ARIS Process Performance Manager

DATENIMPORT

Version 10.1

Oktober 2017

This document applies to PPM Version 10.1 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Inhalt

1 Textkonventionen 1

2 Allgemeines 2

3 XML 3
3.1 XML - Was ist das? 3
3.2 Aufbau eines XML-Dokuments 3

4 XML-Datenimport 4
4.1 Graphformat 5
4.1.1 Objekttypen 6
4.1.2 Kanten 7
4.1.3 Beziehungen (optional) 8
4.1.4 Richtlinien für den Graphaufbau 9
4.1.5 Attribute 9
4.1.6 XML-Beispielgraph 12
4.2 System-Event-Format 14
4.2.1 Definition der Prozessfragmente 15
4.2.2 Definition des Mapping 17
4.2.2.1 Definition des Prozessfragment-Mapping 17
4.2.2.2 Definition des Attribut-Mapping 21
4.2.2.2.1 Attributtransformationen 23
4.2.2.2.1.1 Zeitstempeltransformationen 23
4.2.2.2.1.2 Fließkommazahltransformation 28
4.2.2.3 Organisationseinheiten 29
4.2.2.4 Sonderfall des Attribut-Mapping 30
4.2.3 Erstellen von Fragmentdefinitionen in ARIS 31
4.2.3.1 Modellieren des Gesamtprozesses 32
4.2.3.2 Modellieren der Prozessfragmentdefinitionen 33
4.2.3.3 Format der System-Event-Datei 36
4.2.3.4 ARIS-Report ausführen 37
4.2.4 Generieren der XML-Ausgabedatei 39
4.2.5 Zusammenfassung 39
4.3 Datenformate 44
4.3.1 Sonderzeichen in XML-Dokumenten 45
4.4 Erzeugen der Prozessinstanzfragmente 46
4.4.1 Erweiterung der Attributkonfiguration 46
4.4.1.1 Datentyp unbekannter Attribute bestimmen 48
4.4.2 Erweiterung der Mapping-Konfiguration 52
4.4.3 Mehrwertige System-Event-Attribute 53
4.4.4 Direkter Import von Prozessattributen 54
4.4.5 Sonderfall skaliertes System 56
4.4.6 Archivierung der XML-Importdateien 58
4.5 Kommandozeilenprogramm runxmlimport 59
4.5.1 Argumente von runxmlimport 60
4.6 Mehrere Datenquellen importieren 63
4.7 Erneuter Import derselben Daten 64
4.7.1 Graphformat 64
4.7.2 System-Event-Format 64

5 Import prozessinstanzunabhängiger Daten 66
5.1 Prozessinstanzunabhängige Kennzahlen 66
5.1.1 Datenimportformate 66
5.1.1.1 XML-Format 67
5.1.1.2 CSV-Format 71
5.1.1.3 XLS-Format 72
5.1.2 Erneuter Datenimport 74
5.1.3 Export von Werten prozessinstanzunabhängiger Datenreihen 75
5.1.4 Löschen von Werten prozessinstanzunabhängiger Datenreihen 76
5.1.5 Kommandozeilenprogramm runpikidata 76
5.2 Dimensionswerte 79
5.2.1 XML-Format 79
5.2.2 CSV-Format 81
5.2.3 Standard- und Ersatzwerte 81
5.2.4 Erneuter Datenimport 81
5.2.5 Löschen von Dimensionswerten 82
5.2.6 Kommandozeilenprogramm rundimdata 82
5.3 Data Analytics 84

6 Behandlung großer EPKs 85
6.1 Import großer EPKs 85
6.2 Löschen großer EPKs 85

7 Anhang 86
7.1 Aufbau eines Process Warehouse 86
7.1.1 Prozessfragmente generieren 88
7.1.2 Prozessfragmente zusammenführen 90
7.1.2.1 Kopieren der Prozessinstanzattribute 92
7.1.2.2 Anonymisieren der Organisationseinheiten 92
7.1.3 Prozesse typisieren 93
7.1.4 Kennzahlen berechnen 93
7.1.5 Planwerte überprüfen 94

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 Allgemeines

PPM verwendet **XML** als universelles Datenformat, wobei das gesamte PPM-System mit Hilfe von XML-Dateien konfiguriert werden kann.

Diese technische Referenz beschreibt die XML-Schnittstellen von PPM, über die Quellsystemdaten als XML-Dateien ins PPM-System importiert werden. Zum Datenimport verwendet PPM die eigenen Datenformate **System-Event-Format** und **Graphformat**.
Die Quellsystemdaten werden zuvor mit Hilfe der PPM-Prozessextraktoren (System-Event-Format) oder anderen Adaptern aus beliebigen Quellsystemtypen (z. B. SAP, JDBC, CSV) ausgelesen.

Das Kapitel **Aufbau eines Process Warehouse** (Seite 86) beschreibt aus konzeptioneller Sicht, wie aus extrahierten Quellsystemdaten Prozessinstanzen erzeugt werden, die die Abläufe der Quellsystemprozesse wiedergeben und zur weiteren Kennzahlanalyse in PPM zur Verfügung stehen.

3 XML

Dieses Kapitel enthält Grundlagenwissen zu XML, das zum Verständnis der weiteren Kapitel hilfreich ist.

3.1 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.

3.2 Aufbau eines XML-Dokuments

Ein XML-Dokument ist eine Textdatei und setzt sich aus zwei Zeichenarten zusammen: den eigentlichen Daten und den sogenannten 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 direkt in der XML-Datei in eckigen Klammern angegeben:

```
<?xml version="1.0"?>
<!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 XML-Datenimport

Dieses Kapitel beschreibt den XML-basierten Import von Prozessinstanzdaten.

Die Instanzdaten der real abgelaufenen Prozesse werden durch eine spezielle Software aus dem operativen Anwendungssystem (Quellsystem) ausgelesen und in XML-Ausgabedateien gespeichert. Diese Ausgabedateien werden mit Hilfe der XML-Importschnittstelle nach PPM importiert. Der interne Aufbau der XML-Dateien ist durch eine DTD (Document Type Definition) vorgegeben.

Die XML-Importschnittstelle von PPM unterstützt zwei unterschiedliche Importformate, das PPM-Graphformat und das PPM-System-Event-Format.

PPM-GRAPHFORMAT

Das PPM-Graphformat wird verwendet, um bereits strukturierte Prozessdaten aus prozessorientierten Anwendungssystemen (z. B. Workflow-Systemen) zu importieren. Der anwendungssystemspezifische Adapter generiert XML-Dateien, in denen Prozessinstanzen einschließlich ihrer Ablauflogik im PPM-Graphformat beschrieben werden. Im Unterschied zum PPM-System-Event-Format können vollständige Prozessinstanzen importiert werden. Ein Merge-Vorgang ist nicht notwendig. Beim Import vollständiger Prozessinstanzen müssen bei einem erneuten Import der Instanzdaten immer vollständige Prozessinstanzen eingelesen werden.

Innerhalb des PPM-Systems wird das Graphformat zum universellen Austausch EPK-basierter Daten verwendet.

PPM-SYSTEM-EVENT-FORMAT

Das PPM-System-Event-Format wird für alle tätigkeitsorientierten Anwendungssysteme eingesetzt, bei denen die prozessbildende Information (Ablauflogik) nicht ausgelesen werden kann.
Beim Import von Daten im System-Event-Format werden System-Events in einer XML-Datei protokolliert. Alle Typen von System-Event, die nach PPM importiert werden sollen, müssen vor dem Import in Prozessfragmentmodellen definiert werden. Darüber hinaus werden Regeln definiert, wie diese Prozessfragmentmodelle zu einem Gesamtprozess zusammengeführt werden.
Durch Abbilden (Mapping) der System-Events auf Prozessfragmentmodelle erzeugt PPM Prozessinstanzfragmente. Diese werden anschließend zu Prozessinstanzen verkettet.
Das System-Event-Format ermöglicht die Erweiterung und Veränderung einzelner bereits importierter Prozessinstanzen durch den Import von Deltadaten.

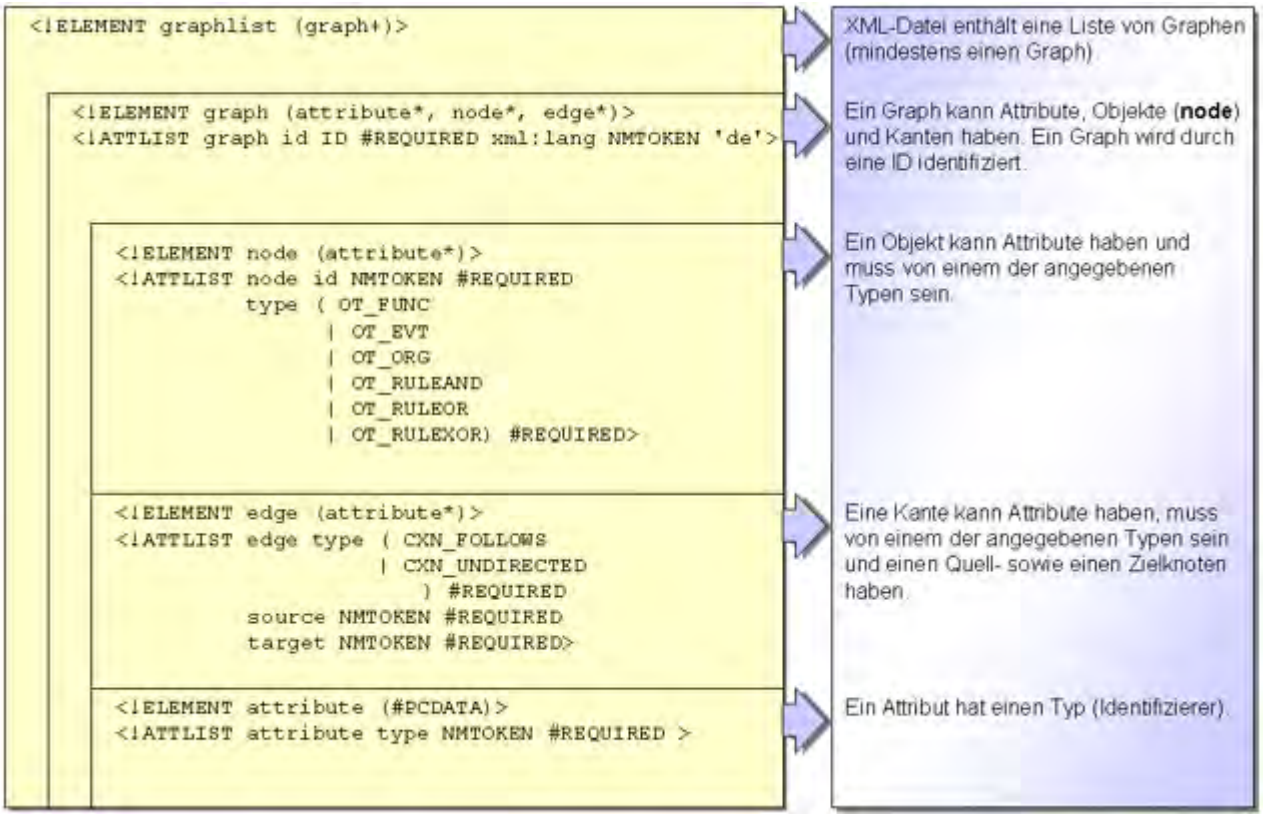
4.1 Graphformat

Eine XML-Datei im PPM-Graphformat beinhaltet eine Liste von Graphen (EPKs). Jeder Graph stellt eine Prozessinstanz oder ein Prozessinstanzfragment dar. Ein Graph setzt sich aus Objekten, Kanten und gegebenenfalls Beziehungen (Relationen) verschiedenen Typs zusammen. Sowohl der Graph als auch die Objekte, Kanten und Beziehungen können Attribute tragen.

Die folgende XML-Datei enthält einen einfachen Graphen, der sich aus drei miteinander verbundenen Objekten (Ereignis - Funktion - Ereignis) zusammensetzt:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE graphlist SYSTEM "graph.dtd">
<graphlist>
  <graph id="00093862" xml:lang="de">
    <attribute type="AT_ID">XMLGraph-Auftrag-00093862</attribute>
    <attribute type="AT_EPK_KEY">00093862</attribute>
    <attribute type="AT_PROCTYPE">Terminauftrag</attribute>
    <attribute type="AT_PROCTYPEGROUP">Auftragsabwicklung</attribute>

    <node id="Start" type="OT_EVT">
      <attribute type="AT_OBJNAME_INTERN">AUFTRAG_ANZU</attribute>
      <attribute type="AT_OBJNAME">Kundenauftrag ist anzulegen</attribute>
    </node>

    <node id="Funktion" type="OT_FUNC">
      <attribute type="AT_OBJNAME_INTERN">AUFTRAG</attribute>
      <attribute type="AT_OBJNAME">Kundenauftrag anlegen</attribute>
      <attribute type="AT_START_TIME">14.2.2000 13:12:57</attribute>
      <attribute type="AT_END_TIME">14.02.2000 13:22:57</attribute>
    </node>

    <node id="Bearbeiter" type="OT_ORG">
      <attribute type="AT_OBJNAME">Frau Schmidt</attribute>
    </node>

    <node id="Ende" type="OT_EVT">
      <attribute type="AT_OBJNAME_INTERN">LIEFERUNG_ANZU</attribute>
      <attribute type="AT_OBJNAME">Lieferung ist anzulegen</attribute>
      <attribute type="AT_ID">XMLGraph-Auftrag-Evt2</attribute>
    </node>

    <edge type="CXN_FOLLOWS" source="Start" target="Funktion" />
    <edge type="CXN_FOLLOWS" source="Funktion" target="Ende" />
    <relationtype name="REL_CARRY_OUT">
      <relation source="Bearbeiter" target="Funktion">
        <attribute type="AT_KI_PK_R">7.5 EUR</attribute>
        <attribute type="AT_KI_RNUM">1</attribute>
      </relation>
    </relationtype>
  </graph>
</graphlist>
```

Beim Einlesen in PPM wird der Aufbau von XML-Dateien im Graphformat gegen folgende DTD verifiziert:



4.1.1 Objekttypen

Folgende Tabelle zeigt alle Objekttypen, die im PPM-Graphformat verwendet werden:

Objekttyp	Identifizierer	Beschreibung
Funktion	OT_FUNC	Funktionen beschreiben Aktivitäten im Prozess. Über Funktionsattribute werden Istwerte für die Kennzahlenberechnung übermittelt.
Ereignis	OT_EVT	Ereignisse sind Prozesszustände und beschreiben den eine Funktion auslösenden Zustand und das Ergebnis einer Funktionsausführung. Prozessfragmente werden über gleichartige Ereignisse zu Prozessinstanzen zusammengeführt.

Objekttyp	Identifizierer	Beschreibung
Organisations-einheit (optional)	OT_ORG	Durch Anonymisieren können Bearbeiter einer Funktion Organisationseinheiten zugeordnet werden. Die Attribute der Organisationseinheiten sind Basis für die Prozesskostenrechnung.
UND-Regel (optional)	OT_RULEAND	Spaltet einen Prozessablauf auf oder führt ihn zusammen. Die beiden Prozesspfade, die der UND-Regel folgen, werden durchlaufen.
ODER-Regel (optional)	OT_RULEOR	Spaltet einen Prozessablauf auf oder führt ihn zusammen. Mindestens einer der Prozesspfade, die der ODER-Regel folgen, wird durchlaufen.
XOR-Regel (optional)	OT_RULEXOR	Spaltet einen Prozessablauf auf oder führt ihn zusammen. Genau einer der Prozesspfade, die der XOR-Regel folgen, wird durchlaufen.

