  </condition>
                  <sourcefield>AEDAT</sourcefield>
                  <sourcefield>ERDAT</sourcefield>
                  <sourcefield>ERNAM</sourcefield>
                  <sourcefield>ERZET</sourcefield>
                  <sourcefield>MATKL</sourcefield>
                  <sourcefield>MATNR</sourcefield>
                  <sourcefield>WERKS</sourcefield>
            </sourcetable>
      </realmtable>
...
</datasource>
```

Das Attribut **dataextractiontype** kann bei Data-Analytics-Datenquellen folgende Werte haben:

- COMPLETE: Es wird bei einem Zeitkriterium ab dem Datum **01.01.1990 00:00:00** bzw. bei einem Integer-Kriterium ab dem Wert **0** ausgelesen. Dies ist auch der Standard, wenn das Attribut nicht vorhanden ist.
- TIME_BASED: Es soll eine zeitstempelbasierte Datenextraktion durchgeführt werden.
- VALUE_BASED: Es soll eine wertbasierte Datenextraktion durchgeführt werden.

Es wird keine Überprüfung vorgenommen, ob die verwendete Datenextraktionsbedingung zu dem konfigurierten Datenextraktionstyp (dataextractiontype=) passt. Wird die Konfiguration mittels CTK erstellt, so wird die Korrektheit vom CTK sichergestellt. Wird die Datei von Hand konfiguriert und es wird eine nicht passende Bedingung verwendet, so kann es zu einem Fehler bei der Datenextraktion kommen.

7.4 Kommandozeilenprogramm

Die XML-Ausgabedatei(en) erzeugen Sie mit dem Kommandozeilenprogramm **runjdbc2ppm.bat**.

Der Aufruf des Programmes ohne Parameter oder mit **-h** oder **-?** gibt die Hilfe auf der Konsole aus, in der alle verfügbaren Optionen beschrieben sind.

7.4.1 Argumente des Kommandozeilenprogramms

7.4.1.1 Allgemeine Argumente

-VERSION

Gibt die Versionsnummer des verwendeten PPM Prozessextraktors aus. Andere angegebene Argumente werden ignoriert.

-INFORMATION YES|NO

Hier legen Sie fest, ob Informationen während des Imports ausgegeben werden sollen (yes) oder nicht (no). Voreingestellt ist **yes**.

-WARNING YES|NO

Hier legen Sie fest, ob Warnmeldungen während des Imports ausgegeben werden sollen (yes) oder nicht (no). Voreingestellt ist **yes**.

-ERROR YES|NO

Hier legen Sie fest, ob Fehlermeldungen während des Imports ausgegeben werden sollen (yes) oder nicht (no).Voreingestellt ist **yes**.

-PROTOCOLFILE <FILENAME>

Hier können Sie eine Protokolldatei festlegen, in die alle Meldungen während des Imports geschrieben werden. Wenn Sie eine Datei festlegen, werden am Bildschirm nur noch kritische Fehlermeldungen ausgegeben, die zum Programmabbruch führen.

-LANGUAGE <ISO-CODE>

Hier geben Sie die Sprache an, in der die Protokollinformationen ausgegeben werden sollen.

7.4.1.2 Quelldatenbankspezifische Argumente

-DATASOURCE <FILENAME> (OPTIONAL)

Hier geben Sie die Datenquelle an, die für den Auslesevorgang verwendet werden soll. Die XML-Datei enthält Angaben zu allen für den Extraktionsvorgang und die XML-Ausgabe zu verwendenden Dateien. Verwenden Sie diesen Parameter nicht zusammen mit einem der Parameter **-datasourcelist**, **-systemconfig**, **-tableconfig**, **-calcconfig**, **-outfile** bzw. **-nozip**, sondern anstelle dieser Parameter.

-DATASOURCELIST <FILENAME>

Mit Hilfe des Arguments **-datasourcelist** können mehrere Datenquellen auf einmal extrahiert werden. Dies entspricht dem mehrfachen, nacheinander Ausführen der Extraktion mittels des Arguments **-datasource**. Es werden nur JDBC-Datenquellen extrahiert.

Sieh auch Kapitel Extraktion mehrere Datenquellen (Seite 120).

-SYSTEMCONFIG <FILENAME> <CONFIGNAME>

Hier geben Sie den Namen der XML-Datei an, die die Systemkonfiguration enthält. Der Inhalt der Datei ist quelldatenbankspezifisch und kann mehrere Konfigurationen enthalten. Im zweiten Argument wird der Name der zu verwendenden Konfiguration angegeben. Enthält die XML-Datei nur eine Konfiguration, wird diese automatisch verwendet. In diesem Fall müssen Sie keinen Konfigurationsnamen angeben.

-TABLECONFIG <FILENAME> <CONFIGNAME>

Hier geben Sie den Namen der XML-Datei an, die die Tabellenkonfiguration enthält. Der Inhalt der Datei ist quelldatenbankspezifisch. Die Datei muss alle in der Systemkonfiguration angegebenen Tabellenkonfigurationen enthalten. Im zweiten Argument wird der Name der Tabellenkonfiguration angegeben. Enthält die XML-Datei nur eine Konfiguration, wird diese automatisch verwendet. In diesem Fall müssen Sie keinen Konfigurationsnamen angeben.

-CALCCONFIG <FILENAME> (OPTIONAL)

Hier geben Sie den Namen der XML-Datei an, mit der Sie Attribute der XML-Ausgabedatei ändern bzw. Attribute einschließlich Attributberechnungen hinzufügen können.

-BEGINDATE <DD.MM.YYYY> (OPTIONAL)

Hier geben Sie das Datum an, ab dem Daten aus der Quelldatenbank gelesen werden sollen. Wenn Sie den Parameter nicht verwenden, wird der Wert des XML-Attributs **lastreaddate** aus der angegebenen Systemkonfiguration bzw. Datenquelle verwendet. Das Format **yyyymmdd** wird unterstützt.

-BEGINTIME <HH:MM:SS> (OPTIONAL)

Hier geben Sie die Uhrzeit an, ab der Daten aus dem Quellsystem gelesen werden sollen. Wenn Sie den Parameter und **-begindate** nicht verwenden, wird der Wert des XML-Attributs **lastreadtime** aus der angegebenen Systemkonfiguration bzw. Datenquelle verwendet. Wenn Sie den Parameter nicht verwenden, aber **-begindate** verwenden, wird der Vorgabewert **000000** gesetzt.

-ENDDATE <DD.MM.YYYY> (OPTIONAL)

Hier geben Sie das Datum an, bis zu dem Daten aus der Quelldatenbank gelesen werden sollen. Wenn Sie den Parameter nicht verwenden, wird das aktuelle Datum verwendet. Das Format **yyyymmdd** wird unterstützt.

-ENDTIME <HH:MM:SS> (OPTIONAL)

Hier geben Sie die Uhrzeit an, bis zu der Daten aus der Quelldatenbank gelesen werden sollen. Wenn Sie diesen Parameter bei Verwendung von
-enddate nicht verwenden, wird der Vorgabewert **235959** gesetzt. Wenn Sie den Parameter und **-enddate** nicht verwenden, wird die aktuelle Uhrzeit verwendet.

Wird PPM Process Extractor JDBC-2-PPM ohne Angabe eines Zeitraumes gestartet, wird der Beginnzeitpunkt aus der angegebenen Konfigurationsdatei gelesen. Als Endzeitpunkt werden das aktuelle Datum und die aktuelle Uhrzeit verwendet.

-VALUECONSTRAINT <PARAMETER1> ... <PARAMETER4> (OPTIONAL)

Mit diesem Parameter schränken Sie die auszulesende Datenmenge anhand eines Integer-Kriteriums ein. Erlaubt sind die Vergleichsoperatoren "<", "<=", ">" bzw. ">=". Sie können den entsprechenden Feldwert mit einem Wert oder mit zwei Werten (Intervallangabe) vergleichen. Wurde bereits ein zuletzt gelesener Wert (**lastreadvalue**) in der Systemkonfiguration bzw. Datenquelle gespeichert, können Sie auch diesen Wert für den Vergleich verwenden (siehe Kap. **Fortlaufendes automatisiertes Auslesen** (Seite 119)).

Beispiele

```
-valueconstraint 25000020 "<="
Es werden nur die Datensätze ausgelesen, deren Integer-Wert des in der
Event-Spezifikation angegebenen Feldes kleiner bzw. gleich dem Wert 25000020 ist.

-valueconstraint 15000020 ">" 25000020 "<"
```

Intervallangabe: Es werden nur die Datensätze ausgelesen, deren Integer-Wert des in der Event-Spezifikation angegebenen Feldes **kleiner** als der Wert **25000020** und **größer** als der Wert **15000020** ist.

-SAVE_VALUE_MINIMUM (OPTIONAL)

Mit diesem Parameter geben Sie an, dass beim einschränkenden Auslesen mit **-valueconstraint** der kleinste zuletzt gelesene Wert in der verwendeten Konfigurationsdatei (Systemkonfiguration bzw. Datenquelle) als **lastreadvalue** gespeichert wird. Vorgabewert: Speichern des größten Wertes

-CPD <INT> (OPTIONAL)

Mit diesem Parameter spezifizieren Sie die Anzahl der parallel auszulesenden System-Events (concurrently processed documents). Mit dem angegebenen Wert werden die Primärschlüsselfelder aus der mit dem XML-Element **doctable** in der Tabellenkonfiguration referenzierten Tabelle schrittweise ausgelesen und im Java-Speicher des Extraktors gehalten. Eine höhere Wertangabe führt zu einer besseren Performance des Auslesevorganges und zu einer effizienteren Speichernutzung.
Wenn Sie den Parameter nicht angeben, wird der empfohlene Standardwert 20000 verwendet.

-PING (OPTIONAL)

Mit diesem Parameter testen Sie die Verbindung zur angegebenen Quelldatenbank. Es werden keine Daten ausgelesen.

-REMOVEEMPTY (OPTIONAL)

Mit diesem Parameter entfernen Sie ausgelesene Attribute ohne Werte vor der Attributtransformation.

7.4.1.3 Ausgabedateispezifische Argumente

-OUTFILE <FILENAME>

Mit diesem Parameter geben Sie den Namen der XML-Ausgabedatei an. Die Dateinamenerweiterung wird automatisch ergänzt. Standardmäßig wird die Datei als ZIP-Datei ausgegeben.

-OUTFILEENCODING <ENCODING> (OPTIONAL)

Mit diesem Parameter geben Sie das Encoding der XML-Ausgabedatei an. Vorgabewert: UTF-8

-NOZIP (OPTIONAL)

Mit diesem Parameter geben Sie an, dass die Ausgabedatei nicht als ZIP-Datei, sondern als XML-Datei ausgegeben wird.

-PIKIDATAMAPPING <FILENAME> <PCNAME> (OPTIONAL)

Mit diesem Parameter geben Sie im ersten Argument den Namen der XML-Datei an, die das Mapping enthält. Im zweiten Argument geben Sie den Namen der Kennzahlenreihe an, in der die prozessinstanzunabhängigen Kennzahlenreihen gespeichert werden sollen.

-DIMDATAMAPPING <FILENAME> <DIMNAME> (OPTIONAL)

Mit diesem Parameter geben Sie im ersten Argument den Namen der XML-Datei an, die das Mapping enthält. Im zweiten Argument geben Sie den Namen der Dimension an, für die die extrahierten Werte eingelesen werden sollen.

-SORTEVENTATTRIBUTES (OPTIONAL)

Mit diesem Parameter sortieren Sie die Quellsystemattribute in den System-Events anhand des Attributtyps in alphanumerischer Reihenfolge. Bei großen Datenmengen kann sich dieser Parameter negativ auf die Geschwindigkeit des Auslesevorgangs auswirken.

7.4.1.4 Fortlaufendes automatisiertes Auslesen

Sie können Daten lückenlos automatisiert auslesen, indem Sie die Parameter **-begindate**, **-begintime**, **-enddate**, **-endtime** auf der Kommandozeile weglassen bzw. **-valueconstraint** ohne Wert für den ersten Vergleichsoperator verwenden. Bei Einschränkungen des auszulesenden Datenbereichs mittels Integer-Werten wird entweder der kleinste oder größte zuletzt gelesene Wert in der verwendeten Konfigurationsdatei gespeichert bzw. aktualisiert. Bei zeitlicher Einschränkung des auszulesenden Datenbereichs wird der Zeitpunkt der letzten Datenextraktion in der Konfigurationsdatei gespeichert bzw. aktualisiert.
Folgende Voraussetzungen müssen gegeben sein:

- In der Event-Spezifikation ist eine korrekte Konfiguration zu **char_creationtimestamp** oder **date_creationtimestamp** bzw. **valueconstraint** für die auszulesenden Daten angegeben (siehe Kap. **Bedingungsoperatoren** (Seite 85))
- Sie verwenden für das fortlaufende automatisierte Auslesen immer den gleichen Kommandozeilenaufruf, z. B.:

```
runjdbc2ppm -datasource datasource.xml -valueconstraint ">"
```

- Achten Sie bei der Verwendung des Parameters **-valueconstraint** darauf, in allen Auslesevorgängen entweder nur den größten zuletzt gelesenen Wert zu speichern (vorgabemäßig) oder mittels Angabe von
  -save_value_minimum den kleinsten zuletzt gelesenen Wert.
  Der Wert für **lastreadvalue** in der Konfigurationsdatei wird nur aktualisiert, wenn bei Ausführen der Kommandozeile ohne Parameter
  -save_value_minimum der größte zuletzt gelesene Wert größer als der bisher gespeicherte Wert ist, bzw. bei Verwendung von
  -save_value_minimum, wenn der kleinste zuletzt gelesene Wert kleiner als der bisher gespeicherte Wert ist.

Beispiel 1 (lastreadvalue mit Vorgabewert "0")
`-valueconstraint ">="`
Es werden nur die Datensätze ausgelesen, deren Integer-Wert zum Einschränken des auszulesenden Datenbereichs **größer** bzw. **gleich** dem Wert ist, der in der Systemkonfiguration bzw. Datenquelle als zuletzt gelesener Wert (**lastreadvalue**) gespeichert ist. Ist kein Wert

gespeichert, wird der Vorgabewert **0** verwendet (im Beispiel würden alle Datensätze mit Werten des Integer-Kriteriums **>=0** gelesen werden). Als **lastreadvalue** wird entsprechend der Vorgabe (siehe auch Parameter **-save_value_minimum**) der größte zuletzt gelesene Wert gespeichert, hier z. B. **300**.

Beispiel 2 (lastreadvalue="40")

```
-valueconstraint ">=" 270 "<="
```

Es werden alle Datensätze ausgelesen, deren Integer-Wert zum Einschränken des auszulesenden Datenbereichs **>=40** und **<=270** ist. Nach dem Extraktionsvorgang wird der Wert für **lastreadvalue** z. B. auf **270** aktualisiert, wenn dies tatsächlich der größte zuletzt gelesene Wert ist.

Beispiel 3 (lastreaddate="19971231" lastreadtime="155959" lastreadvalue="270")

Durch erneuten Aufruf des Befehls