Innerhalb eines Graphen wird ein Objekt durch die **node id** eindeutig bestimmt. Die **node id** steht im XML-Attribut **id** des XML-Elements **node**. Objekte mit gleicher **node id** sind nicht erlaubt und werden zu einem Objekt zusammengefasst.

4.1.2 Kanten

Folgende Tabelle zeigt alle in PPM erlaubten Kanten:

Kantentyp	Identifizierer	Beschreibung
Flusskante	CNX_FOLLOWS	Verbindet strukturbildende Objekte im Prozessablauf (Ereignisse, Funktionen, Regeln) des Graphen.
Kommentarkante	CNX_UNDIRECTED	Ordnet eine Organisationseinheit einer Funktion, die sie ausführt, zu.

4.1.3 Beziehungen (optional)

Folgende Tabelle zeigt alle in PPM erlaubten Beziehungstypen:

Beziehungstyp	Identifizierer	Beschreibung
Führt aus	REL_CARRY_OUT	Stellt eine Beziehung her zwischen einer Funktion und der ausführenden Organisationseinheit.
Arbeitet zusammen mit (ohne Brüche)	REL_WORKS_ TOGETHER	Stellt eine Beziehung her zwischen Organisationseinheiten als Quellbezugsobjekt und Organisationseinheiten als Zielbezugsobjekt. Eine Beziehungsberechnung findet nur statt zwischen direkt aufeinander folgenden Funktionsinstanzen, an denen Organisationseinheiten gepflegt sind. Dazwischen dürfen keine weiteren Funktionsinstanzen ohne Organisationseinheiten vorkommen.
Arbeitet zusammen mit (mit Brüchen)	REL_WORKS_ TOGETHER_LONG_ DISTANCE	Stellt eine Beziehung her zwischen Organisationseinheiten als Quellbezugsobjekt und Organisationseinheiten als Zielbezugsobjekt. Zwischen den Funktionsinstanzen, an denen die Organisationseinheiten gepflegt sind, dürfen weitere Funktionsinstanzen ohne Organisationseinheiten vorkommen. Diese werden bei der Beziehungsberechnung ignoriert, d. h., die Funktionsinstanzen mit den Organisationseinheiten müssen nicht direkt aufeinander folgen.
Ping Pong	REL_PING_PONG	Stellt eine Beziehung her zwischen Organisationseinheiten als Quellbezugsobjekt und Organisationseinheiten als Zielbezugsobjekt.

Die eigentlichen Beziehungsausprägungen tragen die Kennzahlenattribute, auf deren Basis - gegebenenfalls für jeden einzelnen Beziehungstyp - die in der Kennzahlenkonfiguration definierten Beziehungskennzahlen berechnet werden.

Wie Beziehungen definiert und berechnet werden, entnehmen Sie der Technischen Referenz **PPM Customizing**.

4.1.4 Richtlinien für den Graphaufbau

Aus den allgemeinen EPK-Konventionen (Reihenfolge im Kantenfluss **Ereignis - Funktion - Ereignis**) ergeben sich folgende Richtlinien zum Erstellen des Graphen einer Prozessinstanz:

- Eine Prozessinstanz muss mit einem oder mehreren Ereignissen beginnen und enden und mindestens eine Funktion enthalten.
- Prozessinstanzen können als Regeln ausschließlich UND-Regeln enthalten, da es sich um real existierende Arbeitsabläufe handelt.
- Nach einem Ereignis darf keine verzweigende Regel (OR oder XOR) folgen.
- Auf eine Funktion darf eine Funktion folgen. Das verbindende Ereignis zwischen zwei Funktionen darf entfallen.
- Auf ein Ereignis darf nur eine Funktion oder ein zusammenführender Konnektor folgen.

4.1.5 Attribute

Folgende Attribute müssen am Graphen einer Prozessinstanz und an ihren Objekten gepflegt sein:

PROZESSATTRIBUTE

Attributname	Identifizierer	Datentyp	Beschreibung
Prozess-ID (optional)	AT_EPK_KEY	TEXT	Eindeutiger Identifizierer der Prozessinstanz. Der Attributwert sollte mit der Graph-ID (Tag **id** des XML-Elements **<graph>**) übereinstimmen. Dieses Attribut kennzeichnet eine Instanz als abgeschlossen.

Attributname	Identifizierer	Datentyp	Beschreibung
Prozesskenn-zeichnung	AT_ID	TEXT	Prozesskennzeichnung. Attributwert wird in der Prozessinstanzliste und der EPK-Ansicht der Prozess-instanz angezeigt. Bei abge- schlossenen Instanzen bietet sich als Attributwert der Wert des Attributs **AT_EPK_KEY** an.
Prozesstypgruppe	AT_PROCTYPE GROUP	TEXT	Name der Prozesstypgruppe, zu der die Prozessinstanz gehört. Falls das Attribut nicht gepflegt ist, wird es vom Typisierer erzeugt.
Prozesstyp	AT_PROCTYPE	TEXT	Name des Prozesstyps der Prozessinstanz. Falls das Attribut nicht gepflegt ist, wird es vom Typisierer erzeugt.
Instanz	AT_IS_PROC INSTANCE	BOOLEAN	Gibt an, ob es sich um den Graphen einer einzelnen Prozessinstanz (nicht gepflegt oder Wert = TRUE) oder einer verdichteten Prozessinstanz (Wert = FALSE) handelt.

OBJEKTATTRIBUTE

Attributname	Identifizierer	Datentyp	Beschreibung
Name	AT_OBJNAME	TEXT	Objektname. Wird für die EPK-Ansicht verwendet.
Interner Name	AT_OBJNAME_ INTERN	TEXT	Interner sprachunabhängiger Name des Objekts. Wird zur Referenzierung des Objekts verwendet.

Attributname	Identifizierer	Datentyp	Beschreibung
Startzeitpunkt	AT_START_TIME	TIME-STAMP	Gibt den Startzeitpunkt der Ausführung einer Funktion an. Ist optional, wenn an einer Funktion das Attribut **Endzeitpunkt** gepflegt ist.
Endzeitpunkt	AT_END_TIME	TIME-STAMP	Gibt den Endzeitpunkt der Ausführung einer Funktion an. Ist optional, wenn an einer Funktion das Attribut **Startzeitpunkt** gepflegt ist.

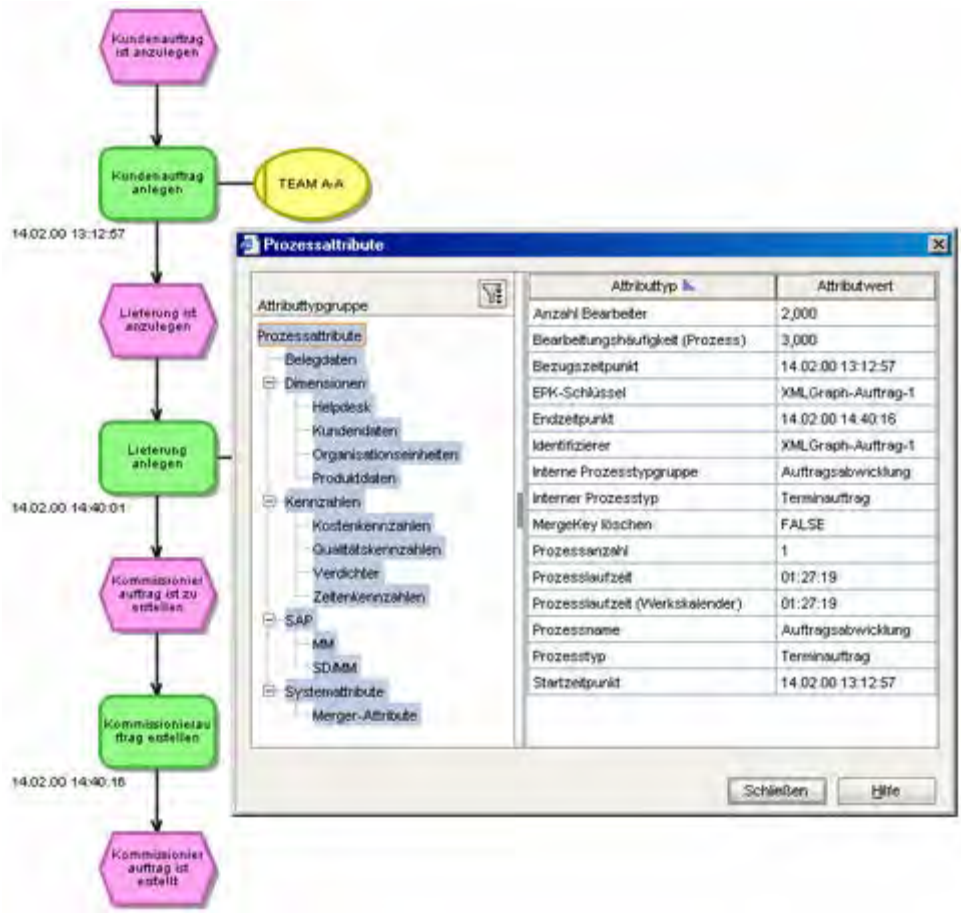
OPTIONALE OBJEKTATTRIBUTE

Folgende Tabelle zeigt beispielhaft einige Objektattribute, die als Basis für die Berechnung von Kennzahlen verwendet werden können.

Attributname	Identifizierer	Typ	Beschreibung
Anzahl Bearbeitungen	AT_COUNT_PROCESSINGS	LONG	Gibt die Ausführungshäufigkeit einer Funktion an.
Leistungs-standard	AT_LS	TIME-SPAN	Wird verwendet, um die durchschnittliche Ausführungsdauer einer Instanz zu berechnen.
Batch User	AT_IS_BATCH USER	BOOLEAN	Gibt an, ob eine Organisationseinheit ein Batch User (Programm) ist.

4.1.6 XML-Beispielgraph

Die folgende Abbildung zeigt einen Graphen mit erzeugten Prozessinstanzattributen (berechnete Kennzahlen und Prozesstyp) als EPK-Ansicht mit aktivem Prozessattributdialog:



Die zugehörige XML-Datei sieht folgendermaßen aus (die Zeilen der Objektdarstellung sind den EPK-Objekten entsprechend farbig hinterlegt):

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE graphlist SYSTEM "graph.dtd">
<graphlist>
  <graph id="XMLGraph-Auftrag-1" xml:lang="de">
    <attribute type="AT_EPK_KEY">XMLGraph-Auftrag-1</attribute>
    <attribute type="AT_TIME">14.2.2000 13:12:57</attribute>
    <attribute type="AT_PROCTYPE">Terminauftrag</attribute>
    <attribute type="AT_PROCTYPEGROUP">Auftragsabwicklung</attribute>
    <attribute type="AT_ID">XMLGraph-Auftrag-1</attribute>
    <node id="XMLGraph-Auftrag-Evt1" type="OT_EVT">
      <attribute type="AT_OBJNAME_INTERN">AUFTRAG_ANZU</attribute>
      <attribute type="AT_OBJNAME">Kundenauftrag ist anzulegen</attribute>
    </node>
    <node id="XMLGraph-Auftrag-Func1" type="OT_FUNC">
    <attribute type="AT_TIME">14.2.2000 13:12:57</attribute>
      <attribute type="AT_OBJNAME_INTERN">AUFTRAG</attribute>
      <attribute type="AT_OBJNAME">Kundenauftrag anlegen</attribute>
      <attribute type="AT_END_TIME">14.02.2000 13:12:57</attribute>
    </node>
    <node id="HDMXMLGraph-Auftrag-Func1" type="OT_ORG">
```