```
runjdbc2ppm -datasource datasource.xml -valueconstraint ">"
```

werden alle Datensätze gelesen, deren Integer-Wert zum Einschränken des auszulesenden Datenbereichs größer als **270** ist und deren Zeitstempel gleichalt oder jünger als der Startzeitpunkt der Datenextraktion (**31.12.1997 15:59:59 Uhr**) ist.

Als **lastreaddate**/**lastreadtime** wird der Startzeitpunkt der Datenextraktion in der Konfigurationsdatei eingetragen.

Durch das Weglassen der Parameter **-begindate**/**-begintime** lesen Sie stets die Datensätze aus, deren Zeitstempel jünger (größer) sind als die mittels **lastreaddate**/**lastreadtime** festgelegten zuletzt gelesenen Zeitstempelwerte. Als **lastreaddate**/**lastreadtime** werden aktuelles Datum bzw. aktuelle Uhrzeit eingetragen, wenn Sie die Parameter **-enddate** und **-endtime** nicht verwenden.

7.5 Extraktion mehrerer Datenquellen

Sie können mit Hilfe einer Datenquellenliste mehrere Datenquellen auf einmal extrahieren. Die Datenquellenliste kann in einer eigenen Konfigurationsdatei angegeben werden.

Jede in CTK neu angelegte JDBC-Datenquelle wird automatisch am Ende der Datenquellenliste des aktuellen Mandanten hinzugefügt. Bei der Extraktion mit der Kommandozeilenoption **-datasourcelist <filename>** werden die Daten dieser Datenquellen nacheinander extrahiert, so als wäre die Datenextraktion mehrmals mit der Option **-datasource <filename>** aufgerufen worden. Die Reihenfolge, in der die Extraktion der Datenquellen erfolgt, ist in der Konfigurationsdatei angegeben. In der Liste vorhandene Datenquellen eines anderen Typs als JDBC werden bei der Extraktion übersprungen.

Für die Fehlerbehandlung gilt Folgendes:

- Wird die Extraktion mit einer gültigen Datenquellenliste aufgerufen, die aber keine passenden Datenquellen enthält, so beendet sie sich ohne eine Fehlermeldung.

- Wird die Extraktion mit einer Datenquellenliste aufgerufen, die mehrere passende Datenquellen enthält und tritt bei der Extraktion einer dieser Datenquellen ein Fehler auf, der zum Abbruch dieses Vorgangs führt, so wird die Verarbeitung mit der nächsten Datenquelle fortgeführt, d. h. der Abbruch bei einer einzelnen Datenquelle führt nicht zu einem Komplettabbruch.
- Tritt bei mindestens einer Datenquelle ein Fehler auf, der bei der Einzel-Extraktion dieser Datenquelle zu einem Exit-Fehlerstatus geführt hätte, so liefert die Gesamtverarbeitung der Datenquellenliste den letzten dieser Exit-Fehlerstatus zurück.

7.6 Anhang

7.6.1 Unterstützte Datentypen

Die unterstützten Datenbanksysteme umfassen zahlreiche Datentypen, die es nicht in allen Datenbanksystemen gibt, bzw. die dort andere Namen oder eine andere Semantik haben.

Um ein nicht erwünschtes Verhalten zu vermeiden, finden Sie nachfolgend eine Liste der Datentypen, die mit PPM Process Extractor JDBC-2-PPM erfolgreich getestet wurden.

Datenbanksystem	Datentypen	Bemerkung
Oracle	CHAR, VARCHAR2, NCHAR, NVARCHAR2, NUMBER, LONG, DATE, TIMESTAMP	Die Datentypen CLOB, NCLOB, BINARY_FLOAT und BINARY_DOUBLE werden nicht unterstützt.
IBM DB2	VARCHAR, CHAR, BIGINT, REAL, DOUBLE, FLOAT, DATE, TIME, TIMESTAMP	Die Datentypen LONG VARCHAR und CLOB werden nicht unterstützt.
Microsoft SQL Server	CHAR, VARCHAR, NCHAR, NVARCHAR, INT, TINYINT, SMALLINT, BIGINT, REAL, FLOAT, DECIMAL, DATETIME, SMALLDATETIME	Die Datentypen LONG VARCHAR und CLOB werden nicht unterstützt.

7.6.2 Sonderzeichen und Groß- und Kleinschreibung

PPM Process Extractor JDBC-2-PPM unterstützt ab Version 9.9 die Verwendung von Sonderzeichen, Groß- und Kleinschreibung in Schema-, Tabellen- und Feldnamen in SQL-Datenbanken.

Damit Datenbanksysteme SQL-Statements mit Sonderzeichen in Schema-, Tabellen und Spaltennamen verarbeiten können sowie Unterschiede in Groß- und Kleinschreibung berücksichtigt werden, müssen diese geklammert werden, wie beispielsweise im folgenden SQL-Statement für eine SQL-Server Datenbank.

```
SELECT [Spalte mit Leerzeichen] FROM [USER.1].[Tabelle $%&]
```

Die Syntax kann sich je nach Datenbanksystem unterscheiden. Unter Oracle und DB2 etwa müssen Namen in Hochkomma gesetzt werden, unter SQL-Server können auch eckige Klammern verwendet werden.

Zur Unterstützung von Sonderzeichen und Groß- und Kleinschreibung enthält die Systemkonfiguration **JdbcConfig.dtd** das Attribut **non-standard-sql-identifiers (true | false)**. Ist das Attribut **non-standard-sql-identifiers** mit Wert **true** angegeben, führt das dazu, dass die Schema-, Tabellen- und Spaltennamen mit für den Datenbanktyp geeigneten Begrenzungszeichen geklammert werden.

```
<!ATTLIST databasesettings
    name ID #REQUIRED
    dbtype (ORACLE | DB2 | SQLSERVER | OTHER) #REQUIRED
    precisionoftime (SECOND|MILLISECOND) "SECOND"
    non-standard-sql-identifiers (true | false) "false"
```

Im jeweiligen Datenbanksystem dürfen Begrenzungszeichen nicht Teil von Datenbankbezeichnern wie Schema-, Tabellen- und Spaltennamen sein.

ABWÄRTSKOMPATIBILITÄT

Um die Kompatibilität mit bestehenden Konfigurationen zu gewährleisten wird beim Öffnen und Speichern bzw. der Migration einer Konfiguration in CTK dem Attribut **non-standard-sql-identifiers** der Wert **false** zugewiesen. Ein vorhandener Wert wird nicht automatisch überschrieben, kann aber manuell in CTK geändert werden.

Beim Neuanlegen einer Konfiguration setzt CTK diesen Wert standardmäßig auf **true**.

7.6.2.1 Sonderfall Schema- und Tabellenname

Das Auslesen von Tabellen, die sich in einem anderen Datenbankschema als dem des angemeldeten Benutzers befinden, ist im JDBC-Extraktor nur über ein Präfix des Tabellennamens möglich. Das Präfix wird mittels Punkt vom Tabellennamen im Attribut **tablename** in den verschiedenen Elementen der Tabellenkonfiguration (jdbc_tableconfiguration.dtd) getrennt.

Beispiel

```
<doctable name="customer"   tablename="USER1.CUSTOMER ">
```

Da der Punkt auch als Sonderzeichen im Schema- als auch im Tabellennamen vorkommen kann, muss jeder Punkt, der Teil des Schema- oder des Tabellennamens ist, mit einem

Backslash-Zeichen **** maskiert werden. Um Backslash-Zeichen erkennen zu können, die Teil des Namens sind, müssen diese immer durch doppeltes Vorkommen maskiert werden.

In den folgenden Beispielkonfigurationen sind Schema- und Tabellennamen damit eindeutig unterscheidbar.

Schemaname	Tabellenname	Syntax für Attribut tablename
USER	1.CUSTOMER	USER.1\.CUSTOMER
USER.1.1	ORDER.DE	USER\.1\.1.ORDER\.DE
USER.	ORDER	USER\..ORDER
USER\2	ORDER.DE	USER\\2.ORDER\.DE
USER\.1	ORDER\.44	USER\\\.1.ORDER\\\.44
USER	1.TABLE.1	USER.1.TABLE.1

Falls mehr als ein Punkt im Schema- und Tabellennamen vorkommt (siehe Beispiel in der letzten Tabellenzeile), der nicht mit dem Backslash maskiert ist, so wird der erste Punkt als Schematrenner interpretiert und die weiteren Punkte dem Tabellennamen zugeschlagen. Dies entspricht dem Verhalten vor Version 9.9 und wird aus Kompatibiltätsgründen beibehalten. Es wird trotzdem empfohlen, alle Punkte außer dem Schematrenner mit einem vorangestellten Backslash zu maskieren.

Die Maskierung jedes anderen Zeichens außer Punkt oder Backslash ist nicht zulässig.

Bei der Definition von Fremdschlüsseln gilt es zu beachten, dass sich der Wert im XML-Attribut **<fktablename>** auf das XML-Attribut **<name>** bezieht und nicht auf **<tablename>**.

Beispiel

```
<jdbc_tableconfiguration>
 <configuration name="Example">
   <docspec>
    <doctable name="USER.1.CUSTOMER" tablename="USER\.1.CUSTOMER">
     <condition fieldname="SEQUENCE" logicaloperator="in">
      <value>001</value>
      <value>099</value>
     </condition>
     <pkfield name="SEQUENCE"/>
    </doctable>
   </docspec>
   <table name="FK_TABLE">
    <pkfield name="SEQUENCE"
      fktablename="USER.1.CUSTOMER" fkfieldname="SEQUENCE"/>
    <fieldtoread name="FK_COLUMN1"/>
    <fieldtoread name="FK_COLUMN2"/>
   </table>
 </configuration>
</jdbc_tableconfiguration>
```

Fehlt das Attribut **tablename**, dann wird der Inhalt des Attributs **name** als Tabellenname zum Zugriff auf die Datenbank herangezogen. Die beiden Attributwerte für die Attribute **name** und **fktablename** werden jedoch als reine Zeichenketten interpretiert und demnach nicht hinsichtlich Trenner und Maskierungszeichen ausgewertet.

Ist beispielsweise die Tabelle nur mit Attribut **name** folgendermaßen spezifiziert:

<doctable name="USER\.CUSTOMER">.

Dann ist das gleichbedeutend mit

<doctable name="USER\.CUSTOMER" tablename="USER\.CUSTOMER">.

Der Extraktor würde demnach auf der Datenbank die Tabelle **USER.CUSTOMER** auslesen.

Falls Sonderzeichen im Tabellennamen vorkommen, wird empfohlen, diesen immer mit Hilfe des Attributs **tablename** anzugeben.

8 PPM Process Extractor CSV-2-PPM

Dieses Kapitel gibt einen Überblick über die Architektur, Funktionsweise und Konfiguration von PPM Process Extractor CSV-2-PPM.

8.1 CSV

Eine Datei im CSV-Format (Comma Separated Values) besteht aus den eigentlichen Datensätzen und einer optionalen Kopfzeile mit den Spaltenüberschriften. Die Spaltenüberschriften der Kopfzeile und die Datenfelder der Datensätze sind durch dasselbe Trennzeichen getrennt, in der Regel Semikolon. Ein Datenfeld darf das Trennzeichen nur enthalten, wenn es innerhalb eines maskierten Werts verwendet wird. Die Spaltenüberschriften werden von PPM Process Extractor CSV-2-PPM als Attributtypen, die Werte der Datensätze als Attributwerte umgesetzt.

Beispiel

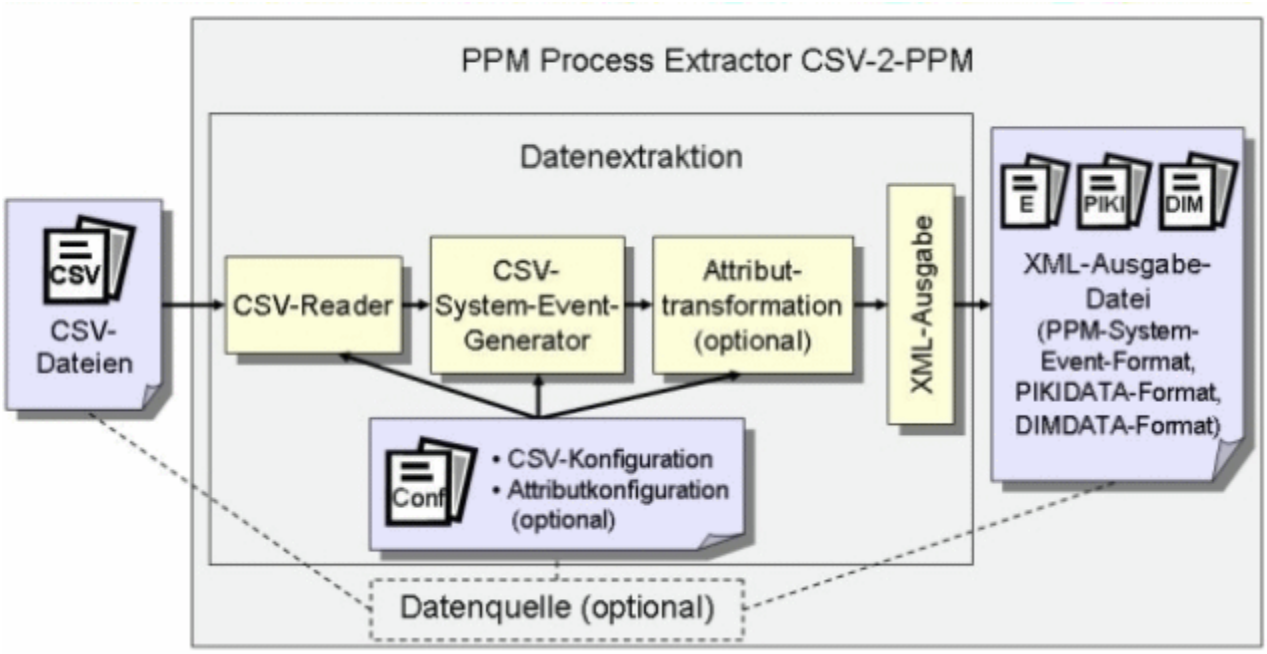
```
AUFTRAGSNUMMER,POSITION,ERFASSER,MATERIAL,MENGE
4711,10,"Harry, ""A"" Williams",Mobile 6600,3
4811,23,"Ben, ""B"" Snyder",Mobile 6601,2
4911,15,"George, ""C"" Nyland",Mobile 6602,1
```

Die CSV-Beispieldatei enthält eine Kopfzeile mit den Spaltenüberschriften für fünf Datenspalten sowie drei Datensätze.

Als Trennzeichen wird das Komma, als Maskierungszeichen das doppelte Hochkomma verwendet. Die Verwendung des Maskierungszeichens innerhalb eines maskierten Datenfeldwertes geschieht durch Verdoppelung des verwendeten Maskierungszeichens. Die Werte der Datenspalte **ERFASSER** dürfen das Trennzeichen enthalten, da sie maskiert sind.

8.2 Architektur

Die folgende Abbildung veranschaulicht die Funktionalität von PPM Process Extractor CSV-2-PPM:



Die CSV-Dateien werden mit Hilfe des CSV-Reader unter Verwendung der CSV-Konfiguration gelesen. Der CSV-System-Event-Generator und ggf. die Attributtransformation überführen die ausgelesenen Informationen wahlweise in das PPM-System-Event-Format, das PIKIDATA- oder das DIMDATA-Format. Die XML-Ausgabe schreibt die gelesenen Daten in die entsprechende XML-Ausgabedatei.

8.3 CSV-Reader und CSV-System-Event-Generator

Der CSV-Reader liest CSV-Dateien. Wenn Sie CSV-Dateien mit Kopfzeile einlesen möchten, müssen Sie in der CSV-Konfigurationsdatei eine entsprechende Angabe machen, damit die erste Zeile der CSV-Dateien als Kopfzeile interpretiert wird (siehe Kap. **CSV-Konfiguration** (Seite 129)).

Die Groß- bzw. Kleinschreibung der Spaltenüberschriften und Datenfeldwerte wird unverändert aus den CSV-Dateien übernommen.

In den folgenden Kapiteln wird das Verhalten des CSV-Reader und des CSV-System-Event-Generators anhand einfacher Beispiele beschrieben. In den Beispielen wird der CSV-Datei jeweils die entsprechende XML-Ausgabedatei im PPM-System-Event-Format gegenüber gestellt. Ein CSV-Datensatz wird mit den Spaltenüberschriften in ein System-Event der XML-Ausgabedatei im PPM-System-Event-Format transformiert.

8.3.1 Beispiel 1: Vollständige Kopfzeile

EINTRAG IN CSV-DATEI

```
AUFTRAGSNUMMER;POSITION;ERFASSER;MATERIAL;MENGE
4711;10;"Harry, ""A"" Williams";Mobile 6600;3
```

EINTRAG IN XML-AUSGABEDATEI

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
     <event>
     <attribute type="AUFTRAGSNUMMER">4711</attribute>
     <attribute type="POSITION">10</attribute>
     <attribute type="ERFASSER">Harry, "A" Williams</attribute>
     <attribute type="MATERIAL">Mobile 6600</attribute>
     <attribute type="MENGE">3</attribute>
     </event>
     ...
</eventlist>
```

Wenn Sie in der CSV-Konfigurationsdatei ein Maskierungszeichen angeben, werden beim Erzeugen der XML-Ausgabedatei alle Werte demaskiert.