```
    <attribute type="AT_OBJNAME">HDM</attribute>
  </node>
  <node id="XMLGraph-Auftrag-Evt2" type="OT_EVT">
    <attribute type="AT_OBJNAME_INTERN">LIEFERUNG_ANZU</attribute>
    <attribute type="AT_OBJNAME">Lieferung ist anzulegen</attribute>
    <attribute type="AT_ID">XMLGraph-Auftrag-Evt2</attribute>
  </node>
  <node id="XMLGraph-Auftrag-Func2" type="OT_FUNC">
  <attribute type="AT_TIME">14.2.2000 14:40:01</attribute>
    <attribute type="AT_OBJNAME_INTERN">LIEFERUNG</attribute>
    <attribute type="AT_OBJNAME">Lieferung anlegen</attribute>
    <attribute type="AT_END_TIME">14.02.2000 14:40:01</attribute>
  </node>
  <node id="HDMXMLGraph-Auftrag-Func2" type="OT_ORG">
    <attribute type="AT_OBJNAME">HDM</attribute>
  </node>
  <node id="XMLGraph-Auftrag-Evt3" type="OT_EVT">
    <attribute type="AT_OBJNAME_INTERN">KOM_AUFTRAG_ANZU</attribute>
    <attribute type="AT_OBJNAME">Kommissionierauftrag ist zu
                                              erstellen</attribute>
    <attribute type="AT_ID">XMLGraph-Auftrag-Evt3</attribute>
  </node>
  <node id="XMLGraph-Auftrag-Func3" type="OT_FUNC">
    <attribute type="AT_TIME">14.2.2000 14:40:16</attribute>
    <attribute type="AT_OBJNAME_INTERN">KOM_AUFTRAG</attribute>
    <attribute type="AT_OBJNAME">Kommissionierauftrag erstellen</attribute>
    <attribute type="AT_END_TIME">14.02.2000 14:40:16</attribute>
  </node>
  <node id="XMLGraph-Auftrag-Evt4" type="OT_EVT">
    <attribute type="AT_OBJNAME_INTERN">KOM_AUFTRAG_ALGT</attribute>
    <attribute type="AT_OBJNAME">Kommissionierauftrag ist erstellt</attribute>
  </node>
  <edge type="CXN_FOLLOWS" source="XMLGraph-Auftrag-Evt1"
                       target="XMLGraph-Auftrag-Func1" />
  <edge type="CXN_FOLLOWS" source="XMLGraph-Auftrag-Func1"
                       target="XMLGraph-Auftrag-Evt2" />
  <edge type="CXN_UNDIRECTED" source="HDMXMLGraph-Auftrag-Func1"
                         target="XMLGraph-Auftrag-Func1">
    <attribute type="AT_COUNT_PROCESSINGS">1</attribute>
  </edge>
  <edge type="CXN_FOLLOWS" source="XMLGraph-Auftrag-Func2"
                       target="XMLGraph-Auftrag-Evt3" />
  <edge type="CXN_UNDIRECTED" source="HDMXMLGraph-Auftrag-Func2"
                         target="XMLGraph-Auftrag-Func2">
    <attribute type="AT_COUNT_PROCESSINGS">1</attribute>
  </edge>
  <edge type="CXN_FOLLOWS" source="XMLGraph-Auftrag-Func3"
                       target="XMLGraph-Auftrag-Evt4" />
  <edge type="CXN_FOLLOWS" source="XMLGraph-Auftrag-Evt2"
                       target="XMLGraph-Auftrag-Func2" />
  <edge type="CXN_FOLLOWS" source="XMLGraph-Auftrag-Evt3"
                       target="XMLGraph-Auftrag-Func3" />
 </graph>
</graphlist>
```

4.2 System-Event-Format

Viele Quellsysteme protokollieren die Bearbeitung eines Vorgangs durch Dokumentation von Statuswechseln oder bestimmten Systemzuständen. Diese werden in Form von System-Events (Quellsystemereignisse) mit ergänzenden Informationen gespeichert. Ein typisches Beispiel ist ein System zur Auftragsbearbeitung, bei dem das Anlegen eines neuen Auftrags oder die Fakturierung eines Auftrags gespeichert wird. Ein weiteres Beispiel ist das SAP R/3 SD-Modul, das die Zustände und Bearbeitungsfortschritte eines Auftrages in einzelnen Transaktionen dokumentiert.

Das Importieren von Prozessinstanzfragmenten im System-Event-Format des PPM-Systems erfolgt in drei Schritten:

1. Prozessfragmente und Mapping definieren

   Alle in den Quellsystemdaten vorkommenden System-Event-Typen, die nach PPM importiert werden sollen (z. B. Auftrag anlegen, Auftrag fakturieren), müssen zuvor in der Fragmentdatei als Prozessfragmentdefinition in Form einer EPK angelegt werden. Zusätzlich muss in der Mapping-Datei festgelegt werden, welche System-Event-Typen beim Import welchen Fragmentdefinitionen zugeordnet werden sollen und welche Attribute nach PPM übernommen werden sollen. In der Mapping-Datei müssen Sie Attribute berücksichtigen, die das Berechnen der Kennzahlen als auch das Zusammenführen der Prozessinstanzfragmente zu einer Prozessinstanz ermöglichen (z. B. Zeitstempel, lfd. System-Event-Nummer, Auftragsnummer).

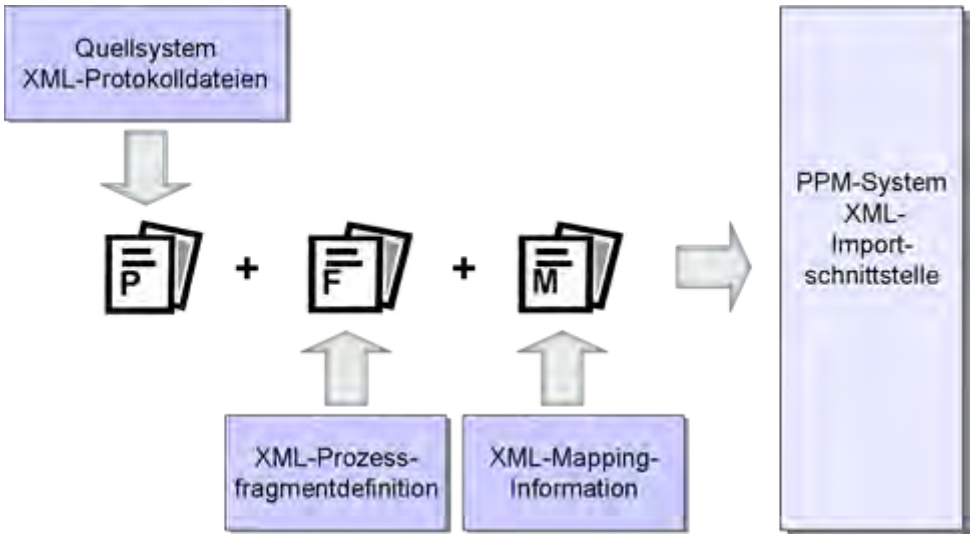
2. Generieren einer XML-Ausgabedatei des Quellsystems

   Aus dem Anwendungssystem werden einzelne System-Events mit zusätzlichen Informationen (Attribute mit Realdaten) dem Aktivitätenfluss entsprechend in eine XML-Ausgabedatei ausgelesen.

3. Generieren der Prozessinstanzfragmente

   Beim Datenimport wird für jedes System-Event der XML-Ausgabedatei die zugeordnete Fragmentdefinition (Typebene) gesucht und in die PPM-Datenbank kopiert. An die Objekte der Fragmentdefinitionskopie werden Attribute des System-Event mit Realdaten (z. B. Ausführungszeitpunkt, Bearbeiter, Auftrags- und Kundennummer) übertragen und so ein diesem System-Event entsprechendes Prozessinstanzfragment (Ausprägungsebene) in der PPM-Datenbank erzeugt.

Die folgende Abbildung verdeutlicht den Ablauf beim Generieren der Prozessinstanzfragmente:



4.2.1 Definition der Prozessfragmente

Um aus den System-Events Prozessinstanzen für das PPM-System generieren zu können, muss jedes System-Event, das nach PPM importiert werden soll, mit einer Prozessfragmentdefinition verknüpft werden. Jedes System-Event ist einem System-Event-Typ zugeordnet. Für jeden System-Event-Typ muss eine Fragmentdefinition erstellt werden.

Jeder System-Event-Typ wird einem Endereignis in einer eigenen EPK zugeordnet. Es wird angenommen, dass dem Endereignis im Prozess eine Funktion vorausgegangen sein muss, die dieses Ereignis ausgelöst hat. Durch Hinzufügen eines Startereignisses vor die Funktion entsteht ein vollständiges modellierungskonformes Prozessfragment. Das Startereignis eines Prozessfragmentes kann dem Endereignis eines anderen Prozessfragmentes entsprechen.
Alle Prozessfragmentdefinitionen werden als Graphliste in einer XML-Datei gespeichert. Die XML-Datei zur Fragmentdefinition verwendet zur Formatbeschreibung die DTD des Graphformates.

Das Verketten der eingelesenen Prozessinstanzfragmente zu einer Prozessinstanz geschieht über die Start- und Endereignisse der Prozessinstanzfragmente. Diese Ereignisse heißen Merge-Ereignisse.

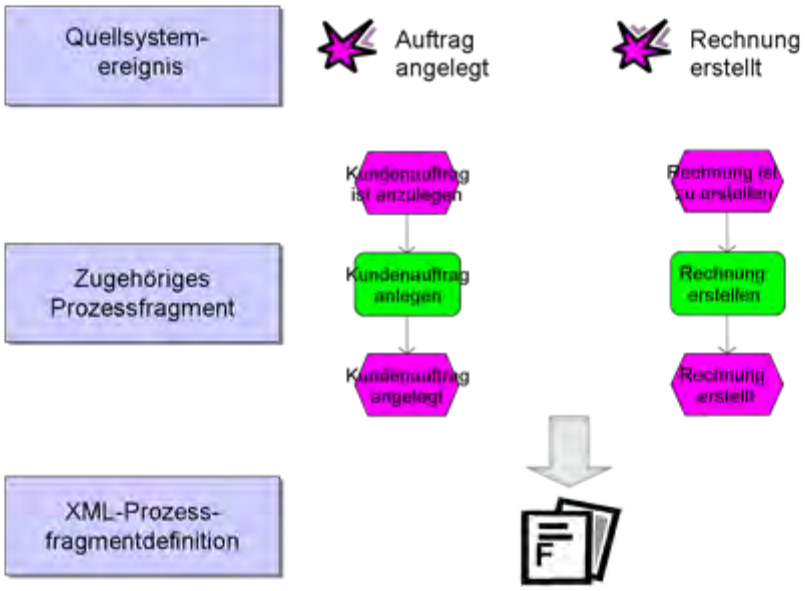
Es ist nicht zwingend erforderlich, dass das System-Event und das Endereignis des Prozessinstanzfragments übereinstimmen, z. B. wenn der Prozessablauf der Fragmentdefinition mit zwei Endereignissen endet. Wichtig ist, dass alle im System-Event enthaltenen Systembeschreibungen durch das erzeugte Prozessinstanzfragment abgebildet sind.

Beispiel

Das Quellsystem enthält die beiden System-Events **Auftrag angelegt** und **Rechnung erstellt**. Diese werden jeweils als Endereignis in eine EPK übernommen und als Ergebnis der Funktion **Kundenauftrag anlegen** bzw. **Rechnung erstellen** interpretiert. Durch weiteres Wissen über das Quellsystem müssen Sie die Ereignisse bestimmen, die diesen beiden Funktionen

vorausgegangen sind. Im Beispiel sind dies die Ereignisse **Kundenauftrag ist anzulegen** und **Rechnung ist zu erstellen**.

Die folgende Abbildung zeigt die Prozessfragmentbeschreibung zu den beiden System-Events.



Der folgende Dateiauszug zeigt eine mögliche Fragmentdefinitionsdatei zu den gezeigten Prozessfragmenten:

```
<?xml version="1.0" encoding='ISO-8859-1'?>
<!DOCTYPE graphlist SYSTEM "graph.dtd">
<graphlist>
  <graph id="FRG_ORD_CREATED">
    <node id="EVT_ORD_TOBECREATED" type="OT_EVT">
      <attribute type="AT_OBJNAME">Kundenauftrag ist anzulegen</attribute>
      <attribute type="AT_OBJNAME_INTERN">EVT_ORD_TOBECREATED</attribute>
    </node>
    <node id="FCT_CREATE_ORDER" type="OT_FUNC">
      <attribute type="AT_OBJNAME">Kundenauftrag anlegen</attribute>
      <attribute type="AT_OBJNAME_INTERN">FCT_CREATE_ORDER</attribute>
    </node>
    <node id="EVT_ORD_CREATED" type="OT_EVT">
      <attribute type="AT_OBJNAME">Kundenauftrag angelegt</attribute>
      <attribute type="AT_OBJNAME_INTERN">EVT_ORD_CREATED</attribute>
    </node>
    <edge type="CXN_FOLLOWS" source="EVT_ORD_TOBECREATED"
                          target="FCT_CREATE_ORDER"/>
    <edge type="CXN_FOLLOWS" source="FCT_CREATE_ORDER"
                          target="EVT_ORD_CREATED"/>
  </graph>
  <graph id="FRG_INVOICED">
    <node id="EVT_TOBE_INVOICED" type="OT_EVT">
      <attribute type="AT_OBJNAME">Rechnung ist zu erstellen</attribute>
      <attribute type="AT_OBJNAME_INTERN">EVT_TOBE_INVOICED</attribute>
    </node>
    <node id="FCT_INVOICE" type="OT_FUNC">
      <attribute type="AT_OBJNAME">Rechnung erstellen</attribute>
```

```
      <attribute type="AT_OBJNAME_INTERN">FCT_INVOICE</attribute>
    </node>
    <node id="EVT_INVOICED" type="OT_EVT">
      <attribute type="AT_OBJNAME">Rechnung erstellt</attribute>
      <attribute type="AT_OBJNAME_INTERN">EVT_INVOICED</attribute>
    </node>
    <edge type="CXN_FOLLOWS" source="EVT_TOBE_INVOICED"
                          target="FCT_INVOICE"/>
    <edge type="CXN_FOLLOWS" source="FCT_INVOICE"
                          target="EVT_INVOICED"/>
  </graph>
</graphlist>
```

Fragmentdefinitionsgraphen sollten keine gepflegten Attribute enthalten. Beim späteren Zusammenführen der Prozessinstanzfragmente (Merge) werden standardmäßig ausschließlich Objektattribute berücksichtigt. Prozessinstanzattribute, die beim Merge erhalten bleiben sollen, können Sie in der Konfiguration des Merger angeben.

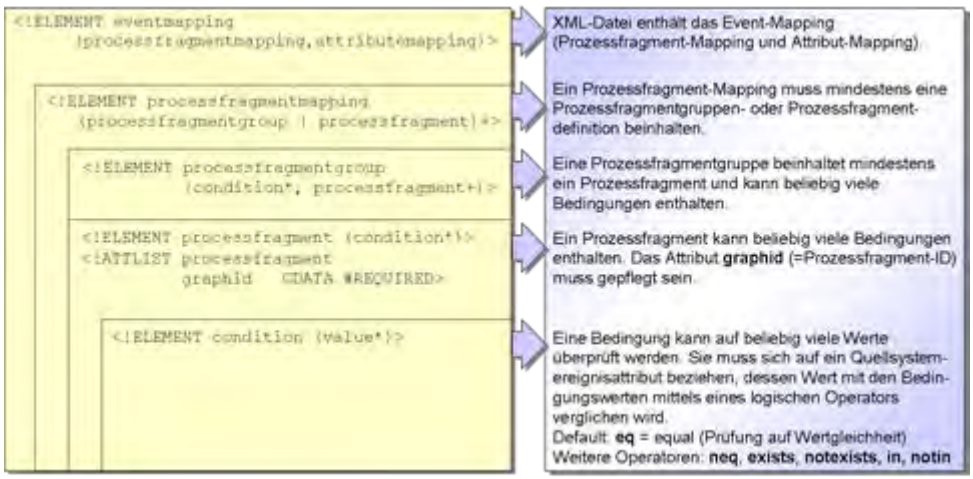
4.2.2 Definition des Mapping

Die Mapping-Datei enthält die Zuordnung der System-Event-Typen zu Prozessfragmentdefinitionen und bestimmt die Attribute der System-Events, die an die instanziierten Prozessfragmente des PPM-Systems kopiert werden.

4.2.2.1 Definition des Prozessfragment-Mapping

Das Prozessfragment-Mapping definiert, welche Prozessfragmentdefinitionen zum Instanziieren der System-Event-Typen verwendet werden. Es kann durch beliebig viele, miteinander UND-verknüpfte Bedingungen (XML-Element **condition**) gesteuert werden.

Die Regeln für den Aufbau des Prozessfragment-Mapping in der XML- Mapping-Datei sind in folgendem Auszug der Datei **eventmapping.dtd** festgelegt:



Beispiel für die bedingte Prozessfragmenterzeugung (Dateiauszug):

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

```
  <processfragmentmapping>
    <processfragment graphid="FRG_ORD_CREATED">
       <condition eventattributetype="AUFTR_TYP">
         <value>C</value>
       </condition>
       <condition eventattributetype="MAT_NR" logicaloperator="in">
         <value>123456</value>
         <value>56789</value>
         <value>78901</value>
       </condition>
  </processfragment>
    ...
  </processfragmentmapping>
  <attributemapping>
    ...
  </attributemapping>
</eventmapping>
```

Das Prozessfragment **FRG_ORD_CREATED** wird erzeugt, wenn die beiden folgenden Bedingungen erfüllt sind:

- Das betreffende System-Event stellt einen Auftragsbeleg dar (Attribut **AUFTR_TYP** hat den Wert **C**).
- Der Wert des System-Event-Attributs **MAT_NR** stimmt mit einer der angegebenen Materialnummern überein.

Eine Prozessfragmentdefinition muss keine Bedingungen enthalten. In diesem Fall wird für jedes importierte System-Event dieselbe angegebene Fragmentdefinition verwendet. Die Objekte des instanziierten Fragments werden beim anschließenden Attribut-Mapping spezifiziert.

DEFINITION EINER PROZESSFRAGMENTGRUPPE

Prozessfragmentdefinitionen können zu Gruppen zusammengefasst werden. Daraus ergeben sich folgende Vorteile:

- Performance-Steigerung
- Vereinfachte Erstellung der Prozessfragment-Mapping-Definitionen
- Verbesserte Übersichtlichkeit

Die folgenden beiden Prozessfragmentdefinitionen sind in einer XML-Mapping-Datei enthalten:

```
<processfragment graphid="AUFTRAG_ANLEGEN">
  <condition eventattributetype="AUFTR_TYP">
    <value>C</value>
  </condition>
  <condition eventattributetype="CHARGEN_PFL" logicaloperator="neq">
    <value>X</value>
  </condition>
</processfragment>
<processfragment graphid="CHPLICHT_AUFTRAG_ANLEGEN">
  <condition eventattributetype="AUFTR_TYP">
    <value>C</value>
  </condition>
  <condition eventattributetype="CHARGEN_PFL">
    <value>X</value>
  </condition>
</processfragment>
```

Das erste Prozessfragment **AUFTRAG_ANLEGEN** wird für ein System-Event des Typs **Auftragsbeleg** erzeugt (Attributwert **AUFTR_TYP** ist gleich **C**), das nicht chargenpflichtig ist (Attributwert **CHARGEN_PFL** ist ungleich **X**).

Das zweite Prozessfragment **CHPLICHT_AUFTRAG_ANLEGEN** wird für ein System-Event des Typs **Chargenpflichtiger Auftragsbeleg** erzeugt (Attributwert **AUFTR_TYP** ist gleich **C** und Attributwert **CHARGEN_PFL** ist gleich **X**).

Die beiden gezeigten Prozessfragmente lassen sich in einer Prozessfragmentgruppe zusammenfassen:

```
<processfragmentgroup>
  <condition eventattributetype="AUFTR_TYP">
    <value>C</value>
  </condition>

  <processfragment graphid="AUFTRAG_ANLEGEN">
    <condition eventattributetype="CHARGEN_PFL" logicaloperator="neq">
      <value>X</value>
    </condition>
  </processfragment>

  <processfragment graphid="CHPLICHT_AUFTRAG_ANLEGEN">
    <condition eventattributetype="CHARGEN_PFL">
      <value>X</value>
    </condition>
  </processfragment>

</processfragmentgroup>
```

Durch Zusammenfassen in einer Prozessfragmentgruppe muss beim Import nur einmal geprüft werden, ob das Quellsystemattribut **AUFTR_TYP** den Wert **C** hat. Ist dies nicht der Fall, werden beide Prozessfragmente der Prozessfragmentgruppe nicht instanziiert.

Verwenden Sie Prozessfragmentgruppen, um die Übersichtlichkeit von Prozessfragment-Mapping-Definitionen und die Performance des Imports zu verbessern.

AUSGABE VON WARNUNGEN UNTERDRÜCKEN

Wenn Sie in Ihrem Customizing bestimmte System-Events nicht importieren möchten und für diese System-Events auch kein Prozess-Mapping definiert haben, können Sie die zu erwartende Fehlermeldung beim Import unterdrücken. Hierfür geben Sie im XML-Element **ignoreevent** Bedingungen an, die die Ausgabe einer Fehlermeldung für bestimmte Fragmente unterdrücken, wenn diese nicht importiert werden können.

Für mit **ignoreevent**-Mapping spezifizierte System-Events wird lediglich die Ausgabe der Warnung unterdrückt, wenn das System-Event nicht importiert werden kann. Das heißt, wenn Sie System-Events importieren, für die sowohl ein Prozessfragment-Mapping als auch ein **ignoreevent**-Mapping zutreffen, werden diese System-Events importiert.

Beispiel

Sie möchten Prozessfragmente importieren, wenn das System-Event-Attribut **EKKO_BSTYP** existiert und das System-Event-Attribut **MSEG_SHKZG** den Wert **S** für „Wareneingang buchen“ oder **H** für „Wareneingang stornieren“ hat. Andere Werte des Attributes **MSEG_SHKZG** führen zur Ausgabe einer Warnung.

Ist das System-Event-Attribut **EKKO_BSTYP** nicht vorhanden, soll kein Prozessfragment importiert und keine Warnung ausgegeben werden.

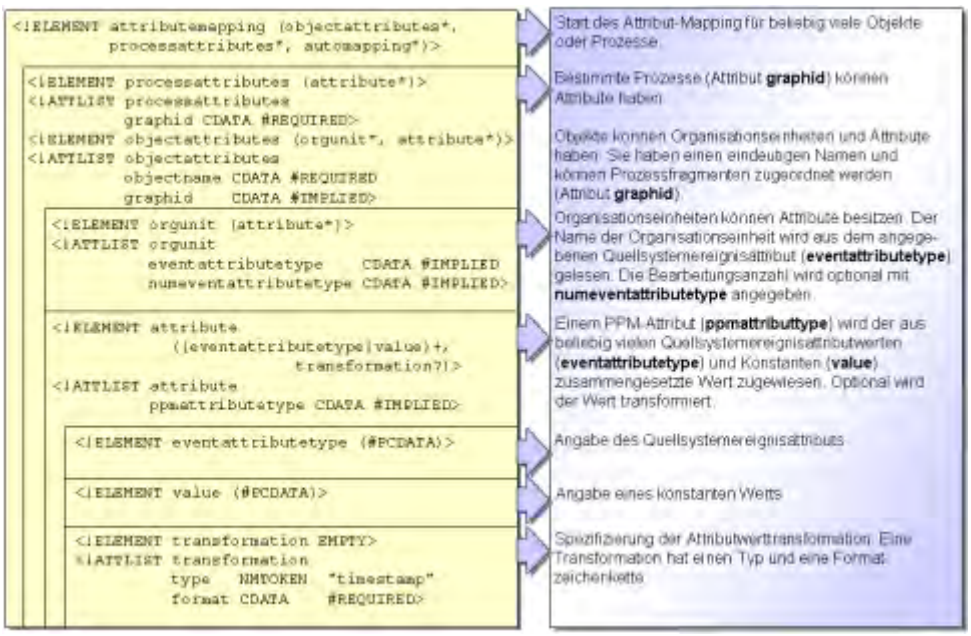
Folgendes Prozess-Mapping erfüllt die genannten Bedingungen:

```
...
<processfragmentgroup>
  <!-- nur Wareneingänge mit MM-Vorgängerbelegen importieren
  -->
  <condition eventattributetype="EKKO-BSTYP" logicaloperator="exists"/>
  <processfragment graphid="GWEOF">
    <!-- Wareneingang buchen
      -->
    <condition eventattributetype="MSEG-SHKZG" logicaloperator="eq">
      <value>S</value>
    </condition>
  </processfragment>

  <processfragment graphid="GWSOF">
  <!-- Wareneingang stornieren
  -->
    <condition eventattributetype="MSEG-SHKZG" logicaloperator="eq">
      <value>H</value>
    </condition>
  </processfragment>
</processfragmentgroup>

<ignoreevent>
  <!-- Keine Warnung ausgeben, wenn kein MM-Vorgängerbeleg vorhanden,
       da diese Systemevents nicht importiert werden sollen
  -->
  <condition eventattributetype="EKKO-BSTYP" logicaloperator="notexists"/>
</ignoreevent>
...
```

4.2.2.2 Definition des Attribut-Mapping

Dieses Kapitel beschreibt die Konfiguration des Attribut-Mapping. Das Attribut-Mapping kopiert Quellsystemattribute an Objekt- und Prozessattribute der Fragmentinstanz (PPM-Attribute).

Die Regeln für den Aufbau des Attribut-Mapping in der XML-Mapping-Datei sind in folgendem Auszug der Datei **eventmapping.dtd** festgelegt:



PPM-Attributwerte können aus einer beliebigen Kombination von Attributwerten des System-Event und unveränderlichen Texten zusammengesetzt werden. Durch optionale Angabe des internen PPM-Attributnamens können Quellsystemattribute auf beliebige PPM-Attribute kopiert werden. PPM-Attribute können mit konstanten Werten (XML-Element **value)** belegt werden.

Beispiel 1 (Standard-Mapping)

```
...
  <attribute>
    <eventattributetype>MATERIAL_CLASS</eventattributetype>
  </attribute>
...
```

Der Wert des System-Event-Attributs **MATERIAL_CLASS** wird in das gleichnamige, um das in der Datenquelle im XML-Attribut **attributeprefix** angegebene Präfix (z. B. **AT_**) erweiterte PPM-Attribut **AT_MATERIAL_CLASS** kopiert.

Beispiel 2 (Explizites Mapping)

```
...
  <attribute ppmattributetype="AT_MATERIAL">
    <eventattributetype>MAT_NR</eventattributetype>
  </attribute>
...
```

Der Wert des System-Event-Attributs **MAT_NR** wird an das PPM-Attribut **AT_MATERIAL** kopiert.

Beispiel 3

```
...
  <attribute ppmattributetype="AT_IS_SHARED_FUNCTION">
    <value>TRUE</value>
  </attribute>
...
```

Dem PPM-Attribut **AT_IS_SHARED_FUNCTION** wird die Konstante **TRUE** zugewiesen.

Beispiel 4

```
...
  <attribute ppmattributetype="AT_ID">
    <eventattributetype>AUFTRAGS_SYSTEM</eventattributetype>
    <value>-</value>
    <eventattributetype>SYSTEM_NR</eventattributetype>
    <value>#</value>
    <eventattributetype>AUFTRAGS_NR</eventattributetype>
    <value>-</value>
    <eventattributetype>POSITIONS_NR</eventattributetype>
    <value>-</value>
    <eventattributetype>AUFTRAGS_TYP</eventattributetype>
  </attribute>
...
```

Dem PPM-Attribut **AT_ID** wird der Wert **XYZ-401#4711-10-C** zugewiesen.

Warnung

Ist beim Datenimport aus der XML-Ausgabedatei ein angegebenes Quellsystemattribut nicht vorhanden, wird das entsprechende PPM-Attribut nicht erzeugt.

ATTRIBUTE OBJEKTEN ZUORDNEN

Erzeugte PPM-Attribute werden über das XML-Element **objectattributes** bestimmten Objekten der Fragmentinstanz zugeordnet. Im XML-Attribut **objname** wird der Identifizierer des gewünschten Objekts angegeben (Objektattribut **AT_OBJNAME_INTERN**). Optional lässt sich die Objektspezifizierung durch Angabe der Graph-ID (XML-Attribut **graphid**) verfeinern.

Das folgende Beispiel kopiert den Wert des Quellsystemattributs **END_TIME** an das PPM-Objektattribut **AT_END_TIME** der Funktion mit dem Identifizierer **SAP.AUFT_ANLEG** der Fragmentinstanz **AUFTRAG_ANLEGEN** (Graph-ID der Fragmentdefinition):

```
...
<attributemapping>
  <objectattributes objectname="SAP.AUFT_ANLEG" graphid="AUFTRAG_ANLEGEN">
    <attribute ppmattributetype="AT_END_TIME">
      <eventattributetype>END_TIME</eventattributetype>
    </attribute>
  </objectattributes>
</attributemapping>
...
```

ATTRIBUTE PROZESSEN ZUORDNEN

Erzeugte PPM-Attribute werden über das XML-Element **processattributes** bestimmten Fragmentinstanzen zugeordnet. Obligatorisch wird im XML-Attribut **graphid** die ID des Fragmentdefinitionsgraphen angegeben.

Damit direkt an den Prozess übertragene Attribute beim Zusammenführen der Fragmentinstanzen erhalten bleiben, müssen Sie die Konfiguration des Merger entsprechend erweitern. Der zum Abgleich der Prozessattribute erforderliche Rechenaufwand bedingt einen geringen Performance-Verlust.

Das folgende Beispiel kopiert den Wert des Quellsystemattributs **PROCESSNAME** an das PPM-Prozessattribut **AT_PROCTYPE** der Fragmentinstanz **AUFTRAG_ANLEGEN** (Graph-ID der Fragmentdefinition):

```
...
<attributemapping>
  <processattributes graphid="AUFTRAG_ANLEGEN">
    <attribute ppmattributetype="AT_PROCTYPE">
      <eventattributetype>PROCESSNAME</eventattributetype>
    </attribute>
  </processattributes>
</attributemapping>
...
```

4.2.2.2.1 Attributtransformationen

Wenn das Format eines Attributwertes eines System-Event nicht mit dem vom PPM-System benötigten Format übereinstimmt, muss der in das PPM-Attribut geschriebene Wert transformiert werden.

4.2.2.2.1.1 Zeitstempeltransformationen

In PPM stehen verschiedene Attributtransformationen vom Typ Zeitstempeltransformation zur Verfügung.

Die im PPM-System verfügbaren Zeitstempeltransformationen wandeln Quellsystem-Zeitstempelwerte, die in beliebigen Formaten vorliegen, in das zulässige, interne PPM-Format mit dem korrekten PPM-Zieldatentyp um. Die folgende Tabelle gibt einen Überblick über die verfügbaren Zeitstempeltransformationen:

Transformation	PPM-Zieldatentyp
timestamp	TIME
timestamp_epoch	TIME
timeofday	TIMEOFDAY
timeofday_epoch	TIMEOFDAY
day	DAY

Transformation	PPM-Zieldatentyp
day_epoch	DAY
SAGDateTime	TIME

ZEITSTEMPELTRANSFORMATION TIMESTAMP, TIMEOFDAY, DAY

Die Vorgaben bezüglich der Konfiguration von Zeitstempeltransformationen sind in der Dokumenttypdefinition **eventmapping.dtd** festgelegt:

```
...
<!ELEMENT transformation EMPTY>
<!ATTLIST transformation
     type NMTOKEN "timestamp"
     format CDATA #REQUIRED
>
...
```

Im XML-Attribut **type** wird angegeben, welche der möglichen Zeitstempeltransformationen verwendet werden soll. Dabei ist die Attributtransformation **timestamp** voreingestellt. Im XML-Attribut **format** wird angegeben, in welchem Format der Zeitstempel im Quellsystemattribut vorliegt.

Die Angabe im Attribut **format** entspricht dem Zeitstempelformat von Java:

Symbol	Beschreibung	Datentyp	Beispiel
G	Epochenbezeichner	Text	AD
y	Jahr	Zahl	2003
MM	Kalendermonat	Zahl	09
MMM	Kalendermonat	Text	Sep
MMMM	Kalendermonat	Text	September
d	Kalendertag	Zahl	26
h	Stunde, amerikanische Notation (1-12)	Zahl	12
H	Stunde (0-23)	Zahl	14
m	Minute	Zahl	42
s	Sekunde	Zahl	57
S	Millisekunde	Zahl	978
E	Wochentag	Text	Donnerstag
F	Wochentagswiederholung im Monat	Zahl	2
D	Tag im Jahr	Zahl	189

Symbol	Beschreibung	Datentyp	Beispiel
w	Kalenderwoche	Zahl	27
W	Woche im Monat	Zahl	2 (2. Woche im Monat)
a	Tageszeitbezeichner	Text	PM
k	Stunde (1-24)	Zahl	24
K	Stunde, amerikanische Notation (0-11)	Zahl	7
z	Zeitzone	Text	GMT
´	Escape-Zeichen für Text	Zeichen	´Beispieltext´
´´	einfaches Hochkomma	Zeichen	´Beispiel´´Text´

Beispiel 1

Der Wert **2002-12-24** (<Jahr>-<Monat>-<Tag>) eines Quellsystemattributs wird durch die Formatzeichenkette yyyy-MM-dd in das PPM-Format transformiert:

```
...
  <transformation type="timestamp" format="yyyy-MM-dd"/>
...
```

Beispiel 2

Zusammengesetzte Quellsystemattributwerte:

```
...
  <attribute ppmattributetype="AT_END_TIME">
    <eventattributetype>ERF_DAT</eventattributetype>
    <eventattributetype>ERF_ZEIT</eventattributetype>
    <transformation format="yyyyMMddHHmmss"/>
  </attribute>
...
```

Im Quellsystemattribut **ERF_DAT** liegt das Erfassungsdatum im Format **yyyyMMdd** und im Quellsystemattribut **ERF_ZEIT** die Erfassungszeit im Format **HHmmss** vor. Auszug der XML-Ausgabedatei:

```
<event>
  ...
  <attribute type='ERF_DAT'>20011230</attribute>
  ...
</event>
<event>
  ...
  <attribute type='ERF_ZEIT'>120730</attribute>
  <attribute type='ERF_DAT'>20011101</attribute>
  ...
</event>
```

Für das erste System-Event wird das Attribut **AT_END_TIME** nicht erzeugt, da das Attribut **ERF_ZEIT** nicht vorhanden ist.

Beim zweiten System-Event sind beide Attribute vorhanden. Die Attributwerte werden in der angegebenen Reihenfolge zu **20011101120730** zusammengefügt, anschließend mit dem angegebenen Format **yyyyMMddHHmmss** ausgewertet und in den PPM-konformen Zeitstempel **01.11.2001 12:07:30** umgewandelt.

Beispiel 3

Im Attribut **ERF_STD** wird nur die Stunde erfasst, in der das System-Event erzeugt wurde. Der Erfassungszeitpunkt liegt dabei immer dreißig Minuten nach der vollen Stunde.