8.3.2 Beispiel 2: Kopfzeile mit einer fehlenden Spaltenüberschrift

Wenn in den CSV-Dateien eine Spaltenüberschrift fehlt, wird diese in der XML-Ausgabedatei automatisch ergänzt. Es wird standardmäßig ein Spaltenname vergeben, der sich aus dem Präfix **FIELD_** und der Positionsnummer der Spalte zusammensetzt. Die erste Spalte hat die Position 1.

EINTRAG IN CSV-DATEI

```
AUFTRAGSNUMMER;POSITION;;MATERIAL;MENGE
4711;10;"Harry, ""A"" Williams";Mobile 6600;3
```

EINTRAG IN XML-AUSGABEDATEI

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="AUFTRAGSNUMMER">4711</attribute>
    <attribute type="POSITION">10</attribute>
    <attribute type="FIELD_3">Harry, "A" Williams
    </attribute>
    <attribute type="MATERIAL">Mobile 6600</attribute>
    <attribute type="MENGE">3</attribute>
  </event>
  ...
</eventlist>
```

Für die fehlende Spaltenüberschrift in den CSV-Daten wurde automatisch die Spaltenüberschrift **FIELD_3** erzeugt.

8.3.3 Beispiel 3: Kopfzeile mit mehreren fehlenden Spaltenüberschriften

Wenn in den CSV-Dateien mehrere Spaltenüberschriften fehlen, werden diese in der XML-Ausgabedatei automatisch ergänzt. Wenn sich die fehlenden Spaltenüberschriften am Ende der Kopfzeile befinden, müssen die Trennzeichen zwischen den fehlenden Spaltenüberschriften dazu nicht vorhanden sein.

EINTRAG IN CSV-DATEI

```
AUFTRAGSNUMMER;POSITION;;;
4711;10;"Harry, ""A"" Williams";Mobile 6600;3
```

oder

```
AUFTRAGSNUMMER;POSITION
4711;10;"Harry, ""A"" Williams";Mobile 6600;3
```

EINTRAG IN XML-AUSGABEDATEI

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="AUFTRAGSNUMMER">4711</attribute>
    <attribute type="POSITION">10</attribute>
    <attribute type="FIELD_3">Harry, "A" Williams</attribute>
    <attribute type="FIELD_4">Mobile 6600</attribute>
    <attribute type="FIELD_5">3</attribute>
  </event>
  ...
</eventlist>
```

Für die fehlenden Spaltenüberschriften in den CSV-Dateien wurden automatisch die Spaltenüberschriften **FIELD_3**, **FIELD_4**, **FIELD_5** erzeugt. Unabhängig vom Vorhandensein des Trennzeichens zwischen den fehlenden Spaltenüberschriften wird jeweils derselbe Eintrag in der XML-Ausgabedatei erzeugt.

8.3.4 Beispiel 4: Keine Kopfzeile

Wenn in den CSV-Dateien keine Kopfzeile vorhanden ist, wird für jede Datenspalte automatisch eine Spaltenüberschrift erzeugt.

EINTRAG IN CSV-DATEI

```
4711;10;"Harry, ""A"" Williams";Mobile 6600;3
```

EINTRAG IN XML-AUSGABEDATEI

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="FIELD_1">4711</attribute>
    <attribute type="FIELD_2">10</attribute>
    <attribute type="FIELD_3">Harry, "A" Williams</attribute>
    <attribute type="FIELD_4">Mobile 6600</attribute>
```

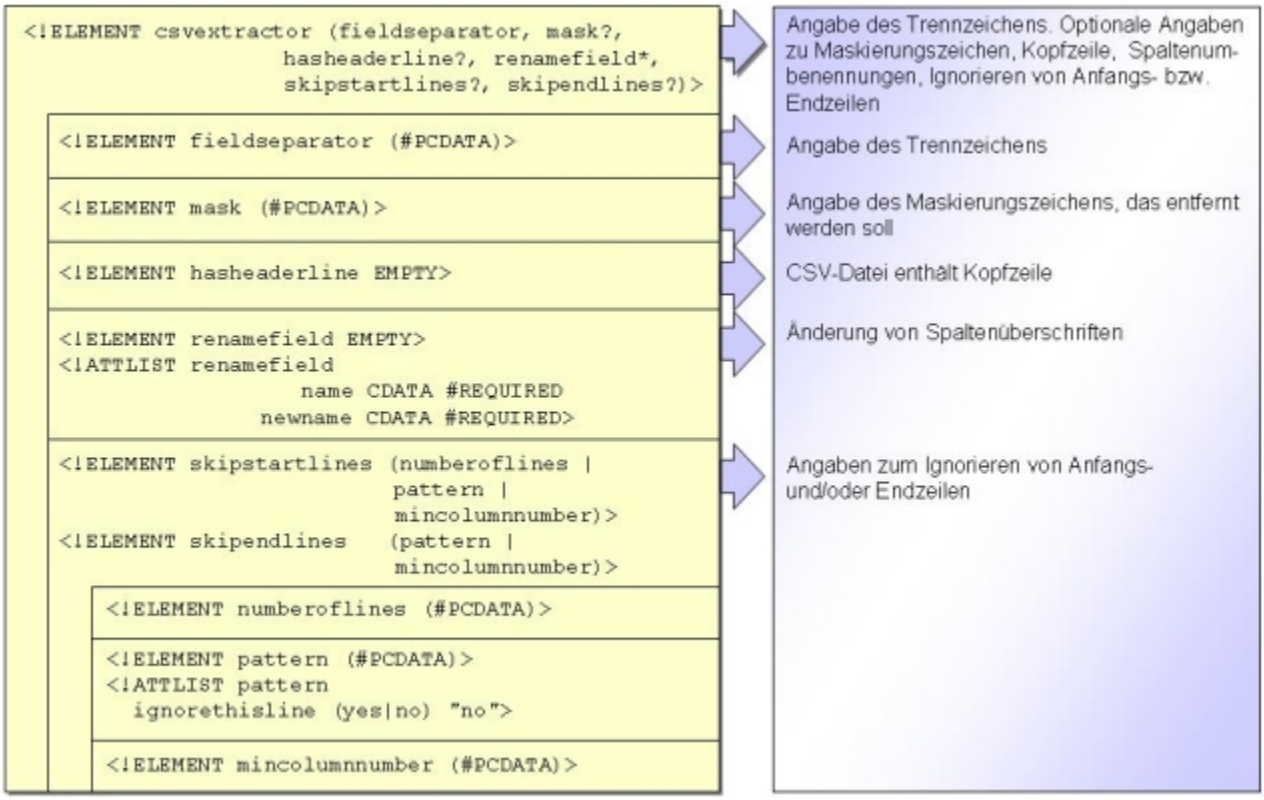
```
    <attribute type="FIELD_5">3</attribute>
  </event>
  ...
</eventlist>
```

8.4 CSV-Konfiguration

Die Art und Weise, in der CSV-Daten beim Einlesen verarbeitet werden, wird in einer XML-Konfigurationsdatei angegeben. Der Name dieser Datei wird dem Kommandozeilenprogramm **runcsv2ppm** als Argument übergeben.

In der Konfigurationsdatei können Sie z. B. das Trennzeichen bestimmen oder Spaltenüberschriften umbenennen.

Das Format der XML-Datei ist durch folgende DTD vorgegeben:



XML-Element bzw. -Attribut	Beschreibung	Beispiel
csvextractor	Konfiguration des CSV-Einlesevorgangs	Siehe unten
fieldseparator	Trennzeichen	,
mask	Maskierungszeichen (eingelesene Werte werden demaskiert)	"
hasheaderline	CSV-Dateien liegen mit Kopfzeile vor	

XML-Element bzw. -Attribut	Beschreibung	Beispiel
renamefield	Umbenennen einer Datenspalte	Siehe unten
name	Aus CSV-Dateien erzeugter Name einer Datenspalte	MATERIAL oder FIELD_3
newname	Neuer Name der Datenspalte	Artikel-bezeich-nung
skipstartlines	Zu ignorierende Anfangszeilen	
skipendlines	Zu ignorierende Endzeilen	
numberoflines	Anzahl an Anfangszeilen, die ignoriert werden sollen. Gezählt wird ab der ersten Zeile.	10
pattern	Zeichenmuster zum Auffinden einer Datenzeile, ab der Zeilen ausgelesen werden (**skipstartlines**: standardmäßig wird gefundene Zeile mit ausgelesen) bzw. bis zu der Zeilen ausgelesen werden (**skipendlines**: gefundene Zeile wird mit allen nachfolgenden Zeilen nicht ausgelesen). Folgende Platzhalter sind erlaubt: ***** Kein bzw. beliebig viele Zeichen **?** Genau ein beliebiges Zeichen **** Maskierungszeichen für die Suche nach Platzhaltern bzw. Maskierungs-zeichen in der Form: **** bzw. ***** bzw. **\?** Die erste Zeile von oben, die dem angegebenen Muster entspricht, wird ausgewählt.	Der Ausdruck ***10?*269** sucht Werte wie z. B. 555108*269 oder 104*269 aber z. B. nicht 104\\555269
ignorethisline (nur bei **pattern** in **skipstartlines**)	Datenzeile, die das gesuchte Muster enthält, im Auslesevorgang ignorieren (**yes**) bzw. nicht ignorieren (**no**). Vorgabewert: **no**	yes

XML-Element bzw. -Attribut	Beschreibung	Beispiel
mincolumn number	Angabe einer Mindestspaltenzahl zur Festlegung von Anfang und/oder Ende des auszulesenden Datenbereichs. **skipstartlines**: ab der ersten gefundenen Zeile (inklusiv) mit mindestens der angegebenen Spaltenzahl werden Zeilen ausgelesen. **skipendlines**: bis zur ersten gefundenen Zeile (exklusiv) mit weniger als der angegebenen Spaltenzahl werden Zeilen ausgelesen.	4

BEISPIELKONFIGURATION (CSVCONFIG.XML)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE csvextractor SYSTEM 'csvextractor.dtd'>
<csvextractor>
  <fieldseparator>;</fieldseparator>
  <mask>"</mask>
  <hasheaderline/>
  <renamefield name="FIELD_3" newname=
                       "Erfasser der Auftragsposition"/>
  <renamefield name="MATERIAL" newname="Artikelbezeichnung"/>
  <skipstartlines>
    <pattern ignorethisline="yes">###*</pattern>
  </skipstartlines>
  <skipendlines>
    <pattern>5???;*;*;*;*</pattern>
  </skipendlines>
</csvextractor>
```

EINZULESENDE CSV-DATEI (EXAMPLE.CSV)

Die Datei besteht aus einer Kommentarzeile, einer Kopfzeile und fünf Datensätzen:

```
### Auftragsdaten 25.03.2006 ###
AUFTRAGSNUMMER;POSITION;;MATERIAL;MENGE
4711;10;"Harry, ""A"" Williams";Mobile 6600;3
4811;23;"Ben, ""B"" Snyder";Mobile 6601;2
4911;15;"George, ""C"" Nyland";Mobile 6602;1
5011;6;"George, ""C"" Nyland";Mobile 5405;2
5211;23;"George, ""C"" Nyland";Mobile 5410;1
```

AUSGABEDATEI IM PPM-SYSTEM-EVENT-FORMAT (EVENT.XML)

Der CSV-System-Event-Generator erzeugt mithilfe der CSV-Beispielkonfiguration folgende XML-Ausgabedatei:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
```

```
  <event>
    <attribute type="AUFTRAGSNUMMER">4711</attribute>
    <attribute type="POSITION">10</attribute>
    <attribute type="Erfasser der Auftragsposition"
                  >Harry, "A" Williams</attribute>
    <attribute type="Artikelbezeichnung">Mobile 6600
    </attribute>
    <attribute type="MENGE">3</attribute>
  </event>
  <event>
    <attribute type="AUFTRAGSNUMMER">4811</attribute>
    <attribute type="POSITION">23</attribute>
    <attribute type="Erfasser der Auftragsposition"
                    >Ben, "B" Snyder</attribute>
    <attribute type="Artikelbezeichnung">Mobile 6601
    </attribute>
    <attribute type="MENGE">2</attribute>
  </event>
  <event>
    <attribute type="AUFTRAGSNUMMER">4911</attribute>
    <attribute type="POSITION">15</attribute>
    <attribute type="Erfasser der Auftragsposition"
                   >George, "C" Nyland</attribute>
    <attribute type="Artikelbezeichnung">Mobile 6602
    </attribute>
    <attribute type="MENGE">1</attribute>
  </event>
</eventlist>
```

Folgender Kommandozeilenaufruf erzeugt aus jedem Datensatz der CSV-Daten entsprechend den Angaben in der CSV-Konfigurationsdatei jeweils ein System-Event in der XML-Ausgabedatei:

```
runcsv2ppm -i example.csv -csvconfig csvconfig.xml -outfile event -nozip
```

Für alle System-Events wird der beim Einlesen automatisch für die dritte Datenspalte vergebene Name **FIELD_3** durch **Erfasser der Auftragsposition** ersetzt, die Datenspalte **MATERIAL** wird in **Artikelbezeichnung** umbenannt. Die maskierten Werte des Attributtyps **Erfasser der Auftragsposition** werden demaskiert. Die Kommentarzeile wird gemäß den Angaben zu **skipstartlines** im Auslesevorgang ignoriert. Die beiden letzten Zeilen des Datenbereichs werden gemäß den Angaben zu **skipendlines** ignoriert (Auftragsnummern **5011**, **5211**).

8.4.1 Erweiterung der CSV-Konfiguration

In der CSV-Quellsystemkonfiguration ist es möglich die CSV-Eingabedatei(en) und das Encoding der Eingabedatei(en) anzugeben. Das XML-Element **csvextractor** kann die folgenden beiden XML-Element enthalten:

XML-Element /-Attribut	Beschreibung
inputfile	Enthält den Pfad der Eingabedatei(en)
encoding	Enthält das Encoding der Eingabedatei(en)

Beispiel

```
<csvextractor>
    <inputfile>../testclient/data/customers.csv</inputfile>
    <encoding>ISO-8859-15</encoding>
    <fieldseparator>,</fieldseparator>
```