```
...
  <attribute ppmattributetype="AT_END_TIME">
    <eventattributetype>ERF_DAT</eventattributetype>
    <value>::</value>
    <eventattributetype>ERF_STD</eventattributetype>
    <value>30</value>
    <transformation format="yyyyMMdd::HHmm"/>
  </attribute>
  ...
```

Zugehöriges System-Event der Ausgabedatei:

```
...
<event>
  ...
  <attribute type='ERF_STD'>12</attribute>
  <attribute type='ERF_DAT'>20011001</attribute>
  ...
</event>
...
```

Die Attributwerte und konstanten Zeichenketten werden in der angegebenen Reihenfolge zur Zeichenkette **20011001::1230** zusammengesetzt und mit dem Format **yyyyMMdd::hhmm** in den PPM-konformen Zeitstempel **01.10.2001 12:30:00** umgewandelt, der an das PPM-Attribut **AT_END_TIME** geschrieben wird.

ZEITSTEMPELTRANSFORMATIONEN TIMESTAMP_EPOCH, TIMEOFDAY_EPOCH UND DAY_EPOCH

Die Zeitstempeltransformation **timestamp_epoch** wandelt einen Ganzzahlenwert, der die seit dem 1. Januar 1970 vergangenen Sekunden bzw. Millisekunden angibt, in das interne PPM-Format um. Im XML-Attribut **format** bestimmen Sie, ob der Ganzzahlenwert die Anzahl der Sekunden (Wert **SECOND**) oder Millisekunden (Wert **MILLISECOND**) seit dem 01.01.1970 0:00:00 GMT angibt. Bei der Berechnung wird die aktuelle Zeitzone des Systems berücksichtigt.

Die Attributtransformationen **timeofday_epoch** und **day_epoch** sind analog zu verwenden.

Beispiel (timestamp_epoch)

Im Quellsystemattribut **WORK_ITEM-END_TIME** ist die Anzahl der Sekunden seit dem 1. 1. 1970 angegeben.

```
...
```

```
  <attribute type="WORK_ITEM-END_TIME">1221482578</attribute>
...
```

Um diesen Attributwert dem Wert des PPM-Attributs **AT_END_TIME** zuzuweisen, können Sie folgende Mapping-Regel verwenden:

```
...
  <attribute ppmattributetype="AT_END_TIME">
    <eventattributetype>
      WORK_ITEM-END_TIME
    </eventattributetype>
    <transformation type="timestamp_epoch"
                     format="SECOND"/>
  </attribute>
...
```

ATTRIBUTTRANSFORMATION SAGDATETIME

Das Zeitformat **SAGDateTime** wird in sogenannten EDA Events verwendet. Event-Attribute vom Typ **SAGDateTime** werden bislang als Textattribute importiert, da das Datumsformat nicht ohne Weiteres von den bestehenden Zeitstempeltransformationen verarbeitet werden kann.

- Das Zeitformat **SAGDateTime** basiert auf dem "W3C date and time format standard". Das Format setzt sich zusammen aus dem Datum, Angaben für Stunden, Minuten, Sekunden sowie einer optionalen Anzahl von Millisekunden und einer Zeitzonenangabe.

  YYYY-MM-DDThh:mm:ss.sssTZD

  Beispiele für Zeitangaben in diesem Format sind:

  2011-04-28T08:15:59.001Z

  2011-04-28T08:15:59.001+06:00
- Die Zeitzonenangabe (TZD) kann entweder „Z" sein für UTC oder "+hh:mm" oder "-hh:mm", um ein Offset zur Zeitzone UTC anzugeben.

TRANSFORMATORKLASSE

Die Transformation **SAGDateTime** enthält die Funktionalität zum Transformieren von Zeitangaben im Format **SAGDateTime** in das PPM-interne Zeitformat.

Die Klasse kann im Attribut-Mapping im Rahmen des PPM-XML- und Process-Import verwendet werden. Dazu muss eine entsprechende Mapping-Regel definiert werden, die wie folgt aussehen könnte.

```
<attribute ppmattributetype="AT_TIMESTAMP_PPM">
    <eventattributetype>TIMESTAMP_SAGDATETIME</eventattributetype>
    <transformation type="SAGDateTime" format="yyyy-MM-dd'T'HH:mm:ss.SSSZ"/>
</attribute>
```

Die Anwendung dieser Regel führt dazu, dass der Inhalt des Event-Attributs **TIMESTAMP_SAGDATETIME** in einem PPM-kompatiblen Format an das PPM-Attribut **AT_TIMESTAMP_PPM** geschrieben wird. Dabei bleibt die Millisekundenangabe unberücksichtigt.

Die Anwendung der obigen Regel auf das Event-Attribut

```
<attribute type="TIMESTAMP_SAGDATETIME">2013-07-15T06:02:33.650Z</attribute>
```

würde in der Zeitzone **UTC** beispielsweise zur Erzeugung des folgenden PPM-Attributs führen:

```
<attribute type="AT_TIMESTAMP_PPM">15.07.2013 06:02:33</attribute>
```

Bei der Spezifikation der Mapping-Regel haben Sie zwei Optionen.

1. Angabe des erwarteten Formats des Event-Attributes

   Wenn bekannt ist, ob die Zeitangabe im Event-Attribut millisekundengenau ist oder nicht, kann im Formatattribut der Transformation das erwartete Format (yyyy-MM-dd'T'HH:mm:ss.SSSZ für millisekundengenaue Zeitangabe, yyyy-MM-dd'T'HH:mm:ssZ für sekundengenaue Zeitangabe) angegeben werden. Die Anwendung einer Transformation für eine millisekundengenaue Zeitangabe auf eine sekundengenaue Zeitangabe und umgekehrt führt zu einem Fehler.

2. Keine Format-Angabe

   Wenn Sie kein Format angeben, werden bei der Transformation beide Zeitformate (dd'T'HH:mm:ss.SSSZ und yyyy-MM-dd'T'HH:mm:ssZ) nacheinander angewendet. Wenn eines der Formate erkannt wird, wird ein entsprechender Wert an das PPM-Attribut geschrieben.

4.2.2.2.1.2 Fließkommazahltransformation

PPM erwartet für den PPM-Datentyp **DOUBLE** Werte ohne Tausendertrennzeichen und mit Punkt als Dezimaltrennzeichen. Entspricht der Quellsystemattributwert nicht diesem Format, so muss er entsprechend umgewandelt werden.

Beim Import von Fließkommazahlen können Sie bei Verwendung der Attributtransformation **double** im Attribut **format** angeben, welche Dezimal- und Tausendertrennzeichen für das Parsen eines Double-Wertes verwendet werden sollen.

```
<transformation type="double" format=","/>
```

```
<transformation type="double" format="."/>
```

Ist nur ein Zeichen angegeben, so muss dies das Dezimaltrennzeichen sein. Werden zwei Zeichen angegeben, so muss das erste Zeichen das Tausendertrennzeichen und das zweite das Dezimaltrennzeichen sein.

Beispiel

Attributwerte mit Tausendertrennzeichen (.) und Dezimaltrennzeichen (,)

1.000,00

2.324.213,42

Beispiel

Definition von attribute-Elementen

```
<attribute ppmattributetype="AT_END_TIME">
   <eventattributetype>TIMESTAMP_FIELD</eventattributetype>
   <transformation type="double" format="."/>
</attribute>
```

4.2.2.3 Organisationseinheiten

Organisationseinheiten werden dynamisch an Funktionen der Fragmentinstanz erzeugt.

ORGANISATIONSEINHEIT DEFINIEREN

Das XML-Element **orgunit** in der Mapping-Datei definiert eine Organisationseinheit. Der Wert des XML-Attributs **eventattributetype** bestimmt den Namen des Quellsystemattributs, das den Namen der Organisationseinheit enthält. Dieser Name wird den Objektattributen **AT_OBJNAME** und **AT_OBJNAME_INTERN** der Organisationseinheit zugewiesen.
Folgender Dateiauszug der Mapping-Datei erzeugt zwei Organisationseinheiten, deren Namen aus den Quellsystemattributen **PROCESSOR_1** und **PROCESSOR_2** gelesen werden. Die Bearbeitungsanzahl der Funktionen wird aus den mit dem XML-Attribut **numeventattributtype** angegebenen Quellsystemattributen **NUM_OF_PROCESSINGS_1** und **NUM_OF_PROCESSINGS_2** gelesen.

```
...
<attributmapping>
  ...
    <orgunit eventattributetype="PROCESSOR_1"
        numeventattributetype="NUM_OF_PROCESSINGS_1"/>
    <orgunit eventattributetype="PROCESSOR_2"
        numeventattributetype="NUM_OF_PROCESSINGS_2"/>
  ...
</attributmapping>
...
```

Durch geeignetes Attribut-Mapping können weitere Attribute an Organisationseinheiten erzeugt werden oder existierende Attribute überschrieben werden.
Folgender Dateiauszug weist durch Überschreiben des Objektattributs **AT_OBJNAME** dem an der PPM-Oberfläche angezeigten Namen der Organisationseinheit unabhängig vom Quellsystemattributwert den Wert **anonymer Bearbeiter** zu.

```
...
  <attributmapping>
  ...
    <orgunit eventattributetype="PROCESSOR_1">
      <attribute ppmattributetype="AT_OBJNAME">
        <value>anonymer Bearbeiter</value>
      </attribute>
    </orgunit>
  ...
  </attributmapping>
...
```

ORGANISATIONSEINHEITEN ZUORDNEN

Organisationseinheiten werden über das XML-Element **objectattributes** bestimmten Funktionen der Fragmentinstanz zugeordnet. Im XML-Attribut **objectname** wird der Identifizierer der gewünschten Funktion angegeben (Objektattribut **AT_OBJNAME_INTERN**). Optional lässt sich die Objektspezifizierung durch Angabe der Graph-ID (XML-Attribut **graphid**) verfeinern.
Das folgende Beispiel erzeugt an der Funktion mit dem Identifizierer **SAP.AUFT_ANLEG** der

Fragmentinstanz **AUFTRAG_ANLEGEN** (Graph-ID der Fragmentdefinition) aus dem Quellsystemattribut **VBAP-ERNAM** eine Organisationseinheit:

```
<attributemapping>
   <objectattributes objectname="SAP.AUFT_ANLEG"
                  graphid="AUFTRAG_ANLEGEN">
      <orgunit eventattributetype="VBAP-ERNAM"/>
   </objectattributes>
</attributemapping>
```

ORGANISATIONSEINHEITEN UND VERDICHTEN

Während des temporären Verdichtungsvorgangs bleiben die während der Anonymisierung erzeugten Organisationseinheiten erhalten, Bearbeiter werden dagegen gelöscht.

Wenn Sie einen realen Bearbeiter ohne Anonymisierung als Organisationseinheit übernehmen möchten, müssen Sie im Attribut-Mapping an der entsprechenden Organisationseinheit das Attribut **AT_ISUSERGROUP** mit dem Wert **TRUE** erzeugen.

Im folgenden Beispiel wird an allen Funktionen mit dem internen Objektnamen **FCT_A** im Prozessinstanzfragment mit der Graph-ID **FRG_B** eine Organisationseinheit erzeugt, deren Inhalt aus dem Quellsystemattribut **UNIT_GROUP_LABEL** ausgelesen wird:

```
<eventmapping>
   <processfragmentmapping>
   ...
   </processfragmentmapping>
   <attributemapping>
   ...
      <objectattributes objectname="FCT_A" graphid="FRG_B">
         <orgunit eventattributetype="UNIT_GROUP_LABEL">
            <attribute ppmattributetype="AT_ISUSERGROUP">
               <value>TRUE</value>
            </attribute>
         </orgunit>
      ...
      </objectattributes>
      ...
   </attributemapping>
</eventmapping>
```

4.2.2.4 Sonderfall des Attribut-Mapping

Existieren innerhalb eines Fragmentdefinitionsgraphen mehrere identische Objekte (identischer Attributwert **AT_OBJNAME_INTERN**), müssen Sie zur Unterscheidung in der Fragmentdefinition ein eindeutiges Objektattribut **AT_NODE_ID** angeben. In der Attribut-Mapping-Datei geben Sie anstelle des Wertes des Objektattributs **AT_OBJNAME_INTERN** den Wert des Objektattributs **AT_NODE_ID** an.

Beispiel

Fragmentdefinition:

Die Fragmentdefinition enthält zwei Endereignisse **Auftrag ist angelegt**.

...

```
  <graph id="CHPFLICHT_AUFTRAG_ANLEGEN">
    <node id="4" type="OT_EVT">
      <attribute type="AT_OBJNAME">Auftrag ist anzulegen</attribute>
      <attribute type="AT_OBJNAME_INTERN">SAP.AUFT_ANZU</attribute>
    </node>
    <node id="5" type="OT_FUNC">
      <attribute type="AT_OBJNAME">Auftrag anlegen</attribute>
      <attribute type="AT_OBJNAME_INTERN">SAP.AUFT_ANLEG</attribute>
    </node>
    <node id="42" type="OT_RULEOR"/>
    <node id="6" type="OT_EVT">
      <attribute type="AT_OBJNAME">Auftrag ist angelegt</attribute>
      <attribute type="AT_OBJNAME_INTERN">SAP.AUFT_ANGELEGT</attribute>
      <attribute type="AT_NODE_ID">Endereignis 1</attribute>
    </node>
    <node id="7" type="OT_EVT">
      <attribute type="AT_OBJNAME">Auftrag ist angelegt</attribute>
      <attribute type="AT_OBJNAME_INTERN">SAP.AUFT_ANGELEGT</attribute>
      <attribute type="AT_NODE_ID">Endereignis 2</attribute>
    </node>
    <edge type="CXN_FOLLOWS" source="4" target="5"/>
    <edge type="CXN_FOLLOWS" source="5" target="42"/>
    <edge type="CXN_FOLLOWS" source="42" target="6"/>
    <edge type="CXN_FOLLOWS" source="42" target="7"/>
  </graph>
...
```

Attribut-Mapping:

Das Attribut-Mapping erzeugt an beiden instanziierten Endereignissen **Auftrag ist angelegt** ein PPM-Attribut **AT_ID** mit unterschiedlichem Wert:

```
...
  <attributemapping>
    <objectattributes objectname="Endereignis 1" graphid="AUFTRAG_ANLEGEN">
      <attribute ppmattributetype="AT_ID">
        <value>Dies ist das erste Endereignis</value>
      </attribute>
    </objectattributes>
    <objectattributes objectname="Endereignis 2" graphid="AUFTRAG_ANLEGEN">
      <attribute ppmattributetype="AT_ID">
        <value>Dies ist das zweite Endereignis</value>
      </attribute>
    </objectattributes>
  </attributemapping>
...
```

4.2.3 Erstellen von Fragmentdefinitionen in ARIS

Für den korrekten Import der Daten spielt es keine Rolle, wie die Fragmentdefinitionsdatei und die Mapping-Datei erzeugt wurden. Einfacher als das direkte Editieren der XML-Dateien ist das Erstellen der Fragmentdefinitionsdatei mit Hilfe von ARIS bzw. der Mapping-Datei mit PPM Customizing Toolkit.

Die Prozessfragmentdefinitionen werden als EPK modelliert. Für das Modellieren steht ein spezieller ARIS-Methodenfilter zur Verfügung. Häufig ist der von PPM zu analysierende

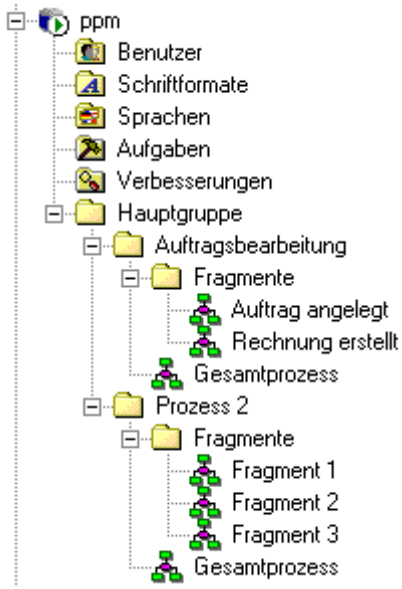
Gesamtprozess bereits mit Hilfe von ARIS modelliert und kann als Ausgangsbasis zur Fragmenterstellung verwendet werden.

Die EPKs der Prozessfragmentdefinitionen können auch direkt modelliert werden. In der Praxis hat sich aus Gründen der Übersichtlichkeit jedoch die Erstellung eines Gesamtprozesses bewährt.

4.2.3.1 Modellieren des Gesamtprozesses

Zunächst wird der gesamte von PPM zu überwachende Prozess anhand des realen Aktivitätenflusses im Anwendungssystem als EPK modelliert. Achten Sie beim Erstellen des Prozesses darauf, dass alle vorkommenden System-Events in der EPK als Ereignis einer vorangegangenen Funktion modelliert sind. An den Funktionen sollten Organisationseinheiten gepflegt sein.

Legen Sie für jeden Gesamtprozess eine eigene Gruppe mit einer Untergruppe für die Prozessfragmentdefinitionen an. Modellieren Sie eine EPK für den Gesamtprozess und für jedes Fragment eine weitere EPK in der Untergruppe.



MODELLIERUNGSRICHTLINIEN FÜR DEN GESAMTPROZESS

- Ein Prozess beginnt mit einem Ereignis und endet mit mindestens einem Ereignis.
- Vor verzweigenden OR- und XOR-Regeln dürfen keine Ereignisse stehen.
- Pflegen Sie für alle Ereignisse und Funktionen die ARIS-Attribute **Name** und **Identifizierer**. Der Name wird beim Ausführen des ARIS-Reports als sprachabhängiges PPM-Attribut **AT_OBJNAME**, der Identifizierer als sprachunabhängiges PPM-Attribut **AT_OBJNAME_INTERN** in die Fragmentdefinition übernommen.

4.2.3.2 Modellieren der Prozessfragmentdefinitionen

Erzeugen Sie zunächst für alle Funktionen des Gesamtprozesses in der Gruppe, die die Fragmentdefinitionen enthalten soll, eine Modellhinterlegung. Der Identifizierer des hinterlegten Modells (Prozessfragmentdefinition) bestimmt den Namen der Fragmentdefinition. Kopieren Sie anschließend alle Ereignisse, die im Gesamtprozess ablauflogisch an eine Funktion angrenzen, in das der Funktion hinterlegte Modell.

Beachten Sie folgende Richtlinien beim Modellieren der Fragmente, unabhängig davon, ob Sie mit einem Gesamtprozess arbeiten oder die Fragmente ohne Gesamtprozess erstellen.

MODELLIERUNGSRICHTLINIEN FRAGMENTDEFINITION

Verwenden Sie den speziellen PPM-Methodenfilter **FragmentXML_ARISToolSet_Filter.amc** aus dem Verzeichnis **<PPM-Installationsverzeichnis>\ppm\server\bin\agentLocalRepo\.unpacked\<Installationszeit>_ppm-client-run-prod-<Version>-runnable.zip\ppm\ctk\ARIS**. Er beschränkt die Menge der Modellierungselemente auf die zulässigen Objekt- und Kantentypen und erleichtert Ihnen so das Modellieren.

- Jedes Prozessfragment muss in einem eigenen Modell des Typs **EPK** modelliert werden. Der Identifizierer des Modells wird als **graph id** in die XML-Fragmentdefinitionsdatei übernommen und bestimmt den Namen des Prozessfragmentes.

Wenn Sie einen Gesamtprozess verwenden, erzeugen Sie beim Modellieren der Fragmente Ausprägungskopien der Objekte des Gesamtprozesses.

- Vermeiden Sie zyklische Prozesspfade (Schleifen). Eine Fragmentdefinition, die Zyklen enthält, wird vom Modellreport erkannt und ignoriert, es erfolgt ein entsprechender Hinweis im Ausgabefenster von ARIS.
- Alle Objekte (Ereignisse, Funktionen, Regeln) der Fragmentmodelle müssen einen eindeutigen Identifizierer haben. Dieser wird als sprachunabhängiges Attribut **AT_OBJNAME_INTERN** nach PPM übernommen.
- Organisationseinheiten müssen nicht modelliert werden. Sie werden bei der Erzeugung der Fragmentdefinitionsdatei mittels ARIS Report ignoriert. Organisationseinheiten weisen Sie über das Attribut-Mapping (XML-Element **orgunit**) den entsprechenden Funktionen dynamisch zu (siehe **Organisationseinheiten** (Seite 29)).

An allen Objekten und Kanten können Sie zusätzlich **Freie Attribute** pflegen. Diese werden als feste PPM-Attribut-Wert-Kombinationen vom Typ **TEXT** an die entsprechenden Objekte bzw. Kanten der Fragmentdefinitionsdatei geschrieben. Die Liste muss beginnend mit **Benutzer Attribut Text 1** aufsteigend gepflegt sein und wird bis zum ersten nicht gepflegten Attribut ausgewertet. Format: **<Attributschlüssel>#<Attributwert>**. Das Trennzeichen **#** darf weder im Attributschlüssel noch im Attributwert vorkommen.

Beispiel

Für das Ereignis **Neukundenauftrag anzulegen** werden folgende Freien Attribute gepflegt:

	Neukunden-auftrag anzulegen
Benutzerattribut Text 1	AT_PLANT_NAME#Duesseldorf
Benutzerattribut Text 2	AT_PLANT_ID#00123
Benutzerattribut Text 3	AT_CUST_ID#456777
Benutzerattribut Text 4	AT_CUST_NAME#Mayer
Benutzerattribut Text 5	
Benutzerattribut Text 6	AT_ORD_NUM#023655
Benutzerattribut Text 7	
Benutzerattribut Text 8	AT_DIVISION_NAME#HIFI-Video
Benutzerattribut Text 9	

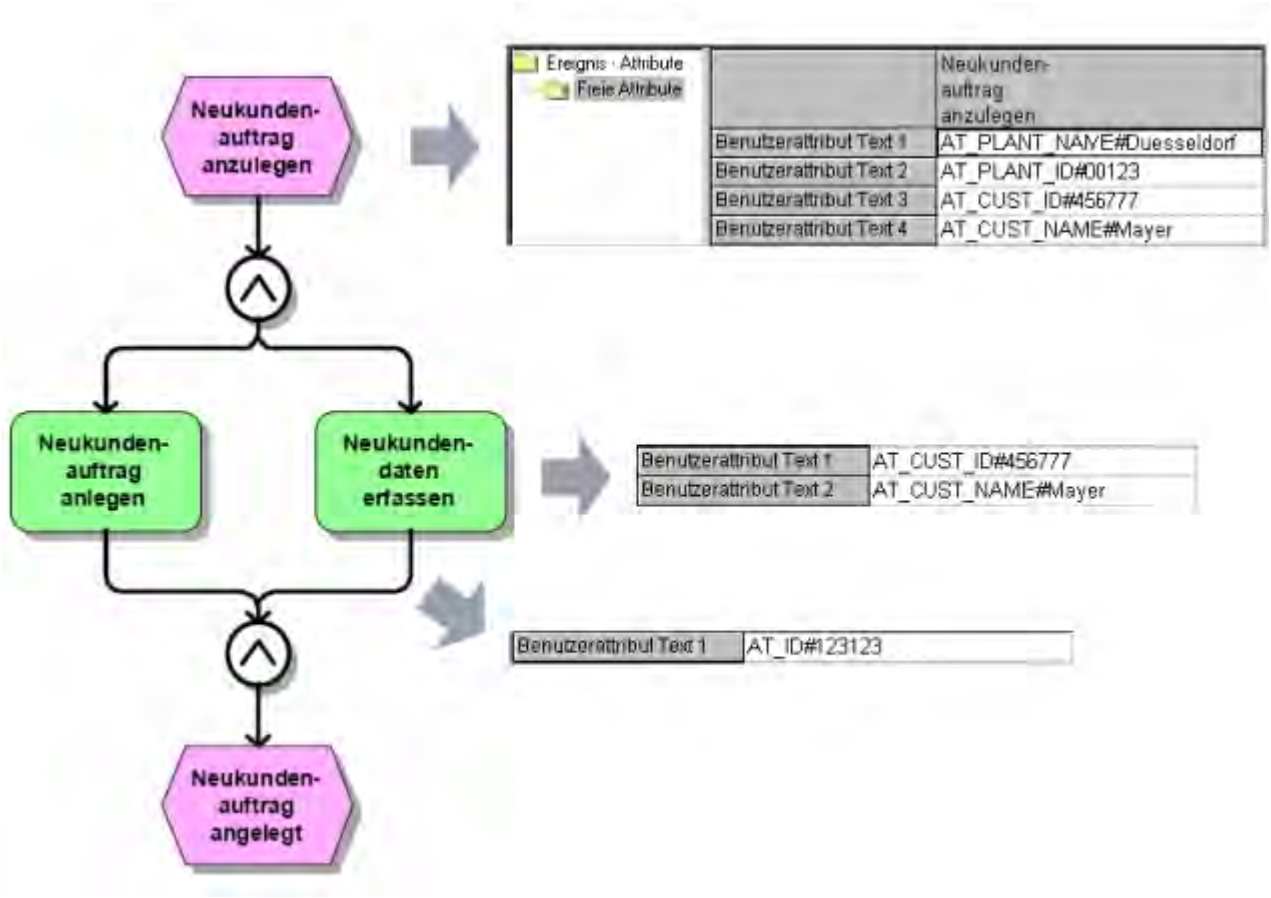
Im folgenden Dateiauszug der erzeugten Fragmentdefinitionsdatei sind die übernommenen Freien Attribute in **Fettschrift** dargestellt.

```
Ermittle die neuen bzw. geänderten MM-Belege ...
<node id="EVT_NEWCUSTORD2BE_CREATED" type="OT_EVT">
     <attribute type="AT_OBJNAME">Neukundenauftrag anzulegen</attribute>
     <attribute type="AT_OBJNAME_INTERN">EVT_NEWCUSTORD2BE_CREATED</attribute>
     <attribute type="AT_PLANT_NAME">Duesseldorf</attribute>
     <attribute type="AT_PLANT_ID">00123</attribute>
     <attribute type="AT_CUST_ID">456777</attribute>
     <attribute type="AT_CUST_NAME">Mayer</attribute>
</node>
Ermittle die neuen bzw. geänderten MM-Belege ...
```

- Folgende ARIS Kantentypen werden von der Reportauswertung berücksichtigt und in den PPM-Kantentyp **CXN_FOLLOWS** überführt:

Quellobjekt	Zielobjekt	ARIS Kantentyp
Ereignis	Funktion	**aktiviert**
Ereignis	Regel	**wird ausgewertet von**
Funktion	Ereignis	**erzeugt**
Funktion	Regel	**führt zu**
Funktion	Funktion	**ist Vorgänger von**
Regel	Ereignis	**führt zu**
Regel	Funktion	aktiviert
Regel	Regel	verknüpft

Die folgende Abbildung zeigt ein konform modelliertes Prozessfragment mit zusätzlich gepflegten Freien Attributen an Objekten und einer Kante:



Vom **ARIS Report** wird die folgende Fragmentdefinitionsdatei erzeugt:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE graphlist SYSTEM "graph.dtd">
<graphlist>
  <graph id="FRG_NEWCUST_ORD">
    <node id="EVT_NEWCUSTORD2BE_CREATED" type="OT_EVT">
      <attribute type="AT_OBJNAME">Neukundenauftrag anzulegen</attribute>
      <attribute type="AT_OBJNAME_INTERN">EVT_NEWCUSTORD2BE_CREATED</attribute>
      <attribute type="AT_PLANT_NAME">Duesseldorf</attribute>
      <attribute type="AT_PLANT_ID">00123</attribute>
      <attribute type="AT_CUST_ID">456777</attribute>
      <attribute type="AT_CUST_NAME">Mayer</attribute>
    </node>
    <node id="FCT_CREATE_NEWCUSTORD" type="OT_FUNC">
      <attribute type="AT_OBJNAME">Neukundenauftrag anlegen</attribute>
      <attribute type="AT_OBJNAME_INTERN">FCT_CREATE_NEWCUSTORD</attribute>
    </node>
    <node id="OT_RULE_AND" type="OT_RULEAND">
      <attribute type="AT_OBJNAME_INTERN">OT_RULE_AND</attribute>
      <attribute type="AT_SAP_ID">455455</attribute>
    </node>
    <node id="FCT_REC_NEWCUSTDAT" type="OT_FUNC">
      <attribute type="AT_OBJNAME">Neukundendaten erfassen</attribute>
      <attribute type="AT_OBJNAME_INTERN">FCT_REC_NEWCUSTDAT</attribute>
      <attribute type="AT_CUST_ID">456777</attribute>
```

```
      <attribute type="AT_CUST_NAME">Mayer</attribute>
    </node>
    <node id="OT_RULE_AND" type="OT_RULEAND">
      <attribute type="AT_OBJNAME_INTERN">OT_RULE_AND</attribute>
    </node>
    <node id="EVT_NEWCUSTORD_CREATED" type="OT_EVT">
      <attribute type="AT_OBJNAME">Neukundenauftrag angelegt</attribute>
      <attribute type="AT_OBJNAME_INTERN">EVT_NEWCUSTORD_CREATED</attribute>
    </node>
    <edge type="CXN_FOLLOWS" source="FCT_REC_NEWCUSTDAT" target="OT_RULE_AND">
      <attribute type="AT_ID">123123</attribute>
    </edge>
    <edge type="CXN_FOLLOWS" source="EVT_NEWCUSTORD2BE_CREATED"
       target="OT_RULE_AND"/>
    <edge type="CXN_FOLLOWS" source="OT_RULE_AND" target="FCT_REC_NEWCUSTDAT"/>
    <edge type="CXN_FOLLOWS" source="OT_RULE_AND"
      target="EVT_NEWCUSTORD_CREATED"/>
    <edge type="CXN_FOLLOWS" source="FCT_CREATE_NEWCUSTORD"
      target="OT_RULE_AND"/>
    <edge type="CXN_FOLLOWS" source="OT_RULE_AND"
       target="FCT_CREATE_NEWCUSTORD"/>
  </graph>
</graphlist>
```