```
    <mask>"</mask>
    <hasheaderline />
</csvextractor>
```

Als Wert beim XML-Element **inputfile** kann der Pfad zur CSV-Datei angegeben werden. Wenn der Pfad angegeben wurde, wird der Kommandozeilenparameter (Seite 134) **-i** ignoriert. Dasselbe gilt für das XML-Element **encoding** und für den Kommandozeilenparameter (Seite 134) **-encoding**.

Wenn der Pfad zur CSV-Datei (inputfile) in der CSV-Konfiguration angegeben wurde, wird auch die Pfadangabe in der Datenquelle ignoriert.

8.5 Kommandozeilenprogramm

Das Erzeugen der XML-Ausgabedatei erfolgt mit dem Kommandozeilenprogramm **runcsv2ppm.bat**.

Der Aufruf des Programmes ohne Parameter oder mit **-h** oder **-?** gibt die Hilfe auf der Konsole aus. In ihr sind alle verfügbaren Optionen beschrieben.

8.5.1 Argumente des Kommandozeilenprogramms

8.5.2 Allgemeine Argumente

-VERSION

Gibt die Versionsnummer des verwendeten PPM Prozessextraktors aus. Andere angegebene Argumente werden ignoriert.

-INFORMATION YES|NO

Hier legen Sie fest, ob Informationen während des Imports ausgegeben werden sollen (yes) oder nicht (no). Voreingestellt ist **yes**.

-WARNING YES|NO

Hier legen Sie fest, ob Warnmeldungen während des Imports ausgegeben werden sollen (yes) oder nicht (no). Voreingestellt ist **yes**.

-ERROR YES|NO

Hier legen Sie fest, ob Fehlermeldungen während des Imports ausgegeben werden sollen (yes) oder nicht (no).Voreingestellt ist **yes**.

-PROTOCOLFILE <FILENAME>

Hier können Sie eine Protokolldatei festlegen, in die alle Meldungen während des Imports geschrieben werden. Wenn Sie eine Datei festlegen, werden am Bildschirm nur noch kritische Fehlermeldungen ausgegeben, die zum Programmabbruch führen.

-LANGUAGE <ISO-CODE>

Hier geben Sie die Sprache an, in der die Protokollinformationen ausgegeben werden sollen.

8.5.3 CSV-spezifische Argumente

-DATASOURCE <FILENAME> (OPTIONAL)

Hier geben Sie die Datenquelle an, die für den Auslesevorgang verwendet werden soll. Die XML-Datei enthält Angaben zu allen für den Extraktionsvorgang und die XML-Ausgabe zu verwendenden Dateien. Verwenden Sie diesen Parameter nicht zusammen mit einem der Parameter **-datasourcelist**, **-csvconfig**, **-i**, **-calcconfig**, **-outfile** bzw. **-nozip**, sondern anstelle dieser Parameter.

-DATASOURCELIST <FILENAME>

Mit Hilfe des Arguments **-datasourcelist** können mehrere Datenquellen auf einmal extrahiert werden. Dies entspricht dem mehrfachen, nacheinander Ausführen der Extraktion mittels des Arguments **-datasource**. Es werden nur CSV-Datenquellen extrahiert.

Sieh auch Kapitel Extraktion mehrere Datenquellen (Seite 135).

-I <CSVFILENAME> [<CSVFILENAME> ...]

Hier geben Sie die CSV-Datei(en) an, aus denen eine XML-Ausgabedatei erzeugt werden soll.

Wenn in der CSV-Konfigurationsdatei der Pfad zur CSV-Eingabedatei angegeben ist, wird der Kommandozeilenparameter **-i** ignoriert.

-CSVCONFIG <FILENAME>

Hier geben Sie den Namen der CSV-Konfigurationsdatei an, die die Angaben zum Erzeugen der XML-Ausgabedatei aus den CSV-Dateien enthält.

-ENCODING "<ENCODINGNAME>" (OPTIONAL)

Hier geben Sie das Encoding der CSV-Dateien an. Wenn Sie keine Angabe machen, wird **ISO-8859-15** verwendet.

Wenn in der CSV-Konfigurationsdatei das Encoding angegeben ist, wird der Kommandozeilenparameter **-encoding** ignoriert.

-CALCCONFIG <FILENAME> (OPTIONAL)

Hier geben Sie den Namen der XML-Konfigurationsdatei an, mit der Sie Attribute der XML-Ausgabedatei ändern bzw. Attribute einschließlich Attributtransformationen hinzufügen können.

-REMOVEEMPTY (OPTIONAL)

Mit diesem Parameter entfernen Sie ausgelesene Attribute ohne Werte vor der Attributtransformation.

8.5.4 Ausgabedateispezifische Argumente

-OUTFILE <FILENAME>

Mit diesem Parameter geben Sie den Namen der XML-Ausgabedatei an. Die Dateinamenerweiterung wird automatisch ergänzt. Standardmäßig wird die Datei als ZIP-Datei ausgegeben.

-OUTFILEENCODING <ENCODING> (OPTIONAL)

Mit diesem Parameter geben Sie das Encoding der XML-Ausgabedatei an. Vorgabewert: UTF-8

-NOZIP (OPTIONAL)

Mit diesem Parameter geben Sie an, dass die Ausgabedatei nicht als ZIP-Datei, sondern als XML-Datei ausgegeben wird.

-PIKIDATAMAPPING <FILENAME> <PCNAME> (OPTIONAL)

Mit diesem Parameter geben Sie im ersten Argument den Namen der XML-Datei an, die das Mapping enthält. Im zweiten Argument geben Sie den Namen der Kennzahlenreihe an, in der die prozessinstanzunabhängigen Kennzahlenreihen gespeichert werden sollen.

-DIMDATAMAPPING <FILENAME> <DIMNAME> (OPTIONAL)

Mit diesem Parameter geben Sie im ersten Argument den Namen der XML-Datei an, die das Mapping enthält. Im zweiten Argument geben Sie den Namen der Dimension an, für die die extrahierten Werte eingelesen werden sollen.

-SORTEVENTATTRIBUTES (OPTIONAL)

Mit diesem Parameter sortieren Sie die Quellsystemattribute in den System-Events anhand des Attributtyps in alphanumerischer Reihenfolge. Bei großen Datenmengen kann sich dieser Parameter negativ auf die Geschwindigkeit des Auslesevorgangs auswirken.

8.6 Extraktion mehrerer Datenquellen

Sie können mit Hilfe einer Datenquellenliste mehrere Datenquellen auf einmal extrahieren. Die Datenquellenliste kann in einer eigenen Konfigurationsdatei angegeben werden.

Jede in CTK neu angelegte CSV-Datenquelle wird automatisch am Ende der Datenquellenliste des aktuellen Mandanten hinzugefügt. Bei der Extraktion mit der Kommandozeilenoption **-datasourcelist <filename>** werden die Daten dieser Datenquellen nacheinander extrahiert, so als wäre die Datenextraktion mehrmals mit der Option **-datasource <filename>** aufgerufen worden. Die Reihenfolge, in der die Extraktion der Datenquellen erfolgt, ist in der Konfigurationsdatei angegeben. In der Liste vorhandene Datenquellen eines anderen Typs als CSV werden bei der Extraktion übersprungen.

Für die Fehlerbehandlung gilt Folgendes:

- Wird die Extraktion mit einer gültigen Datenquellenliste aufgerufen, die aber keine passenden Datenquellen enthält, so beendet sie sich ohne eine Fehlermeldung.

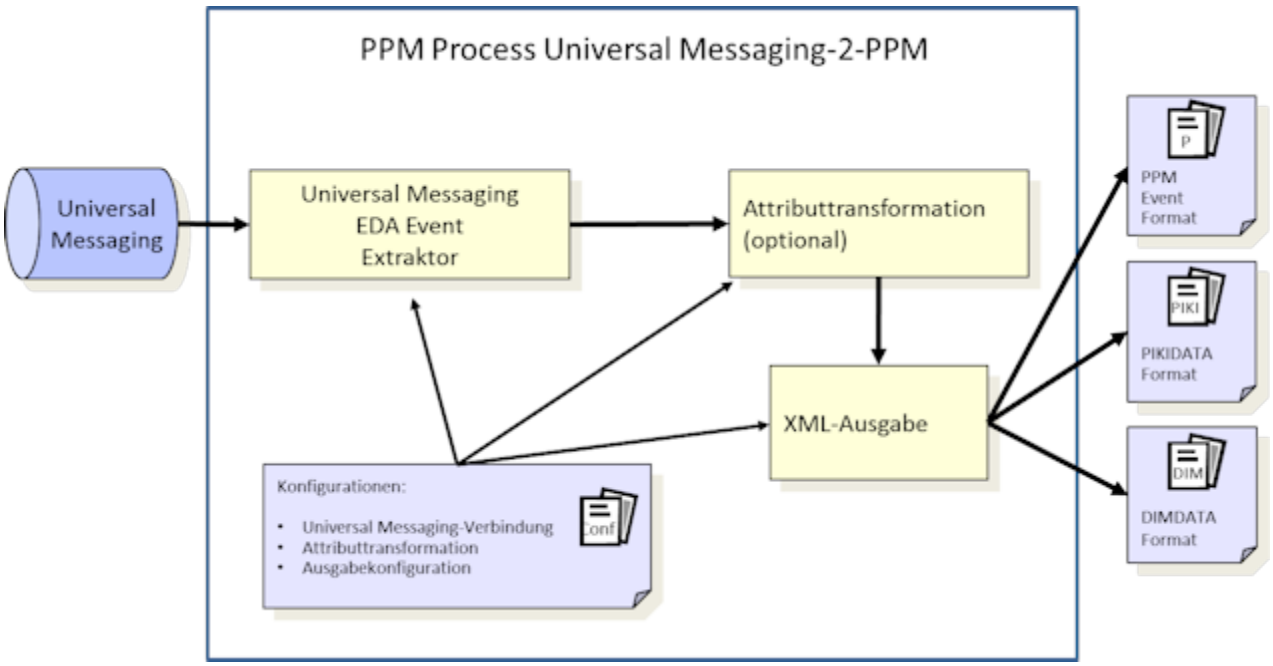
- Wird die Extraktion mit einer Datenquellenliste aufgerufen, die mehrere passende Datenquellen enthält und tritt bei der Extraktion einer dieser Datenquellen ein Fehler auf, der zum Abbruch dieses Vorgangs führt, so wird die Verarbeitung mit der nächsten Datenquelle fortgeführt, d. h. der Abbruch bei einer einzelnen Datenquelle führt nicht zu einem Komplettabbruch.
- Tritt bei mindestens einer Datenquelle ein Fehler auf, der bei der Einzel-Extraktion dieser Datenquelle zu einem Exit-Fehlerstatus geführt hätte, so liefert die Gesamtverarbeitung der Datenquellenliste den letzten dieser Exit-Fehlerstatus zurück.

9 PPM Process Extractor Universal Messaging-2-PPM

Dieses Kapitel gibt einen Überblick über die Architektur, Funktionsweise und Konfiguration von PPM Process Extractor Universal Messaging-2-PPM.

9.1 Architektur

Die folgende Abbildung veranschaulicht die Funktionalität von PPM Process Extractor Universal Messaging-2-PPM:



Mit den Verbindungsdaten der Universal Messaging-Systemkonfiguration wird eine Verbindung zum entsprechenden Channel des Universal Messaging-Systems hergestellt. Anschließend werden EDA-Events, die eine ID größer gleich der angegebenen Start-ID haben, ausgelesen und in ein PPM kompatibles System-Event-Format umgewandelt. Anschließend erfolgt eine optionale Attributtransformation und die Ausgabe in XML-Dateien eines PPM-konformen Ausgabeformats.

Es werden nur unverschlüsselte Verbindungen zu Universal Messaging unterstützt.

9.2 Universal Messaging-Systemkonfiguration

Die Verbindungsdaten zum Universal Messaging-System werden bei Verwendung einer Datenquelle in einer XML-Datei angegeben.

Das Format der XML-Datei ist durch folgende DTD **nirvanaconfig.dtd** vorgegeben:

```
<!ELEMENT nirvanasystemconf (nirvanasystem*)>
<!ELEMENT nirvanasystem (realmname, channel)>
<!ATTLIST nirvanasystem
    name ID #REQUIRED
    timeout CDATA "10"
>
<!ELEMENT realmname (#PCDATA)>
<!ELEMENT channel (#PCDATA)>
```

XML-Bezeichner	Beschreibung	Beispiel
Realmname	Eindeutige URL des Universal Messaging-Realms („nsp://<ip>:<port>")	nsp://172.20.210.21:9000
Channel	Bezeichnung des Universal Messaging-Channels	Event/WebM/Process/V1_0/ProcessInstanceChange
name	Eindeutiger Name des Universal Messaging-Systems	NVTestsystem
timeout	Zeitdauer (in Sekunden), nach der die Verbindung abgebrochen wird, wenn keine Daten übertragen werden. Vorgabewert: 10	15

Beispiel

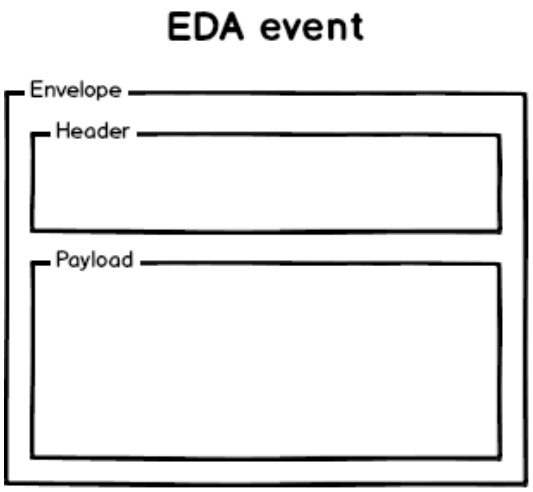
```
...
<nirvanasystem name = "NVTestsystem" timeout = "15">
    <realmname>nsp://172.20.210.21:9000</realmname>
    <channel>Event/WebM/Process/V1_0/ProcessStepInstanceChange</channel>
</nirvanasystem>
...
```

9.3 Konvertierung von Universal Messaging-EDA-Events

In folgenden Kapiteln wird beschrieben, wie Universal Messaging-EDA-Events in PPM-Events konvertiert werden.

9.3.1 Universal Messaging-EDA-Events

Jedes EDA-Event besteht aus einem Kopf (Header) und Nutzdaten (Payload):



Kopf und Nutzdaten enthalten Datenattribute mit Werten. Die Datenattribute im Kopf sind bei allen EDA-Events die gleichen. Die Datenattribute der Nutzdaten sind abhängig vom Typ des EDA-Event.

9.3.2 Konvertierung in ein PPM-Event

9.3.2.1 Kopfdaten

Der Kopf eines EDA-Events enthält eine Liste von Eigenschaften mit jeweils einem Schlüssel und einem Wert.

Bei den Schlüsseln wird das Zeichen **$** als Trenner für Teile des Schlüsselnamens verwendet, wie z. B. „$Event$Start". NV2PPM entfernt das erste $-Zeichen ersatzlos, das am Anfang des Schlüsselnamens steht. Alle anderen $-Zeichen werden durch einen Unterstrich _ ersetzt.

Beispiel

Universal Messaging-Event: $Event$Start

PPM-Event: Event_Start

Schlüssel eines Universal Messaging-Events werden in PPM-System-Events zu Attributtypen:

```
<attribute type="Event_Start"></attribute>
```

Die Werte der Eigenschaften des Universal Messaging-Events werden zu Werten von PPM-System-Event-Attributen.

Die im Kopf enthaltenen Eigenschaften **$Event$Start** und **$Event$End** werden speziell behandelt. Jede dieser Eigenschaften enthält einen sekundengenauen Zeitstempel als Wert. NV2PPM konvertiert diese Zeitstempel in einen Wert der Form **yyyy-MM-ddTHH:mm:ss.SSSZ**.

Beispiel

```
<attribute type="Event_Start">2013-01-23T13:19:41.444+0100</attribute>
```

Da der PPM-Datentyp **TIME** keine Millisekunden unterstützt, gehen die Millisekunden beim Import in PPM verloren. Da es aber manchmal nützlich ist den Millisekundenwert auch in PPM verwenden zu können (z. B. zur Verwendung bei SortMerge), erzeugt PPM für die beiden Universal Messaging-Eigenschaften jeweils ein weiteres System-Event-Attribut mit den Millisekunden als Wert. Diese Attribute haben den System-Event-Attributtyp **Event_Start_MilliSecond** bzw. **Event_End_MilliSecond**.

Beispiel

```
<attribute type="Event_Start">2013-01-23T13:19:41.444+0100</attribute>
<attribute type="Event_Start_MilliSeconds">444</attribute>
```

Hier ein komplettes Beispiel von konvertieren Eigenschaften eines EDA-Event-Kopfs:

```
<attribute type="EventID">83434</attribute>
<attributetype="Event_EventID">cb79c5aa-2304-4b98-a7e0-bf9bc6a07e8c</attribute>
<attribute type="Event_FormatVersion">9.0</attribute>
<attribute type="Event_Start">2013-01-23T13:19:41.444+0100</attribute>
<attribute type="Event_Start_MilliSeconds">444</attribute>
<attribute
type="Event_Type">{http://namespaces.softwareag.com/EDA/WebM/Process/1.0}Process
InstanceChange</attribute>
<attribute
type="breadcrumbId">ID-sbrvppmop03-54594-1358167091138-0-1193</attribute>
```

9.3.2.2 Nutzdaten

Die Nutzdaten enthalten ein XML-Dokument basierend auf einem vordefinierten Schema.

Beispiel (EDA-Event Nutzdaten)

```
<ns:ProcessInstanceChange
xmlns:ns="http://namespaces.softwareag.com/EDA/WebM/Process/1.0">
    <ns:ProcessModel>
     <ns:ID>PEBVT_Subprocess/ANDJoinSubprocess</ns:ID>
     <ns:DisplayName>ANDJoinSubprocess</ns:DisplayName>
     <ns:Version>1</ns:Version>
    </ns:ProcessModel>
    <ns:ProcessInstance>
     <ns:ID>29eb1ad0-6557-11e2-badd-8ea453f9c825</ns:ID>
     <ns:Iteration>1</ns:Iteration>
    </ns:ProcessInstance>
    <ns:Status>Started</ns:Status>
</ns:ProcessInstanceChange>
```

In einem PPM-System-Event werden die XML-Elemente zu Attributtypen, wobei die Hierarchie der XML-Elemente des EDA-Event in eine flache Struktur transformiert wird. Der PPM-Attributtyp wird aus den hierarchischen XML-Elementnamen getrennt und durch einen Unterstrich zusammengesetzt.

Beispiel

EDA-Event:

```
<ns:ProcessInstanceChange>
    <ns:ProcessModel>
        <ns:ID>PEBVT_Subprocess/ANDJoinSubprocess</ns:ID>
…
    </ns:ProcessModel>
…
</ns:ProcessInstanceChange>
```

PPM-System-Event:

```
<attribute
type="ProcessInstanceChange_ProcessModel_ID">PEBVT_Subprocess/ANDJoinSubprocess<
/attribute>
```

Die am Anfang dieses Unterkapitel aufgeführten Nutzdaten eines EDA-Event werden somit in folgende PPM-System-Event-Attribute umgewandelt:

Beispiel (PPM-Systemeventattribute)

```
<attribute
type="ProcessInstanceChange_ProcessInstance_ID">29eb1ad0-6557-11e2-badd-8ea453f9
c825</attribute>
<attribute type="ProcessInstanceChange_ProcessInstance_Iteration">1</attribute>
<attribute
type="ProcessInstanceChange_ProcessModel_DisplayName">ANDJoinSubprocess</attribu
te>
<attribute
type="ProcessInstanceChange_ProcessModel_ID">PEBVT_Subprocess/ANDJoinSubprocess<
/attribute>
<attribute type="ProcessInstanceChange_ProcessModel_Version">1</attribute>
<attribute type="ProcessInstanceChange_Status">Started</attribute>
<attribute
type="breadcrumbId">ID-sbrvppmop03-54594-1358167091138-0-1193</attribute>
```

9.3.3 Konvertierung der Namespaces in PSICs

Für das EDA-Event vom Typ **ProcessStepInstanceChange** (PSIC) werden die Namespaces, die für die Elemente auf der ersten Ebene unterhalb ProcessStepInstanceChange_BusinessData verwendet werden, in geschweiften Klammern als Präfix für das XML-Element im Attribut-Type in der Ausgabedatei verwendet.

Beispiel

```
<ns:ProcessStepInstanceChange>
  …
  <ns:BusinessData>
    <p:element3 xmlns:p="http://some.name/space1">
      <p:element4>some_value</p:element4>
    </p:element3>
    …
  </ns::BusinessData>
  …
</ns:ProcessStepInstanceChange>
```

Der Namespace wird transformiert in

```
<attribute
type="ProcessStepInstanceChange_BusinessData_{http://some.name/space1}element3_e
lement4">some_value</attribute>
```

Alle anderen Namespaces werden ignoriert. Das Attribut könnte auch von folgendem Namespace abgeleitet werden.

```
<ns:ProcessStepInstanceChange xmlns="http://some.name/space_root">
  …
  <ns:BusinessData>
    <p:element3 xmlns:p="http://some.name/space1">
      <t:element4 xmlns:t="http://some.name/space3">some_value</t:element4>
    </p:element3>
    …
  </ns::BusinessData>
  …
</ns:ProcessStepInstanceChange>
```

9.4 Kommandozeilenprogramm

Die XML-Ausgabedatei(en) erzeugen Sie mit dem Kommandozeilenprogramm **runnv2ppm.bat**.

Der Aufruf des Programmes ohne Parameter oder mit **-h** oder **-?** gibt die Hilfe auf der Konsole aus, in der alle verfügbaren Optionen beschrieben sind.

9.4.1 Argumente des Kommandozeilenprogramms

9.4.1.1 Allgemeine Argumente

-VERSION

Gibt die Versionsnummer des verwendeten PPM-Prozessextraktors aus. Andere angegebene Argumente werden ignoriert.

-INFORMATION YES|NO

Hier legen Sie fest, ob Informationen während des Imports ausgegeben werden sollen (yes) oder nicht (no). Voreingestellt ist **yes**.

-WARNING YES|NO

Hier legen Sie fest, ob Warnmeldungen während des Imports ausgegeben werden sollen (yes) oder nicht (no). Voreingestellt ist **yes**.

-ERROR YES|NO

Hier legen Sie fest, ob Fehlermeldungen während des Imports ausgegeben werden sollen (yes) oder nicht (no).Voreingestellt ist **yes**.

-PROTOCOLFILE <FILENAME>

Hier können Sie eine Protokolldatei festlegen, in die alle Meldungen während des Imports geschrieben werden. Wenn Sie eine Datei festlegen, werden am Bildschirm nur noch kritische Fehlermeldungen ausgegeben, die zum Programmabbruch führen.

-LANGUAGE <ISO-CODE>

Hier geben Sie die Sprache an, in der die Protokollinformationen ausgegeben werden sollen.

9.4.1.2 Quellsystemspezifische Argumente

-DATASOURCE <FILENAME> (OPTIONAL)

Hier geben Sie die Datenquelle an, die für den Auslesevorgang verwendet werden soll. Die XML-Datei enthält Angaben zu allen für den Extraktionsvorgang und die XML-Ausgabe zu verwendenden Dateien. Verwenden Sie diesen Parameter nicht zusammen mit einem der Parameter **-datasourcelist**, **-systemconfig**, **-tableconfig**, **-calcconfig**, **-outfile** bzw. **-nozip**, sondern anstelle dieser Parameter.

-DATASOURCELIST <FILENAME>

Mit Hilfe des Arguments **-datasourcelist** können mehrere Datenquellen auf einmal extrahieren werden. Dies entspricht dem mehrfachen, nacheinander Ausführen der Extraktion mittels des Arguments **-datasource**. Es werden nur Universal Messaging-Datenquellen extrahiert.

Sieh auch Kapitel Extraktion mehrere Datenquellen (Seite 144).

-NIRVANAREALM <UNIVERSAL MESSAGING REALM>

Hier geben Sie die URL inklusive Port-Nummer an, mit der eine Verbindung zum Universal Messaging-Realm-Server hergestellt werden kann. Es werden nur unverschlüsselte Verbindungen unterstützt.

-CHANNEL <UNIVERSAL MESSAGING CHANNEL>

Hier geben Sie den Namen des Universal Messaging-Channel an, aus dem EDA-Events extrahiert werden sollen.

-STARTID <NUMBER> (OPTIONAL)

Hier geben Sie eine Universal Messaging-EDA-Event-ID, d. h. eine Zahl größer oder gleich 0, an. Alle Universal Messaging-EDA-Events mit dieser oder einer größeren ID werden extrahiert.

-TIMEOUT <NUMBER> (OPTIONAL)

Hier geben Sie eine Zahl an. Sie gibt die Zeit in Sekunden an, die gewartet wird, bis ein EDA-Event vom Universal Messaging-Server übertragen wird. Wird innerhalb dieser Zeit kein EDA-Event übertragen, weil z. B. kein passendes EDA-Event in dem betreffenden Universal Messaging-Channel vorhanden ist oder weil die Netzwerkverbindung zu langsam ist, so wird die Datenextraktion abgebrochen.

9.4.1.3 Ausgabedateispezifische Argumente

-OUTFILE <FILENAME>

Mit diesem Parameter geben Sie den Namen der XML-Ausgabedatei an. Die Dateinamenerweiterung wird automatisch ergänzt. Standardmäßig wird die Datei als ZIP-Datei ausgegeben.

-OUTFILEENCODING <ENCODING> (OPTIONAL)

Mit diesem Parameter geben Sie das Encoding der XML-Ausgabedatei an. Vorgabewert: UTF-8

-NOZIP (OPTIONAL)

Mit diesem Parameter geben Sie an, dass die Ausgabedatei nicht als ZIP-Datei, sondern als XML-Datei ausgegeben wird.

-PIKIDATAMAPPING <FILENAME> <PCNAME> (OPTIONAL)

Mit diesem Parameter geben Sie im ersten Argument den Namen der XML-Datei an, die das Mapping enthält. Im zweiten Argument geben Sie den Namen der Kennzahlenreihe an, in der die prozessinstanzunabhängigen Kennzahlenreihen gespeichert werden sollen.

-DIMDATAMAPPING <FILENAME> <DIMNAME> (OPTIONAL)

Mit diesem Parameter geben Sie im ersten Argument den Namen der XML-Datei an, die das Mapping enthält. Im zweiten Argument geben Sie den Namen der Dimension an, für die die extrahierten Werte eingelesen werden sollen.

-SORTEVENTATTRIBUTES (OPTIONAL)

Mit diesem Parameter sortieren Sie die Quellsystemattribute in den System-Events anhand des Attributtyps in alphanumerischer Reihenfolge. Bei großen Datenmengen kann sich dieser Parameter negativ auf die Geschwindigkeit des Auslesevorgangs auswirken.

9.4.1.4 Fortlaufendes automatisiertes Auslesen

Sie können Daten lückenlos automatisiert auslesen, indem Sie den Parameter **-startid** auf der Kommandozeile weglassen und eine Datenquelle (-datasource) verwenden. Die nächste zu lesende EDA-Event-ID wird in der verwendeten Konfigurationsdatei gespeichert bzw. aktualisiert.

Verwenden Sie für das fortlaufende automatisierte Auslesen immer den gleichen Kommandozeilenaufruf, z. B.:
runnv2ppm -datasource datasource.xml.

9.5 Extraktion mehrerer Datenquellen

Sie können mit Hilfe einer Datenquellenliste mehrere Datenquellen auf einmal extrahieren. Die Datenquellenliste kann in einer eigenen Konfigurationsdatei angegeben werden.

Jede in CTK neu angelegte Universal Messaging-Datenquelle wird automatisch am Ende der Datenquellenliste des aktuellen Mandanten hinzugefügt. Bei der Extraktion mit der Kommandozeilenoption **-datasourcelist <filename>** werden die Daten dieser Datenquellen

nacheinander extrahiert, so als wäre die Datenextraktion mehrmals mit der Option **-datasource <filename>** aufgerufen worden. Die Reihenfolge, in der die Extraktion der Datenquellen erfolgt, ist in der Konfigurationsdatei angegeben. In der Liste vorhandene Datenquellen eines anderen Typs als **Universal Messaging** werden bei der Extraktion übersprungen.

Für die Fehlerbehandlung gilt Folgendes:

- Wird die Extraktion mit einer gültigen Datenquellenliste aufgerufen, die aber keine passenden Datenquellen enthält, so beendet sie sich ohne eine Fehlermeldung.
- Wird die Extraktion mit einer Datenquellenliste aufgerufen, die mehrere passende Datenquellen enthält und tritt bei der Extraktion einer dieser Datenquellen ein Fehler auf, der zum Abbruch dieses Vorgangs führt, so wird die Verarbeitung mit der nächsten Datenquelle fortgeführt, d. h. der Abbruch bei einer einzelnen Datenquelle führt nicht zu einem Komplettabbruch.
- Tritt bei mindestens einer Datenquelle ein Fehler auf, der bei der Einzel-Extraktion dieser Datenquelle zu einem Exit-Fehlerstatus geführt hätte, so liefert die Gesamtverarbeitung der Datenquellenliste den letzten dieser Exit-Fehlerstatus zurück.

10 Funktionalitäten aller Extraktoren

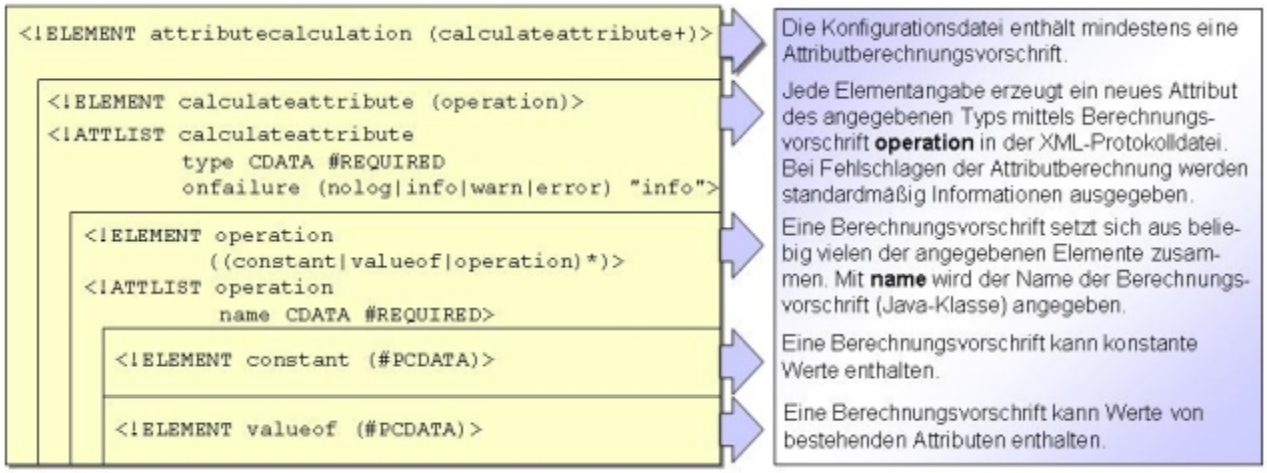
In diesem Kapitel sind die Programmeigenschaften beschrieben, die alle Prozessextraktoren gemein haben.

10.1 Attributtransformation

Wenn Sie Werte von bestimmten Attributtypen in den Ausgabedateien ändern oder zusätzliche Attributtypen erzeugen möchten, müssen Sie in einer XML-Konfigurationsdatei entsprechende Angaben machen. Der Name dieser XML-Konfigurationsdatei wird dem Kommandozeilenprogramm als Argument übergeben.

Diese Konfigurationsdatei enthält die Einstellungen zur Bearbeitung existierender Attributtypen sowie zur Erzeugung und Transformation neuer Attributtypen beim Erstellen der XML-Ausgabedatei(en).

Das Format der XML-Datei ist durch folgende DTD vorgegeben:



XML-Element bzw. XML-Attribut	Beschreibung
attributecalculation	Konfiguration der Attributtransformation
calculateattribute	Erzeugung eines System-Event-Attributtyps bei erfolgreicher Ausführung der Transformations-vorschrift (**operation**)
type	Name des erzeugten System-Event-Attributtyps

XML-Element bzw. XML-Attribut	Beschreibung
onfailure	Abbruchverhalten bei Transformationsfehlschlag. Mögliche Werte: **nolog** (keine Protokollausgabe) **info** (Ausgabe als Information) **warn** (Ausgabe als Warnung) **error** (Ausgabe als Fehlermeldung) Standardwert: **info**
operation	Transformationsvorschrift, die durch eine Java-Klasse implementiert wird. Jede Operation hat genau einen Rückgabewert in Form einer Zeichenfolge. Schlägt die Operation fehl, wird nichts zurückgegeben.
name	Name der Operation
constant	Konstanter Wert, der bei der Transformation verwendet werden soll
valueof	Wert eines vorhandenen bzw. eines bereits transformierten Quellsystemattributtyps

Die Elemente **calculateattribute** werden in der Reihenfolge abgearbeitet, in der sie in der Konfigurationsdatei angegeben sind. Verschachtelungen der XML-Elemente **operation** werden von innen nach außen bzw. oben nach unten bearbeitet.

Aus Effizienzgründen werden während einer Attributtransformation nur die Argumente ausgewertet, die für die Transformation unerlässlich sind. Beispielsweise wird in der Operation **if_then_else** nach Prüfung des ersten Arguments auf Wahrheit entweder nur das zweite Argument (then) oder das dritte Argument (else) abgearbeitet. Die Attributtransformation wird mit einer Fehlermeldung abgebrochen, sobald ein benötigtes Element keinen Wert zurückgeben kann.

Wenn das XML-Attribut **onfailure** in der Attributtransformationsdatei den Wert **error** hat, wird bei nicht erfolgreicher Attributerzeugung in der XML-Ausgabedatei das entsprechende System-Event mit dem Attribut **EVENT_HAS_ATTRIBUTE_ERRORS=true** versehen. Zusätzlich wird in der Protokolldatei eine Fehlermeldung erzeugt.

Wenn das XML-Attribut **onfailure** den Wert **warn** hat, wird bei nicht erfolgreicher Attributerzeugung in der XML-Ausgabedatei das entsprechende System-Event mit dem Attribut **EVENT_HAS_ATTRIBUTE_WARNINGS=true** versehen. Zusätzlich wird in der Protokolldatei eine Warnung erzeugt.

Warnung

Mit **calculateattribute** werden bei erfolgreicher Ausführung der entsprechenden Transformationsvorschriften (**operation**) Werte bereits vorhandener Attributtypen überschrieben. Es wird keine Warnung ausgegeben. Stellen Sie gegebenenfalls sicher, dass die Werte bereits vorhandener Attributtypen nicht versehentlich überschrieben werden, indem Sie mit einer geeigneten Transformationsvorschrift eine bedingte Existenzprüfung durchführen. Dazu können Sie die Operation **exists** (siehe Kapitel **Operationen** (Seite 149)) verwenden.

Wenn die komplette Transformation eines Attributs abgebrochen wird, bleibt der bestehende Attributwert unverändert. Abhängig von Ihren Angaben zur Option **onfailure** wird eine entsprechende Meldung mit einem Verweis auf das erzeugte System-Event ausgegeben. Bei **nolog** erfolgt keine Protokollausgabe.

Beispiel

Der Auslesevorgang von Quellsystemdaten aus dem R/3-System und die anschließende Bearbeitung durch den XML-Generator haben folgenden Eintrag in der XML-Ausgabedatei **event.xml** erzeugt:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="VBAP-VBELN">3866</attribute>
    <attribute type="VBAP-POSNR">20</attribute>
    <attribute type="VBAP-ERNAM">SCHMIDT</attribute>
    <attribute type="VBAP-MATNR-MAKTX">100038</attribute>
    <attribute type="VBAP-KWMENG">3</attribute>
  </event>
</eventlist>
```