ÜBERSICHT MODELLIERUNGSRICHTLINIEN PROZESSFRAGMENTE

ARIS-Element	ARIS-Attribut	PPM-Fragmentdefinition
EPK (Fragmentmodell)	Name	**graph id** des erzeugten Definitionsgraphen
Funktion	Name	AT_OBJNAME
	Identifizierer	AT_OBJNAME_INTERN
Ereignis	Name	AT_OBJNAME
	Identifizierer	AT_OBJNAME_INTERN
Regel	Identifizierer	AT_OBJNAME_INTERN
Freie Attribute	Benutzer Attribut Text <x> x = Ganzzahl von 1 bis 37	PPM-Attribut vom Typ **TEXT** inklusive Wert an Objekten bzw. Kanten)

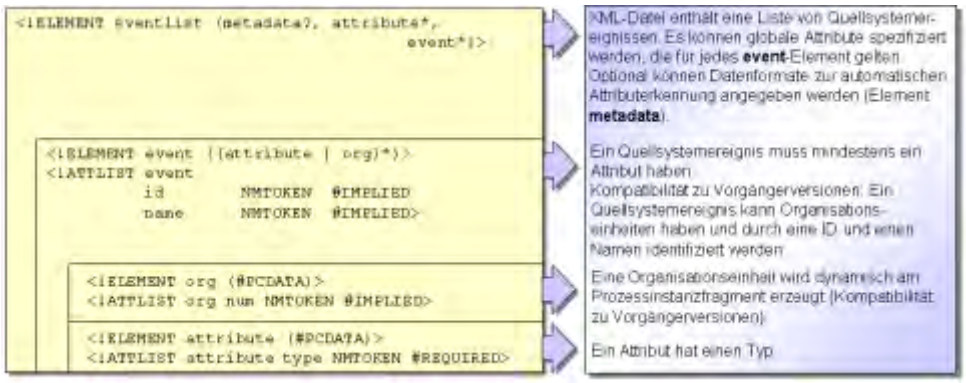
4.2.3.3 Format der System-Event-Datei

Die System-Event-Datei beinhaltet die eigentlichen Instanzinformationen über den tatsächlichen Ablauf der Prozesse. Ein Auszug aus der XML-Datei zum oben gezeigten Beispiel könnte folgendermaßen aussehen:

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

```
...
  <event>
    <attribute type="PPM_EVENT_ID">A-Nr 4711</attribute>
    <attribute type="PPM_EVENT_NAME">AUFTRAG_ANGELEGT</attribute>
    <attribute type="AUFTRAGNR">4711</attribute>
    <attribute type="AUFTRAGART">Sofortauftrag</attribute>
    <attribute type="START_TIME">11.11.2002 11:11:11</attribute>
    <attribute type="PPM_ORG_NAME">Herr Müller</attribute>
    <attribute type="PPM_ORG_NUM">1</attribute>
</event>
...
  <event>
    <attribute type="PPM_EVENT_ID">A-Nr 4711</attribute>
    <attribute type="PPM_EVENT_NAME">RECHNUNG_ERSTELLT</attribute>
    <attribute type="AUFTRAGNR">4711</attribute>
    <attribute type="AUFTRAGART">Sofortauftrag</attribute>
    <attribute type="START_TIME">12.11.2002 14:21:15</attribute>
    <attribute type="PPM_ORG_NAME">Frau Schmitt</attribute>
    <attribute type="PPM_ORG_NUM">1</attribute>
  </event>
...
</eventlist>
```

Das Format der XML-Ausgabedatei ist durch die DTD des PPM-System-Event-Formats vorgegeben:



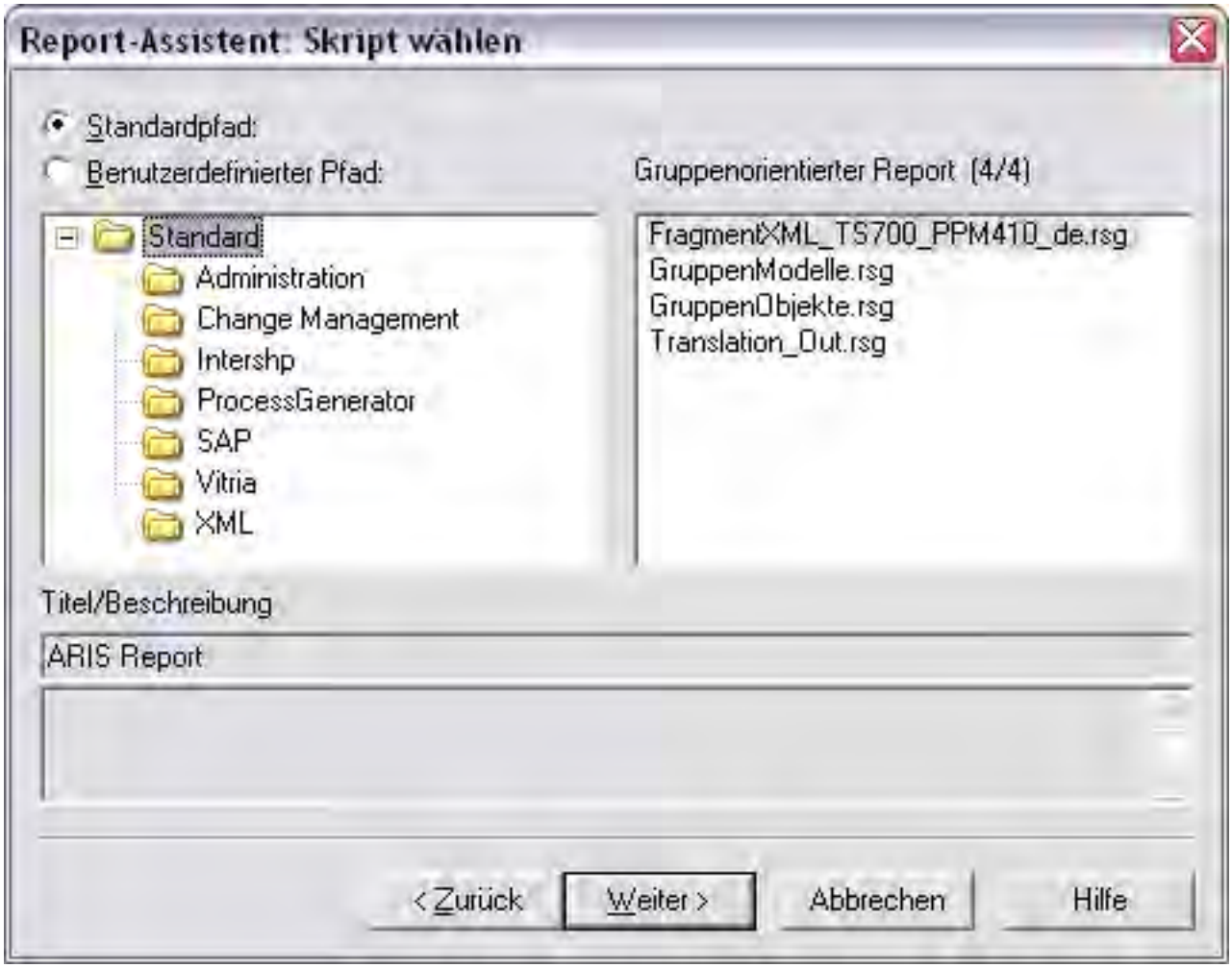
Die XML-Attribute **id** und **name** des XML-Elements **event** sowie das XML-Element **org** ermöglichen die unveränderte Weiterverwendung von System-Event-Dateien, die für die PPM-Versionen 1.x und 2.x erzeugt wurden.
Sie werden ab PPM Version 3.0 durch die System-Event-Attribute **PPM_EVENT_ID** und **PPM_EVENT_NAME** sowie durch das XML-Element **orgunit** ersetzt.

4.2.3.4 ARIS-Report ausführen

Der ARIS-Report zum Erzeugen der XML-Datei zur Fragmentdefinition ist als Gruppenreport ausgelegt.

Kopieren Sie das Reportskript **FragmentXML_TS700_PPM410_de.rsg** im <PPM-Installationsverzeichnis>\ppm\server\bin\agentLocalRepo\.unpacked\<Installationszeit>_ppm-client-run-prod-<Version>-runnable.zip\ppm\ctk\ARIS in das Unterverzeichnis **script\report\de** Ihrer ARIS-Installation (ARIS-Standardverzeichnis für Reportskripte).

Wählen Sie zum Starten des Reports das Kontextmenü **Auswerten/Report** auf der Gruppe mit den Fragmentdefinitionen.



Wenn Sie ARIS Architect verwenden, importieren Sie das Skript **FragmentXML_BA700_PPM410.arx**. Wir empfehlen für den Import eine eigene Kategorie **PPM** anzulegen. Zum Ausführen des Reports verwenden Sie anschließend den Eintrag **FragmentXML**.

Nach erfolgreicher Ausführung des Reports befindet sich im gewählten Ausgabeverzeichnis die XML-Fragmentdefinitionsdatei. Standardmäßig wird das Verzeichnis **script\report\out** der ARIS-Installation verwendet. Der Dateiname setzt sich aus dem Präfix **Frag_** und einem Zeitstempel zusammen.

- Achten Sie darauf, dass der Report ohne Warnungen und Fehlermeldungen durchläuft. Die Meldung **Zugriff auf ein nicht gepflegtes Attribut** weist darauf hin, dass ein benötigtes Attribut nicht gepflegt ist, z. B. das Attribut **Identifizierer**. Die erzeugten XML-Dateien sind in diesem Fall unbrauchbar.
- Wenn Sie kein aktuelles Reportskript zur Erstellung von Fragmentdefinitionsdateien zur Verfügung haben, müssen Sie das Skript **FragmentXML.rsg** aus dem Verzeichnis **\server\bin\agentLocalRepo\.unpacked\<Installationszeit>_ppm-client-run-prod-<Version>-runnable.zip\ppm\ctk\ARIS** Ihrer PPM-Installation mittels **ARIS Skriptumsetzer** in eine aktuell gültige Form konvertieren.

4.2.4 Generieren der XML-Ausgabedatei

Zum Generieren der Ausgabedatei müssen die möglichen System-Events identifiziert und benannt werden. Das folgende Beispiel verwendet die System-Events **Auftrag angelegt** und **Rechnung erstellt**.

Das Auftreten dieser System-Events wird mit zusätzlichen Informationen (z. B. Auftragsnummer, Bearbeitungszeiten, Basiswerten für die Kennzahlenberechnung und Bildung von Dimensionen) in eine XML-Ausgabedatei geschrieben.

4.2.5 Zusammenfassung

Das folgende einfache Beispiel soll den beschriebenen Import im System-Event-Format verdeutlichen.

Zum Importieren der Ausgabedateien wurden die Fragmentdefinitionsdatei **frag.xml** und die Mapping-Datei **map.xml** erstellt:

Fragmentdefinitionen in Datei **frag.xml**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE graphlist SYSTEM "Graph.dtd">
<graphlist>
  <graph id="FRG_ORD_CREATED">
    <node id="EVT_ORD_TOBECREATED" type="OT_EVT">
      <attribute type="AT_OBJNAME">Kundenauftrag ist anzulegen</attribute>
      <attribute type="AT_OBJNAME_INTERN">EVT_ORD_TOBECREATED</attribute>
    </node>
    <node id="FCT_CREATE_ORDER" type="OT_FUNC">
      <attribute type="AT_OBJNAME">Kundenauftrag anlegen</attribute>
      <attribute type="AT_OBJNAME_INTERN">FCT_CREATE_ORDER</attribute>
    </node>
    <node id="EVT_ORD_CREATED" type="OT_EVT">
      <attribute type="AT_OBJNAME">Kundenauftrag angelegt</attribute>
      <attribute type="AT_OBJNAME_INTERN">EVT_ORD_CREATED</attribute>
    </node>
    <edge type="CXN_FOLLOWS" source="EVT_ORD_TOBECREATED"
                          target="FCT_CREATE_ORDER"/>
    <edge type="CXN_FOLLOWS" source="FCT_CREATE_ORDER" target="EVT_ORD_CREATED"/>
  </graph>
  <graph id="FRG_INVOICED">
    <node id="EVT_TOBE_INVOICED" type="OT_EVT">
```