Durch folgende Angaben in der Attributtransformationsdatei **attributetransform.xml** können Sie die Wertdarstellung des Attributtyps **VBAP-ERNAM** so abändern, dass als neuer Wert die Konkatenation des ursprünglichen Werts mit einer konstanten Zeichenfolge verwendet wird:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE attributecalculation SYSTEM
         "attributetransformation.dtd">
<attributecalculation>
  <calculateattribute type="VBAP-ERNAM">
  <operation name="concat">
  <constant>Walter, "A" </constant>
  <valueof>VBAP-ERNAM</valueof>
  </operation>
  </calculateattribute>
  Ermittle die neuen bzw. geänderten MM-Belege ...
</attributecalculation>
```

Folgender Kommandozeilenaufruf erzeugt aus der ursprünglichen XML-Ausgabedatei unter Verwendung der o. g. Attributtransformationsdatei die erwünschte XML-Ausgabedatei:

```
runjdbc2ppm -systemconfig SysConfig.xml ides_doc_vdap -tableconfig TableConf.xml
SD_C -begindate 01.01.2005 -enddate 31.01.2005 -calcconfig attributetransform.xml
-outfile event -outfileencoding ISO-8859-1 -nozip
```

Der gleiche Eintrag wie oben sieht dann folgendermaßen aus:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
```

```
  <event>
    <attribute type="VBAP-VBELN">3866</attribute>
    <attribute type="VBAP-POSNR">20</attribute>
    <attribute type="VBAP-ERNAM">Walter, "A" SCHMIDT
    </attribute>
    <attribute type="VBAP-MATNR-MAKTX">100038</attribute>
    <attribute type="VBAP-KWMENG">3</attribute>
  </event>
</eventlist>
```

10.1.1 Operationen

Die PPM Prozessextraktoren stellen die in den folgenden Kapiteln aufgeführten Operationen für die Transformation bzw. Änderung von Attributtypen zur Verfügung.

Bei den gezeigten Syntaxbeispielen für das XML-Element **operation** werden der Einfachheit halber ausschließlich Konstanten verwendet. In einer Operation können jedoch auch die Elemente **valueof** und **operation** angegeben werden.

Kann mit einer Transformationsvorschrift kein Wert ermittelt werden, wird die Transformation des Attributtyps nur dann abgebrochen, wenn der nicht ermittelte Wert für die weitere Transformation unerlässlich ist. Abhängig von der Angabe zur Option **onfailure** wird in diesem Fall gegebenenfalls eine entsprechende Meldung ausgegeben.

Der Rückgabewert der Operationen ist generell eine alphanumerische Zeichenkette.

10.1.1.1 unmask

Beschreibung:	Entfernung der Maskierung eines Wertes
Syntax:	Zwei Parameter: 1. Maskierungszeichen 2. Wert, der demaskiert werden soll

Beispiel:	`<operation name="unmask">`
	`  <constant>\</constant>`
	`  <constant>\C:\\ppm3\\ppm320\\build3714\\dtd\`
	`  </constant>`
	`</operation>`

Rückgabewert:	**C:\ppm3\ppm320\build3714\dtd** Kann für einen der Parameter kein Wert ermittelt werden, wird nichts zurückgegeben.

Der zu demaskierende Wert muss mit dem angegebenen Maskierungszeichen beginnen und enden.

10.1.1.2 trim

Beschreibung:	Entfernung der Leerzeichen an Anfang und Ende eines Wertes
Syntax:	Ein Parameter: Wert, von dessen Anfang und Ende Leerzeichen entfernt werden sollen

Beispiel:	`<operation name="trim">`
	`  <constant> PPM </constant>`
	`</operation>`

Rückgabewert:	`<...>PPM<...>` Kann für einen der Parameter kein Wert ermittelt werden, wird nichts zurückgegeben.
Anmerkung:	Die Anzahl der Leerzeichen an Anfang und Ende des Feldwertes spielt keine Rolle.

10.1.1.3 concat

Beschreibung:	Konkatenation beliebig vieler Werte
Syntax:	Ein bis n Parameter

Beispiel:	`<operation name="concat">`
	`  <constant>Auftragsabwicklung</constant>`
	`  <constant>/</constant>`
	`  <constant>Barverkauf</constant>`
	`</operation>`

Rückgabewert:	**Auftragsabwicklung/Barverkauf** Kann für einen der Parameter kein Wert ermittelt werden, wird nichts zurückgegeben.

10.1.1.4 substring

Beschreibung:	Ermittlung einer Teilzeichenkette aus einem Wert
Syntax:	Zwei bis drei Parameter: 1. Wert (Zeichenkette), aus dem eine Teilzeichenkette ermittelt werden soll 2. Startposition: 1 = erstes Zeichen von links -1 = erstes Zeichen von rechts 3. Anzahl der auszulesenden Zeichen (optional). Fehlt diese Angabe, wird eine Teilzeichenkette von der angegebenen Startposition bis zum Ende des Werts extrahiert.

Beispiel:	`<operation name="substring">`
	`<constant>Order processing/Cash sales`
	`</constant>`
	`<constant>-1</constant>`
	`<constant>10</constant>`
	`</operation>`

Rückgabewert:	**Cash sales** Kann für einen der Parameter kein Wert ermittelt werden bzw. werden für die Startposition und/oder Zeichenanzahl keine Ganzzahlen angegeben, wird nichts zurückgegeben.

10.1.1.5 if_then_else

Beschreibung:	Bedingte Attributtransformation
Syntax:	Zwei bis drei Parameter: 1. Wert, der auf Gleichheit geprüft wird 2. Wert, der bei Gleichheit zurückgeliefert wird 3. Wert, der bei Ungleichheit zurückgeliefert wird

Beispiel:	Wert, der auf Gleichheit mit der Zeichenfolge **true** geprüft wird:
	`<operation name="if_then_else">`
	`  <constant>not true</constant>`
	`  <constant>Bedingung ist erfüllt</constant>`
	`  <constant>Bedingung ist nicht erfüllt`
	`  </constant>`
	`</operation>`

Rückgabewert:	**Bedingung ist nicht erfüllt** Wenn der erste Wert gleich der Zeichenkette **true** ist (keine Unterscheidung von Groß-/Kleinschreibung), wird der zweite Wert zurückgeliefert, andernfalls (falls vorhanden) der dritte Wert. Kann für den ersten Parameter (Bedingung) oder den dritten Parameter bei nicht erfüllter Bedingung kein Wert ermittelt werden, wird nichts zurückgegeben.

10.1.1.6 set_with_default

Beschreibung:	Setzen eines Wertes mit einem Vorgabewert
Syntax:	Zwei Parameter: 1. Wert, der zurückgeliefert werden soll, wenn er existiert 2. Wert, der zurückgeliefert wird, wenn für den ersten Parameter kein Wert ermittelt werden kann.

Beispiel:	`<operation name="set_with_default">`
	`  <constant>42</constant>`
	`  <constant>1</constant>`
	`</operation>`

Rückgabewert:	**42** Kann für beide Parameter kein Wert ermittelt werden, wird nichts zurückgegeben.

10.1.1.7 integer_plus

Beschreibung:	Addition ganzzahliger Parameter
Syntax:	Zwei bis n Parameter: zu addierende Werte

Beispiel:	`<operation name="integer_plus">`
	`  <constant>1</constant>`
	`  <constant>4</constant>`
	`  <constant>37</constant>`
	`</operation>`

Rückgabewert:	**42** Kann für einen der Parameter kein Wert ermittelt werden bzw. hat einer der Parameter einen nicht ganzzahligen Wert, wird nichts zurückgegeben.

10.1.1.8 integer_minus

Beschreibung:	Subtrahieren ganzzahliger Parameter von einem ganzzahligen Parameter
Syntax:	Zwei bis n Parameter: 1. Wert, von dem die übrigen Werte subtrahiert werden sollen 2. Wert, der subtrahiert werden soll ... " n. Wert, der subtrahiert werden soll

Beispiel:	`<operation name="integer_minus">`
	`  <constant>1</constant>`
	`  <constant>4</constant>`
	`  <constant>37</constant>`
	`</operation>`

Rückgabewert:	**-40** Kann für einen der Parameter kein Wert ermittelt werden bzw. hat einer der Parameter einen nicht ganzzahligen Wert, wird nichts zurückgegeben.

10.1.1.9 string_equal

Beschreibung:	Vergleich von Zeichenketten: Liefert die Zeichenkette **true** zurück, wenn alle Werte gleich sind, ansonsten die Zeichenkette **false**.
Syntax:	Zwei bis n Parameter: zu vergleichende Werte

Beispiel:	`<operation name="string_equal">`
	`  <constant>42</constant>`
	`  <constant>42</constant>`
	`  <constant>37</constant>`
	`</operation>`

Rückgabewert:	**false** Sobald ein Wert gefunden wird, der nicht mit dem ersten überprüften Wert übereinstimmt, wird **false** zurückgeliefert, auch wenn noch andere Werte folgen sollten. Kann für einen der überprüften Parameter kein Wert ermittelt werden, wird nichts zurückgegeben.

10.1.1.10 string_in

Beschreibung:	Vergleicht eine Zeichenkette mit einer Auflistung von Zeichenketten. Liefert die Zeichenkette **true** zurück, wenn die Zeichenkette mit mindestens einer der anderen Zeichenketten identisch ist, ansonsten die Zeichenkette **false**.
Syntax:	Drei bis n Parameter: zu vergleichende Werte

Beispiel:	`<operation name="string_in">`
	`  <constant>42</constant>`
	`  <constant>42</constant>`
	`  <constant>37</constant>`
	`  <constant>42</constant>`
	`</operation>`

Rückgabewert:	**true** Sobald eine Zeichenkette gefunden wird, die mit der zu vergleichenden Zeichenkette identisch ist, wird **true** zurückgeliefert, auch wenn noch andere Werte folgen sollten. Wenn für den ersten Parameter kein Wert ermittelt werden kann, wird nichts zurückgegeben.

10.1.1.11 boolean_not

Beschreibung:	Liefert die Zeichenkette **false** zurück, falls der Parameter gleich **true** ist (ohne Berücksichtigung der Groß-/ Kleinschreibung), ansonsten die Zeichenkette **true**.
Syntax:	Ein Parameter: Boolescher Wert, der negiert werden soll

Beispiel:	`<operation name="boolean_not">`
	`..<constant>this is not true</constant>`
	`</operation>`

Rückgabewert:	**true** Kann für den Parameter kein Wert ermittelt werden, wird nichts zurückgegeben.

10.1.1.12 boolean_and

Beschreibung:	Liefert die Zeichenkette **true** zurück, falls alle Parameter gleich **true** sind (ohne Berücksichtigung der Groß-/ Kleinschreibung), ansonsten die Zeichenkette **false**.
Syntax:	Zwei bis n Parameter: Boolesche Werte, die mit dem booleschen AND verknüpft werden sollen

Beispiel:	`<operation name="boolean_and">`
	`..<constant>truE</constant>`
	`..<constant>True</constant>`
	`..<constant>TRUE</constant>`
	`</operation>`

Rückgabewert:	**true** Sobald ein Wert gefunden wird, der nicht **true** ist, wird **false** zurückgeliefert. Eine weitere Überprüfung nachfolgender Werte findet nicht statt. Kann für einen der überprüften Parameter kein Wert ermittelt werden, wird nichts zurückgegeben.

10.1.1.13 boolean_or

Beschreibung:	Liefert die Zeichenkette **true** zurück, falls einer der Parameter gleich **true** ist (ohne Berücksichtigung der Groß-/Kleinschreibung), ansonsten die Zeichenkette **false**.
Syntax:	Zwei bis n Parameter: Boolesche Werte, die mit dem booleschen OR verknüpft werden sollen

Beispiel:	`<operation name="boolean_or">`
	`  <constant>false</constant>`
	`  <constant>True</constant>`
	`  <constant>this is not true</constant>`
	`</operation>`

Rückgabewert:	**true** Sobald ein Wert gefunden wird, der **true** ist, wird direkt **true** zurückgeliefert. Eine weitere Überprüfung nachfolgender Werte findet nicht statt. Kann für einen der überprüften Parameter kein Wert ermittelt werden, wird nichts zurückgegeben.

10.1.1.14 exists

Beschreibung:	Existenzprüfung
Syntax:	Ein bis n Parameter: Werte, die auf Existenz geprüft werden sollen

Beispiel:	`<operation name="exists">`
	`  <constant>false</constant>`
	`  <constant>Order volume</constant>`
	`  <constant>Turnover</constant>`
	`</operation>`

Rückgabewert:	**true** Liefert **true** zurück, falls für alle Parameter ein Wert ermittelt werden konnte, ansonsten **false**.

10.1.1.15 get_null

Beschreibung:	Liefert nichts zurück. Falls die äußerste Operation einer Transformationsvorschrift nichts zurückliefert, wird der Attributtyp nicht in die XML-Ausgabedatei übernommen.
Syntax:	Keine Parameter
Beispiel:	`<operation name="get_null"/>`
Rückgabewert:	Keiner

10.1.1.16 getCurrentTimestamp

Beschreibung:	Liefert einen aktuellen Zeitstempel zurück. Das Format wird im ersten Parameter angegeben. In Frage kommen alle durch die Klasse **java.text.SimpleDateFormat** vorgegebenen Datums- und Uhrzeitformate (analog zum condition-Operator **creationtimestamp**), wie sie in der Technischen Referenz **Datenimport** im Abschnitt **Transformationen** beschrieben sind. Wenn Sie nur einen Parameter angeben, wird als entsprechender Attributwert in allen System-Events der Ausgabedatei(en) derselbe Zeitstempel erzeugt, der dem ersten Aufruf der Operation entspricht. Wenn Sie zwei Parameter angeben, wird an jedes System-Event der jeweilige Zeitpunkt des Aufrufs der Operation als Zeitstempelattribut geschrieben, d. h. die Werte können je nach gewähltem Format und Auslesedauer unterschiedlich sein. Die Plausibilität des zweiten Parameters wird nicht geprüft, er muss lediglich vorhanden sein. Kann für den ersten Parameter kein Wert ermittelt werden, wird nichts zurückgegeben.
Syntax:	Ein bis zwei Parameter

Beispiel:	`<operation name="getCurrentTimestamp">`
	`  <constant>dd.MM.yyyy HH:mm:ss</constant>`
	`  <constant/>`
	`</operation>`

Rückgabewert:	z. B. **20.07.2006 15:45:08**, ansonsten unterschiedliche Werte für verschiedene System-Events je nach Auslesedauer

10.1.2 Beispiel für die Attributtransformation

Das Einlesen von Quellsystemdaten hat folgenden Eintrag in der XML-Ausgabedatei erzeugt:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="VBAP-VBELN">3866</attribute>
    <attribute type="VBAP-POSNR">20</attribute>
    <attribute type="VBAP-ERNAM">SCHMIDT</attribute>
    <attribute type="VBAP-MATNR-MAKTX">100038</attribute>
    <attribute type="VBAP-KWMENG">3</attribute>
  </event>
</eventlist>
```

Sie möchten den Wert des Attributtyps **VBAP-ERNAM** mit einer konstanten Zeichenkette verknüpfen und in den neuen Attributtyp **Erfasser der Auftragsposition** schreiben. Darüber hinaus möchten Sie den Attributtyp **Bemerkung zur Auftragserfassung** erzeugen, der den Vor- und Nachnamen aus dem Attributtyp **VBAP-ERNAM** extrahiert und zur folgenden Zeichenkette zusammenfügt:

<Vorname>< ><Nachname>< hat Position ><VBAP-POSNR>< des Auftrags ><VBAP-VBELN>< erfasst.>

Die Angaben in spitzen Klammern stehen dabei für Attributwerte, die aus bestehenden Attributtypen oder Konstanten ausgelesen bzw. transformiert werden. Die dazu notwendigen Angaben machen Sie in der XML-Attributtransformationsdatei:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE attributecalculation SYSTEM
                        "attributetransformation.dtd">
<attributecalculation>
  <calculateattribute type="Erfasser der Auftragsposition">
    <operation name="concat">
      <constant>Walter, "A" </constant>
      <valueof>VBAP-ERNAM</valueof>
    </operation>
  </calculateattribute>
  <calculateattribute type="Bemerkung zur Auftragserfassung">
    <operation name="concat">
      <operation name="substring">
        <valueof>Erfasser der Auftragsposition</valueof>
        <constant>1</constant>
        <constant>6</constant>
      </operation>
      <constant> </constant>
      <operation name="substring">
        <valueof>Erfasser der Auftragsposition</valueof>
```

```
      <constant>-1</constant>
      <constant>7</constant>
    </operation>
    <constant> hat Position </constant>
    <valueof>VBAP-POSNR</valueof>
    <constant> des Auftrags </constant>
    <valueof>VBAP-VBELN</valueof>
    <constant> erfasst.</constant>
  </operation>
 </calculateattribute>
</attributecalculation>
```

Nach einem erneuten Aufruf des jeweiligen Kommandozeilenprogramms nun mit dem Parameter **-calcconfig <XML-Attributtransformationsdatei>** wird folgende XML-Ausgabedatei erzeugt:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
 <event>
  <attribute type="VBAP-VBELN">3866</attribute>
  <attribute type="VBAP-POSNR">20</attribute>
  <attribute type="VBAP-ERNAM">SCHMIDT</attribute>
  <attribute type="Erfasser der Auftragsposition">
              Walter, "A" SCHMIDT
  </attribute>
  <attribute type="VBAP-MATNR-MAKTX">100038</attribute>
  <attribute type="VBAP-KWMENG">3</attribute>
  <attribute type="Bemerkung zur Auftragserfassung">
  Walter SCHMIDT hat Position 20 des Auftrags 3866 erfasst.
  </attribute>
 </event>
</eventlist>
```

10.2 Datenquelle

Alle für den Auslesevorgang erforderlichen Konfigurationsdateien sowie die Ausgabedatei können Sie in einer Datenquelldatei zusammen angeben. Die XML-Datei wird mit dem Parameter **-datasource** auf der Konsole angegeben. Die Verwendung des Parameters erfolgt anstelle der Parameter **-systemconfig**, **-tableconfig**, **-csvconfig**, **-i**, **-calcconfig**, **-outfile** und **-nozip**.

Das Format der XML-Datei ist durch die DTD **datasource.dtd** vorgegeben:

XML-Element	XML-Attribut	Beschreibung
datasource	name type lastreaddate lastreadtime	Eine Datenquelle hat einen eindeutigen Namen, ist genau von einem der Typen **EVENT, GRAPH, MYSAP, JDBC** bzw. **CSV** und enthält den letzten Auslesezeitpunkt.
dataextraction (siehe Kapitel Konfiguration mehrerer		Angabe der XML-Ausgabedatei(en). Die Dateiendung (**.zip** bzw. **.xml**) muss explizit angegeben werden.

XML-Element	XML-Attribut	Beschreibung
Ausgabedateien (Seite 161))		Der Dateiendung entsprechend wird entweder eine gezippte oder ungezippte XML-Datei erzeugt.
systemconfig		Angabe der System-konfigurationsdatei. Diese darf nur eine Systemzugangskonfiguration enthalten.
eventspec		Angabe der Tabellen-konfigurationsdatei (Event-Spezifikation). Diese darf nur eine Tabellenkonfiguration (XML-Element **configuration**) enthalten.
attribute transformation (optional)		Angabe der Attributtransformationsdatei

Wenn Sie dem Kommandozeilenprogramm die Datenquelldatei mit dem Parameter **-datasource** übergeben, wird der letzte Auslesezeitpunkt (**lastreaddate**, **lastreadtime**) nur in dieser Datei verwendet und aktualisiert. Eine Anpassung des Zeitpunkts in der verwendeten Systemkonfiguration findet dann nicht mehr statt.

Beispiel

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE datasource SYSTEM "datasource.dtd">
<datasource name="BILLING" type="MYSAP"
      lastreaddate="19700101" lastreadtime="000000">
   <description name="desc_de" language="de">
                              (Storno-)Rechnung,
                            (Storno-)Gutschrift,
                             interne Gutschrift,
                            interne Verrechnung,
                    Lastschrift, Proformarechnung
   </description>
   <description name="desc_en" language="en">
                         (Storno-)Rechnung,
                      (Storno-)Gutschrift,
                           interne Gutschrift,
                            interne Verrechnung,
                    Lastschrift, Proformarechnung
   </description>
   <dataextraction> (siehe Kapitel
      Konfiguration mehrerer Ausgabedateien (Seite 161))
   </dataextraction>
   <systemconfig>C:\Programs\jdbc2ppm\xml\Systemconfig.xml
   </systemconfig>
   <eventspec>C:\Programs\jdbc2ppm\xml\TableConfiguration.xml
   </eventspec>
   <attributetransformation>C:\Programs\jdbc2ppm\xml\
```

```
                        attributecalculation.xml
   </attributetransformation>
</datasource>
```

10.2.1 Konfiguration mehrerer Ausgabedateien

Werden sehr große Datenbestände extrahiert und als System-Events in eine einzige Ausgabedatei geschrieben, wird diese u. U. unhandlich. In solchen Fällen kann bei entsprechender Konfiguration der verwendeten Datenquelle der extrahierte Datenbestand in beliebig viele XML-Ausgabedateien geschrieben werden. Sie brauchen dazu lediglich im XML-Element **dataextraction** Angaben zur maximalen Anzahl von System-Events pro Ausgabedatei und zum Ausgabedateinamen zu machen.

Beispiel (für eine CSV-Datenquelle, für die Typen JDBC und SAP analog)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE datasource SYSTEM "datasource.dtd">
<datasource name="BILLING" type="CSV">
  <dataextraction>
    <outputfilename>..\custom\<clientname>\data\
      BILLING_data_$EXTRACTIONDATE$_$EXTRACTIONTIME$.zip
    </outputfilename>
    <numberofeventsperxmlfile>
      100000
    </numberofeventsperxmlfile>
  </dataextraction>
  ...
  <systemconfig>...</systemconfig>
  <eventspec>...</eventspec>
  ...
</datasource>
```

Für die Datenquelle **BILLING** wird im XML-Element **numberofeventsperxmlfile** festgelegt, dass jede XML-Ausgabedatei 100000 System-Events enthalten soll, wobei die zuletzt erzeugte Ausgabedatei die restliche Anzahl an Events enthält.
Im XML-Element **outputfilename** sind Pfad und Name der Ausgabedateien angegeben. Mittels dieser Angaben werden im angegebenen Ausgabeverzeichnis XML-Dateien mit Namen in der Form **BILLING_data_$EXTRACTIONDATE$_$EXTRACTIONTIME$.zip** erzeugt, wobei die Namensvariable **$EXTRACTIONDATE$** das Extraktionsdatum und **$EXTRACTIONTIME$** die Extraktionszeit enthält. Die erzeugten Ausgabedateien werden mittels **_<x>** am Namensende durchnummeriert, wobei **x** eine fortlaufende Ziffer ist und die zuerst erzeugte Ausgabedatei nicht nummeriert wird.

Wären am 23.06.2007 um 11:36:45 Uhr mit der gezeigten Beispielkonfiguration 369000 System-Events ausgelesen worden, wären folgende Ausgabedateien erzeugt worden:

- **BILLING_data_20070623_113645.zip**
  (mit 100000 System-Events)
- **BILLING_data_20070623_113645_1.zip**
  (mit 100000 System-Events)
- **BILLING_data_20070623_113645_2.zip**
  (mit 100000 System-Events)
- **BILLING_data_20070623_113645_3.zip**
  (mit 69000 System-Events)

In der folgenden Tabelle sind alle Konfigurationsmöglichkeiten aufgelistet:

XML-Element	Beschreibung
numberofeventsperxmlfile	Feste Anzahl an System-Events, die in eine Ausgabedatei geschrieben wird. Die zuletzt erzeugte Ausgabedatei enthält die restliche Anzahl der Events. Fehlt das Element, werden alle System-Events in eine Ausgabedatei geschrieben.
outputfilename	Pfad und Namensmuster der Ausgabedateien. Es werden die Formate XML und ZIP unterstützt. Eine ZIP-Ausgabedatei enthält eine gleichnamige XML-Ausgabedatei. Die Ausgabedateien eines Extraktionsvorganges werden mittels _**<x>** am Namensende fortlaufend nummeriert, wobei die zuerst erzeugte Datei nicht nummeriert wird.

Folgende Variablen sind im Ausgabedateinamen (**outputfilename**) erlaubt und untereinander frei kombinierbar:

Variable (Datenquellentyp)	Beschreibung
$EXTRACTIONDATE$ (CSV, SAP, JDBC)	Extraktionsdatum (Format: yyyyMMdd)
$EXTRACTIONTIME$ (CSV, SAP, JDBC)	Extraktionszeit (Format: HHmmss)
$BEGINDATE$ (SAP, JDBC)	Startdatum des Auslesezeitraums (Format: yyyyMMdd)

Variable (Datenquellentyp)	Beschreibung
$BEGINTIME$ (SAP, JDBC)	Startzeit des Auslesezeitraums (Format: HHmmss)
$ENDDATE$ (SAP, JDBC)	Enddatum des Auslesezeitraums (Format: yyyyMMdd)
$ENDTIME$ (SAP, JDBC)	Endzeit des Auslesezeitraums (Format: HHmmss)
$VALUECONSTRAINT$ (SAP, s. Kap. Bedingungsoperatoren (Seite 27) sowie JDBC, s. Kap. Bedingungsoperatoren (Seite 85))	Ausgabeformat: <Operator>_Wert1 oder <Operator>_Wert1_<Operator>_Wert2 Ausgabe der Operatoren und ganzzahligen Vergleichswerte, die zum Einschränken des gelesenen Datenbestandes verwendet werden, der Name einer Ausgabedatei könnte bspw. lauten: **outfile_gt_230_le_300_<x>.xml**, wobei **x** die Ziffer der durchlaufenden Nummerierung ist. Darstellung der Operatoren: größer als: **gt** größer oder gleich: **ge** kleiner als: **lt** kleiner oder gleich: **le**

Die Aufteilung des gelesenen Datenbestandes auf mehrere Ausgabedateien können Sie bequem im PPM Customizing Toolkit in der **Datenquellenverwaltung** des Mandanten über **Weitere Einstellungen...** im Bereich **Datenextraktion** einstellen.

10.2.2 Konfiguration eines zeit- bzw. wertversetzten Auslesevorgangs

Die in diesem Kapitel beschriebenen Funktionalitäten werden ausschließlich von den beiden Prozessextraktoren **Process Extractor JDBC-2-PPM** und **Process Extractor JDBC-2-PPM** und nur bei Verwendung einer Datenquellenkonfiguration unterstützt.

Für die zeitbasierte Extraktion können Sie in der Datenquellenkonfiguration einen Zeitwert angeben. Der Wert wird bei Ausführung des entsprechenden Kommandozeilenprogramms ohne Endzeitparameter (**-enddate** bzw. **-endtime**) vom aktuellen Ausführungszeitpunkt als

Endzeitpunkt abgezogen. Die Datenextraktion wird um den angegebenen Wert (in Sekunden) früher beendet.

Ohne Angabe eines Zeitversatzwertes kann es vorkommen, dass bereits angelegte, aber noch nicht fest in die auszulesende Datenbank geschriebene Datensätze aufgrund ihres rückversetzten Zeitstempels in keinem von zwei aufeinanderfolgenden Extraktionsvorgängen (vgl. **Fortlaufendes automatisiertes Auslesen** (Seite 65)) ausgelesen werden. Dies können Sie durch die Angabe eines Versatzwertes verhindern.

Beispiel (SAP-Datenquelle)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE datasource SYSTEM "datasource.dtd">

<datasource name="SAP" type="MYSAP" lastreaddate="19700101"
                lastreadtime="000000" readoffset="86400">
    <description name="default_description" language="en"/>
    <description name="default_description" language="de">
       SAP
    </description>
    Ermittle die neuen bzw. geänderten MM-Belege ...
</datasource>
```

Mit dem XML-Attribut **readoffset** ist ein Versatzwert von **86400 Sekunden** angegeben. Bei Aufruf des Kommandozeilenprogramms mit

```
runjdbc2ppm -datasource SAP.xml
```

endet der Auslesevorgang einen Tag (86400 Sekunden) vor dem aktuellen Ausführungszeitpunkt. Ist der Ausführungszeitpunkt des Programms bspw. der 11.08.08 9:48:05 Uhr, endet der Extraktionsvorgang am 10.08.08 um 9:48:05 Uhr.

```
Ermittle die neuen bzw. geänderten MM-Belege ...
I: 11.08.08 09:48:06: [XML] Lese Daten mit Startdatum/-zeit 19700101/000000 und
Enddatum/-zeit 20080810/094805...
Ermittle die neuen bzw. geänderten MM-Belege ...
```

Der zeitlich nach hinten versetzte Wert des letzten Auslesezeitpunkts wird als Anfangszeitpunkt der nächsten Datenextraktion in die Datenquellendatei geschrieben.

Für das bedingte Auslesen mittels eines Ganzzahlkriteriums (logischer Operator **valueconstraint**, siehe **Auslesen mit Bedingungen** (Seite 84)) kann analog ein Versatzwert angegeben werden, der bei der Programmausführung vom Startwert (**lastreadvalue** aus der Datenquellendatei) subtrahiert wird. Sollte sich aufgrund Ihrer Angaben ein negativer Wert für das Ganzzahlkriterium ergeben, wird dieser wie ein positiver Wert behandelt.

Ein in der Datenquellenkonfiguration angegebener Versatzwert wird nur dann bei der Datenextraktion berücksichtigt, wenn der Endzeitpunkt bzw. die Untergrenze des Ganzzahlkriteriums nicht explizit angegeben wird.

Tipp

Sie können auf einfache Weise Versatzwerte in **PPM Customizing Toolkit** in der Modulgruppe **Mandant** unter **Datenquellenverwaltung** konfigurieren. Geben Sie die Werte als zusätzliche Angaben zum letzten Auslesezeitpunkt bzw. zum letzten ausgelesenen Wert in den Komponenten der betreffenden Datenquelle an.

ANGABE EINER ZEITZONE

Wenn Sie in der Kommandozeile den Endzeitpunkt nicht angegeben haben (Argumente **-enddate** bzw. **-endtime**), wird die aktuelle Zeit des lokalen Rechners standardmäßig als Endzeitpunkt verwendet. Dieser Endzeitpunkt wird bei den Operatoren **creationtimestamp**, **date_creationtimestamp** bzw. **char_creationtimestamp** verwendet, um die auszulesende Datenmenge einzuschränken. Wenn das in diesen Operatoren referenzierte Datenfeld Zeitstempel enthält, die einer anderen Zeitzone zugeordnet sind, können Sie diese Zeitzone im XML-Attribut **sourcefieldtimezone** angeben (z. B. **Australia/Canberra**). Bei der folgenden Datenextraktion wird die lokale Zeit des Rechners vor der Extraktion in die angegebene Zeitzone umgerechnet und nach erfolgreicher Extraktion als letzter Extraktionszeitpunkt (XML-Attribute **lastreaddate** und **lastreadtime**) in der Datenquelle gespeichert.

Wenn im XML-Attribut **sourcefieldtimezone** keine gültige Zeitzone, oder das Attribut selbst nicht angegeben ist, wird keine Zeitzonenumrechnung des Endzeitpunktes durchgeführt. Mittels CTK kann die Zeitzone komfortabel über die Oberfläche gewählt werden.

Beispiele Zeitzonen

Africa/Dakar, Africa/Johannesburg, Australia/Sydney, America/Denver, America/Los_Angeles, America/Mexico_City, Canada/Central, Europe/Berlin, Europe/London, Mexico/General, US/Central, US/Hawaii

10.3 XML-Ausgabedatei (Formate)

Standardmäßig werden die mit den PPM Prozessextraktoren extrahierten Daten in eine XML-Ausgabedatei im PPM-System-Event-Format geschrieben.
Durch Angabe eines der optionalen Kommandozeilenparameter **-pikidatamapping** bzw. **-dimdatamapping** können Sie die gelesenen Daten alternativ im PIKIDATA- bzw. DIMDATA-Format in die Ausgabedatei(en) schreiben lassen.
Die verschiedenen Ausgabeformate sind ausführlich im Handbuch **PPM Datenimport** beschrieben.

10.3.1 PPM-System-Event-Format

Standardmäßiges Format der XML-Ausgabedatei(en), wenn keiner der beiden Kommandozeilenparameter **-pikidatamapping** bzw. **-dimdatamapping** verwendet wird (siehe Kapitel **XML-Ausgabedatei (PPM-System-Event-Format)** (Seite 35)).

10.3.2 PIKIDATA-Format

Die ausgelesenen Daten sollen als Daten prozessinstanzunabhängiger Kennzahlen (PIKI) in eine XML-Ausgabedatei für den PPM-Import mit dem Kommandozeilenprogramm **runpikidata.bat** geschrieben werden.

Die XML-Konfigurationsdatei hat grundsätzlich folgenden Aufbau:

```
...
<pikidatamapping>
  <pikicube name="...">
    <pikicolmapping>
      <datacol name="..."/>
      <keyspec key="..."/>
    </pikicolmapping>
    ...
  </pikicube>
  ...
</pikidatamapping>
```

XML-Element	XML-Attribut	Beschreibung
pikidatamapping		Root-Element der Mapping-Definition. Enthält eine Liste zu konfigurierender PPM-Kennzahlreihen
pikicube	name	Identifizierer der PPM-Kennzahlreihe (entspricht dem Wert des XML-Attributes **name** der Kennzahlreihen-konfiguration in PPM)
pikicolmapping		Gruppiert Konfigurationselemente für eine Datenspalte
datacol	name	Name der Datenspalte, in der die PIKI-Werte gespeichert werden (entspricht dem internen Namen der PIKI in der Kennzahlenkonfiguration)
	format (optional)	Formatzeichenkette zum Transformieren des Datenwertes
keyspec	key	Name des Schlüssels der Schlüssel-Wert-Zuweisung
	desckey	Name des Schlüssels der Schlüssel-Beschreibung-Zuweisung
value		Wert der prozessinstanzunabhängigen Kennzahl

Folgender Dateiauszug zeigt die bestehende PPM-Kennzahlenreihe **PCA-Import**:

```
...
<pikicube name="PIKICUBE_TURNOVER">
  <description language="de" name="PCA-Import"/>
  <description language="en" name="PCA Import"/>
  <pikidef name="TURNOVER" retrievertype="NUM_KEYINDICATOR"
           dimreferring="LOOSE" kigroup="KI_GROUP_COST">
    <description language="de" name="Umsatz">
                               Umsatz
    </description>
    <description language="en" name="Sales revenues">
                               Sales revenues
    </description>
    <datatype name="COST"/>
  </pikidef>
  <refdim name="VKORG"/>
  <refdim name="DIVISION"/>
  <refdim name="TIME" refinement="BY_MONTH"/>
</pikicube>
...
```

Beispiel

Als Ergebnis des Auslesevorgangs werden folgende Schlüssel-Wert-Paare in der XML-Ausgabedatei im PPM-System-Event-Format erwartet:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="MARA-TURNOVER">225489 EUR</attribute>
    <attribute type="MARB-VKORG_ID">1000</attribute>
    <attribute type="MARB-VKORG_DESC">Germany Hamburg
    </attribute>
    <attribute type="DIM_ROUGH_ID">01</attribute>
    <attribute type="DIM_ROUGH_DESC">Product category 01
    </attribute>
    <attribute type="DIM_DETAILED_ID">8112</attribute>
    <attribute type="DIM_DETAILED_DESC">8112 description
    </attribute>
    <attribute type="CAL-TIME">January 2002</attribute>
  </event>
  <event>
    <attribute type="MARA-TURNOVER">135699 EUR</attribute>
    <attribute type="MARB-VKORG_ID">4000</attribute>
    <attribute type="MARB-VKORG_DESC">Austria Vienna
    </attribute>
    <attribute type="DIM_ROUGH_ID">01</attribute>
    <attribute type="DIM_ROUGH_DESC">Product category 01
    </attribute>
    <attribute type="DIM_DETAILED_ID">8112</attribute>
    <attribute type="DIM_DETAILED_DESC">8112 description
    </attribute>
    <attribute type="CAL-TIME">January 2002</attribute>
  </event>
  <event>
    <attribute type="MARA-TURNOVER">363521 EUR</attribute>
    <attribute type="MARB-VKORG_ID">1000</attribute>
    <attribute type="MARB-VKORG_DESC">Germany Hamburg
    </attribute>
```

```
    <attribute type="DIM_ROUGH_ID">07</attribute>
    <attribute type="DIM_ROUGH_DESC">High Tech</attribute>
    <attribute type="DIM_DETAILED_ID">9128</attribute>
    <attribute type="DIM_DETAILED_DESC">9128 description
    </attribute>
    <attribute type="CAL-TIME">January 2002</attribute>
  </event>
</eventlist>
```

Durch folgende Mapping-Datei werden die Schlüssel-Wert-Paare in das PPM-konforme XML-Format zum Einlesen von PIKI-Werten überführt:

```
...
<pikidatamapping>
  <pikicube name="PIKICUBE_TURNOVER">
    <pikicolmapping>
      <datacol name="TURNOVER"/>
      <keyspec key="MARA-TURNOVER"/>
    </pikicolmapping>
    <pikicolmapping>
      <datacol name="VKORG"/>
      <keyspec key="MARB-VKORG_ID"
                          desckey="MARB-VKORG_DESC"/>
    </pikicolmapping>
    <pikicolmapping>
      <datacol name="DIVISION"/>
      <keyspec key="DIM_ROUGH_ID"
                           desckey="DIM_ROUGH_DESC"/>
    </pikicolmapping>
    <pikicolmapping>
      <datacol name="DIVISION"/>
      <keyspec key="DIM_DETAILED_ID"
                         desckey="DIM_DETAILED_DESC"/>
    </pikicolmapping>
    <pikicolmapping>
      <datacol name="TIME" format="MMM yyyy"/>
      <keyspec key="CAL-TIME"/>
    </pikicolmapping>
  </pikicube>
</pikidatamapping>
```

Unter Verwendung dieser Mapping-Datei wird bei Programmausführung folgende XML-Ausgabedatei erzeugt:

```
...
<pikidata>
  <pikicube name="PIKICUBE_TURNOVER">
    <datacols>
      <datacol name="TURNOVER"/>
      <datacol name="VKORG"/>
      <datacol name="DIVISION"/>
      <datacol name="DIVISION"/>
      <datacol name="TIME" format="MMM yyyy"/>
    </datacols>
    ...
    <datarow>
      <value>225489 EUR</value>
      <value>1000{Germany Hamburg}</value>
      <value>01{Product category 01}</value>
      <value>8112{8112 description}</value>
      <value>January 2002</value>
    </datarow>
    <datarow>
```

```
      <value>135699 EUR</value>
      <value>4000{Austria Vienna}</value>
      <value>01{Product category 01}</value>
      <value>8112{8112 description}</value>
      <value>January 2002</value>
    </datarow>
    <datarow>
      <value>363521 EUR</value>
      <value>1000{Germany Hamburg}</value>
      <value>07{High Tech}</value>
      <value>9128{9128 description}</value>
      <vallue>January 2002</value>
    </datarow>
    ...
  </pikicube>
</pikidata>
...
```

10.3.3 DIMDATA-Format

Die durch den jeweiligen PPM Prozessextraktor ausgelesenen Daten sollen in dem speziellen Importformat für ein-, zwei- bzw. mehrstufige Dimensionen in die XML-Ausgabedatei geschrieben werden. XML-Ausgabedateien dieses Formats können mit dem Kommandozeilenprogramm **rundimdata.bat** nach PPM importiert werden (siehe Technische Referenz **PPM Datenimport**).

Für die korrekte Datenextraktion benötigt der jeweilige Prozessextraktor eine Mapping-Datei, die den Schlüsseln (und ggf. Beschreibungen) der einzelnen Dimensionsstufen die korrekten, gelesenen Feldwerte zuordnet. Diese XML-Konfigurationsdatei hat folgenden Aufbau:

```
...
<dimdatamapping>
  <dimension name="...">
    <dimcolmapping>
      <datacol name="..."/>
      <keyspec key="..."/>
    </dimcolmapping>
    ...
  </dimension>
</dimdatamapping>
...
```

XML-Element	XML-Attribut	Beschreibung
dimdatamapping		Root-Element der Mapping-Definition. Enthält eine Liste zu konfigurierender PPM-Dimensionen.
dimension	name	Identifizierer der PPM-Dimension (entspricht dem Wert des XML-Attributes **name** der Dimensions-konfiguration im PPM-System)

XML-Element	XML-Attribut	Beschreibung
dimcolmapping		Gruppiert Konfigurationselemente für eine Datenspalte.
datacol	name	Name der Datenspalte, in der die Dimensionswerte gespeichert werden
	format (optional)	Formatzeichenkette zum Transformieren des Datenwertes zur Verwendung im PPM-System
keyspec	key	Name des Schlüssels der gelesenen Schlüssel-Wert-Zuweisungen
	desckey	Name des Schlüssels der gelesenen Schlüssel-Beschreibung-Zuweisungen
value		Angabe eines konstanten Dimensions-wertes

Beispiel

Als Ergebnis des Auslesevorgangs werden folgende Schlüssel-Wert-Paare in der XML-Ausgabedatei im PPM-System-Event-Format erwartet:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="MAT1_ID-MATNR">1000</attribute>
    <attribute type="MAT1_DESC-HTEXT">Motor44</attribute>
    <attribute type="MAT2_ID-MTART">32</attribute>
    <attribute type="MAT2_DESC-MTBEZ">Components</attribute>
  </event>
  <event>
    <attribute type="MAT1_ID-MATNR">1001</attribute>
    <attribute type="MAT1_DESC-HTEXT">Wrench77</attribute>
    <attribute type="MAT2_ID-MTART">45</attribute>
    <attribute type="MAT2_DESC-MTBEZ">Tools</attribute>
  </event>
  <event>
    <attribute type="MAT1_ID-MATNR">1002</attribute>
    <attribute type="MAT1_DESC-HTEXT">Pump43</attribute>
    <attribute type="MAT2_ID-MTART">33</attribute>
    <attribute type="MAT2_DESC-MTBEZ">Trade goods</attribute>
  </event>
</eventlist>
```

Durch folgende Mapping-Datei werden die Schlüssel-Wert-Paare in das PPM-konforme XML-Format zum Einlesen von Dimensionswerten überführt:

```
...
```

```
<dimdatamapping>
  <dimension name="MATERIAL">
    <dimcolmapping>
      <datacol name="LEVEL1_ID"/>
      <keyspec key="MAT1_ID-MATNR"/>
    </dimcolmapping>
    <dimcolmapping>
      <datacol name="LEVEL1_DESC"/>
      <keyspec key="MAT1_DESC-HTEXT"/>
    </dimcolmapping>
    <dimcolmapping>
      <datacol name="LEVEL2_ID"/>
      <keyspec key="MAT2_ID-MTART"/>
    </dimcolmapping>
    <dimcolmapping>
      <datacol name="LEVEL2_DESC"
      <keyspec key="MAT2_DESC-MTBEZ"/>
    </dimcolmapping>
  </dimension>
</dimdatamapping>
```

Die Namen der Datenspalten einer PPM-Dimensionsstufe sind fest vorgegeben (**LEVEL1_ID**, **LEVEL1_DESC**, **LEVEL2_ID**, **LEVEL2_DESC**, ...) und dürfen nicht geändert werden.

Die folgende Tabelle veranschaulicht die Zuordnung der Datenspalten zu den Schlüsseln und Beschreibungen der einzelnen Dimensionsstufen:

dimdata-Konfiguration	Dimensionskonfiguration
LEVEL1_ID	Identifizierer der ersten Dimensionsstufe
LEVEL1_DESC	Beschreibung der ersten Dimensionsstufe
LEVEL2_ID	Identifizierer der zweiten Dimensionsstufe
LEVEL2_DESC	Beschreibung der zweiten Dimensionsstufe
LEVELn_ID	Identifizierer der n-ten Dimensionsstufe
LEVELn_DESC	Beschreibung der n-ten Dimensionsstufe

Die erzeugte XML-Ausgabedatei im DIMDATA-Format hat folgenden Inhalt:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE dimdata SYSTEM "dimdata.dtd">
<dimdata>
  <dim name="MATERIAL">
    <datacols>
      <datacol name="LEVEL1_ID"/>
      <datacol name="LEVEL1_DESC"/>
      <datacol name="LEVEL2_ID"/>
      <datacol name="LEVEL2_DESC"/>
    </datacols>
    <datarow>
      <value>1001</value>
      <value>Motor44</value>
      <value>32</value>
```

```
    <value>Components</value>
  </datarow>
  <datarow>
    <value>1002</value>
    <value>Wrench77</value>
    <value>45</value>
    <value>Tools</value>
    </datarow>
  <datarow>
    <value>1003</value>
    <value>Pump43</value>
    <value>33</value>
    <value>Trade goods</value>
  </datarow>
</dim>
```