```
      <attribute type="AT_OBJNAME">Rechnung ist zu erstellen</attribute>
      <attribute type="AT_OBJNAME_INTERN">EVT_TOBE_INVOICED</attribute>
    </node>
    <node id="FCT_INVOICE" type="OT_FUNC">
      <attribute type="AT_OBJNAME">Rechnung erstellen</attribute>
      <attribute type="AT_OBJNAME_INTERN">FCT_INVOICE</attribute>
    </node>
    <node id="EVT_INVOICED" type="OT_EVT">
      <attribute type="AT_OBJNAME">Rechnung erstellt</attribute>
      <attribute type="AT_OBJNAME_INTERN">EVT_INVOICED</attribute>
    </node>
    <edge type="CXN_FOLLOWS" source="EVT_TOBE_INVOICED" target="FCT_INVOICE"/>
    <edge type="CXN_FOLLOWS" source="FCT_INVOICE" target="EVT_INVOICED"/>
  </graph>
</graphlist>
```

Mapping-Definitionen in Datei **map.xml**:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventmapping SYSTEM "eventmapping.dtd">
<eventmapping>
  <processfragmentmapping>
    <processfragment graphid="FRG_ORD_CREATED">
      <condition eventattributetype="PPM_EVENT_NAME">
        <value>FRG_ORD_CREATED</value>
      </condition>
    </processfragment>
    <processfragment graphid="FRG_INVOICED">
      <condition eventattributetype="PPM_EVENT_NAME">
        <value>FRG_INVOICED</value>
      </condition>
    </processfragment>
  </processfragmentmapping>
  <attributemapping>
    <objectattributes objectname="EVT_ORD_TOBECREATED" graphid="FRG_ORD_CREATED">
      <attribute ppmattributetype="AT_MERGE_KEY_1">
        <eventattributetype>EVT_PRED</eventattributetype>
      </attribute>
    </objectattributes>
    <objectattributes objectname="FCT_CREATE_ORDER" graphid="FRG_ORD_CREATED">
      <orgunit eventattributetype="PPM_ORG_NAME"/>
      <attribute ppmattributetype="AT_MERGE_KEY_2">
        <eventattributetype>START_TIME</eventattributetype>
      </attribute>
      <attribute ppmattributetype="AT_KI_FBZ">
        <eventattributetype>PROCESSINGTIME</eventattributetype>
      </attribute>
      <attribute ppmattributetype="AT_SAPCLIENT">
        <eventattributetype>KUNDENNR</eventattributetype>
      </attribute>
      <attribute ppmattributetype="AT_SAP_BSTYP">
        <eventattributetype>AUFTRAGAT</eventattributetype>
      </attribute>
    </objectattributes>
    <objectattributes objectname="EVT_ORD_CREATED" graphid="FRG_ORD_CREATED">
      <attribute ppmattributetype="AT_ID">
        <eventattributetype>AUFTRAGNR</eventattributetype>
      </attribute>
      <attribute ppmattributetype="AT_SAPCLIENT">
        <eventattributetype>KUNDENNR</eventattributetype>
      </attribute>
```

```
      <attribute ppmattributetype="AT_MERGE_KEY_1">
        <eventattributetype>EVT_NR</eventattributetype>
      </attribute>
    </objectattributes>
    <objectattributes objectname="EVT_TOBE_INVOICED" graphid="FRG_INVOICED">
      <attribute ppmattributetype="AT_MERGE_KEY_1">
        <eventattributetype>EVT_PRED</eventattributetype>
      </attribute>
      <attribute ppmattributetype="AT_ID">
        <eventattributetype>AUFTRAGNR</eventattributetype>
      </attribute>
    </objectattributes>
    <objectattributes objectname="FCT_INVOICE" graphid="FRG_INVOICED">
      <orgunit eventattributetype="PPM_ORG_NAME"/>
      <attribute ppmattributetype="AT_MERGE_KEY_2">
        <eventattributetype>START_TIME</eventattributetype>
      </attribute>
      <attribute ppmattributetype="AT_KI_FBZ">
        <eventattributetype>PROCESSINGTIME</eventattributetype>
      </attribute>
    </objectattributes>
    <objectattributes objectname="EVT_INVOICED" graphid="FRG_INVOICED">
      <attribute ppmattributetype="AT_ID">
        <eventattributetype>AUFTRAGNR</eventattributetype>
      </attribute>
      <attribute ppmattributetype="AT_SAPCLIENT">
        <eventattributetype>KUNDENNR</eventattributetype>
      </attribute>
      <attribute ppmattributetype="AT_MERGE_KEY_1">
        <eventattributetype>EVT_NR</eventattributetype>
      </attribute>
    </objectattributes>
  </attributemapping>
</eventmapping>
```

Das Auslesen des Anwendungssystems hat die XML-Ausgabedatei **events.xml** erzeugt:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventlist SYSTEM "event.dtd">
<eventlist>
  <event>
    <attribute type="EVT_PRED">0</attribute>
    <attribute type="PPM_EVENT_ID">4711</attribute>
    <attribute type="PPM_EVENT_NAME">FRG_ORD_CREATED</attribute>
    <attribute type="PPM_ORG_NAME">Herr Müller</attribute>
    <attribute type="AUFTRAGNR">4711</attribute>
    <attribute type="AUFTRAGART">Sofortauftrag</attribute>
    <attribute type="START_TIME">11.11.2002 11:11:11</attribute>
    <attribute type="KUNDENNR">5711</attribute>
    <attribute type="PROCESSINGTIME">7 MINUTE</attribute>
    <attribute type="EVT_NR">1</attribute>
  </event>
  <event>
    <attribute type="EVT_PRED">1</attribute>
    <attribute type="PPM_EVENT_ID">4711</attribute>
    <attribute type="PPM_EVENT_NAME">FRG_INVOICED</attribute>
    <attribute type="PPM_ORG_NAME">Frau Schmitt</attribute>
    <attribute type="AUFTRAGNR">4711</attribute>
    <attribute type="KUNDENNR">5711</attribute>
    <attribute type="START_TIME">12.11.2002 14:21:15</attribute>
    <attribute type="PROCESSINGTIME">14 MINUTE</attribute>
```

```
    <attribute type="EVT_NR">2</attribute>
  </event>
</eventlist>
```

Durch Ausführen der Kommandozeile

runxmlimport -user system -password manager -f frag.xml -m map.xml -i event.xml

werden in der PPM-Datenbank Prozessinstanzfragmente erzeugt. Folgendes Listing im PPM-Graphformat veranschaulicht die in der PPM-Datenbank erzeugten Prozessinstanzfragmente. Im Listing sind die Prozessinstanzfragment-Informationen, die der Fragmentdefinition hinzugefügt wurden, durch Fettschrift gekennzeichnet:

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE graphlist SYSTEM "graph.dtd">
<graphlist>
  <graph id="FRG_ORD_CREATED" xml:lang="de">
    <node id="FRG_ORD_CREATED-EVT_ORD_TOBECREATED-1-8835472025300230616"
                                                  type="OT_EVT">
      <attribute type="AT_OBJNAME">Kundenauftrag ist anzulegen</attribute>
      <attribute type="AT_OBJNAME_INTERN">EVT_ORD_TOBECREATED</attribute>
      <attribute type="AT_EPK_FRAGMENT_ID">FRG_ORD_CREATED</attribute>
      <attribute type="AT_INTERNAL_OBJECT_KEY">EVT_ORD_TOBECREATED0</attribute>
      <attribute type="AT_MERGE_KEY_1">0</attribute>
      <attribute type="AT_ORIG_EPK_ID">1</attribute>
    </node>
    <node id="FRG_ORD_CREATED-FCT_CREATE_ORDER-2-8344758599463564840"
                                                  type="OT_FUNC">
      <attribute type="AT_OBJNAME">Kundenauftrag anlegen</attribute>
      <attribute type="AT_OBJNAME_INTERN">FCT_CREATE_ORDER</attribute>
      <attribute type="AT_EPK_FRAGMENT_ID">FRG_ORD_CREATED</attribute>
      <attribute type="AT_INTERNAL_OBJECT_KEY">
                          FCT_CREATE_ORDER11.11.2002 11:11:11</attribute>
      <attribute type="AT_KI_FBZ">7 MINUTE</attribute>
      <attribute type="AT_MERGE_KEY_2">11.11.2002 11:11:11</attribute>
      <attribute type="AT_ORIG_EPK_ID">1</attribute>
      <attribute type="AT_SAPCLIENT">5711</attribute>
    </node>
    <node id="FRG_ORD_CREATED-EVT_ORD_CREATED-3-7299716715746582786"
                                                  type="OT_EVT">
      <attribute type="AT_OBJNAME">Kundenauftrag angelegt</attribute>
      <attribute type="AT_OBJNAME_INTERN">EVT_ORD_CREATED</attribute>
      <attribute type="AT_EPK_FRAGMENT_ID">FRG_ORD_CREATED</attribute>
      <attribute type="AT_ID">4711</attribute>
      <attribute type="AT_INTERNAL_OBJECT_KEY">EVT_ORD_CREATED1</attribute>
      <attribute type="AT_MERGE_KEY_1">1</attribute>
      <attribute type="AT_ORIG_EPK_ID">1</attribute>
      <attribute type="AT_SAPCLIENT">5711</attribute>
    </node>
    <node id="FRG_ORD_CREATED-1--2--7661491465378853387" type="OT_ORG">
      <attribute type="AT_INTERNAL_OBJECT_KEY">
                   FRG_ORD_CREATED-1--2-7661491465378853387</attribute>
      <attribute type="AT_OBJNAME">Herr Müller</attribute>
    </node>
    <edge type="CXN_FOLLOWS"
          source="FRG_ORD_CREATED-EVT_ORD_TOBECREATED-1-8835472025300230616"
          target="FRG_ORD_CREATED-FCT_CREATE_ORDER-2-8344758599463564840" />
    <edge type="CXN_FOLLOWS"
          source="FRG_ORD_CREATED-FCT_CREATE_ORDER-2-8344758599463564840"
          target="FRG_ORD_CREATED-EVT_ORD_CREATED-3-7299716715746582786" />
    <edge type="CXN_UNDIRECTED"
```

```
        source="FRG_ORD_CREATED-1--2-7661491465378853387"
        target="FRG_ORD_CREATED-FCT_CREATE_ORDER-2-8344758599463564840">
      <attribute type="AT_COUNT_PROCESSINGS">1</attribute>
    </edge>
  </graph>
  <graph id="FRG_INVOICED" xml:lang="de">
    <node id="FRG_INVOICED-EVT_TOBE_INVOICED-1-246376083169224336" type="OT_EVT">
      <attribute type="AT_OBJNAME">Rechnung ist zu erstellen</attribute>
      <attribute type="AT_OBJNAME_INTERN">EVT_TOBE_INVOICED</attribute>
      <attribute type="AT_EPK_FRAGMENT_ID">FRG_INVOICED</attribute>
      <attribute type="AT_ID">4711</attribute>
      <attribute type="AT_INTERNAL_OBJECT_KEY">EVT_TOBE_INVOICED1</attribute>
      <attribute type="AT_MERGE_KEY_1">1</attribute>
      <attribute type="AT_ORIG_EPK_ID">2</attribute>
    </node>
    <node id="FRG_INVOICED-FCT_INVOICE-2--1285282699037363371" type="OT_FUNC">
      <attribute type="AT_OBJNAME">Rechnung erstellen</attribute>
      <attribute type="AT_OBJNAME_INTERN">FCT_INVOICE</attribute>
      <attribute type="AT_EPK_FRAGMENT_ID">FRG_INVOICED</attribute>
      <attribute type="AT_INTERNAL_OBJECT_KEY">FCT_INVOICE12.11.2002
                                        14:21:15</attribute>
      <attribute type="AT_KI_FBZ">14 MINUTE</attribute>
      <attribute type="AT_MERGE_KEY_2">12.11.2002 14:21:15</attribute>
      <attribute type="AT_ORIG_EPK_ID">2</attribute>
    </node>
    <node id="FRG_INVOICED-EVT_INVOICED-3-1541227247726154405" type="OT_EVT">
      <attribute type="AT_OBJNAME">Rechnung erstellt</attribute>
      <attribute type="AT_OBJNAME_INTERN">EVT_INVOICED</attribute>
      <attribute type="AT_EPK_FRAGMENT_ID">FRG_INVOICED</attribute>
      <attribute type="AT_ID">4711</attribute>
      <attribute type="AT_INTERNAL_OBJECT_KEY">EVT_INVOICED2</attribute>
      <attribute type="AT_MERGE_KEY_1">2</attribute>
      <attribute type="AT_ORIG_EPK_ID">2</attribute>
      <attribute type="AT_SAPCLIENT">5711</attribute>
    </node>
    <node id="FRG_INVOICED-1--2--8231301810541650135" type="OT_ORG">
      <attribute type="AT_INTERNAL_OBJECT_KEY">FRG_INVOICED-1--2-
                           8231301810541650135</attribute>
      <attribute type="AT_OBJNAME">Frau Schmitt</attribute>
    </node>
    <edge type="CXN_FOLLOWS"
          source="FRG_INVOICED-EVT_TOBE_INVOICED-1-246376083169224336"
          target="FRG_INVOICED-FCT_INVOICE-2--1285282699037363371" />
    <edge type="CXN_FOLLOWS"
          source="FRG_INVOICED-FCT_INVOICE-2-1285282699037363371"
          target="FRG_INVOICED-EVT_INVOICED-3-1541227247726154405" />
    <edge type="CXN_UNDIRECTED"
          source="FRG_INVOICED-1--2--8231301810541650135"
          target="FRG_INVOICED-FCT_INVOICE-2--1285282699037363371">
      <attribute type="AT_COUNT_PROCESSINGS">1</attribute>
    </edge>
  </graph>
</graphlist>
```

4.3 Datenformate

Dieses Kapitel beschreibt die Formate der Quellsystemdaten, die in Form von Attributwerten in der XML-Ausgabedatei erwartet werden.

Durch das Attribut-Mapping ist den Quellsystemattributen ein PPM-Attributtyp zugeordnet. Quellsystemattribute sind reine Informationsträger in Form einer Zeichenkette. Der Datentyp des PPM-Attributtyps legt fest, wie der Text des Quellsystemattributs interpretiert wird.

NUMERISCHE DATEN

Numerische Daten werden grundsätzlich als Angabe im Dezimalsystem interpretiert. Fließkommazahlen verwenden als Trennzeichen zwischen Vor- und Nachkommastellen unabhängig von der Einstellung des Betriebssystems den Punkt.

Attributwerte werden mit Angabe einer Einheit eingelesen, z. B. **57 USD**. Ohne Angabe einer Einheit wird die Basiseinheit des Datentyps angenommen, auf dem der PPM-Attributtyp basiert. Prozentwerte werden entweder mit **58 PERCENT** oder als Faktor mit **0.58 VALUE_ONLY** angegeben.

ZEITBASIERTE DATEN

PPM unterscheidet drei Kategorien zeitbasierter Nutzdaten:

Datum:
Es wird der Kalendertag angegeben, z. B. **26.06.2003**.

Uhrzeit:
Es wird die Tageszeit im 24-Stunden-Format angegeben, z. B. **14:29**.

Zeitstempel:
Die Angabe eines Zeitstempels setzt sich aus einer Datumsangabe mit Uhrzeit zusammen, z. B. **26.06.2003 14:29**.

Zum Parsen der Datenzeichenkette wird die Formatzeichenkette verwendet, die in der Datei **AdapterConfig_settings.properties** im Mandantenkonfigurationsverzeichnis für die jeweilige Kategorie angegeben ist.

Warnung

Die Formatzeichenkettenangaben in der Datei **AdapterConfig_settings.properties** werden beim Import im Graphformat verwendet. Zum Einlesen von Zeitdaten, die vom PPM-Standardformat abweichen, ist ab PPM 3.0 das XML-Attribut **format** des Attribut-Mapping vorgesehen.

Als Erweiterung zum Java-Standard (siehe **Definition des Attribut-Mapping** (Seite 20), Abschnitt **Transformationen**) bietet PPM mit der Formatzeichenkette **Q.yyyy** die Möglichkeit, Zeitwerte als Quartal einzulesen, z. B. **2.2003** für das zweite Quartal 2003.

4.3.1 Sonderzeichen in XML-Dokumenten

Die Zeichen **&**, **>**, **<** und **"** sind Steuerzeichen (Meta-Zeichen) des XML-Dokuments. Kommen diese Zeichen direkt in den Quelldaten vor, führt dies beim Einlesen der XML-Datei zu Fehlinterpretationen. Um diese Zeichen dennoch in den Nutzdaten verwenden zu können, müssen sie durch folgende Entities (Zeichenfolgen) ersetzt werden:

Zeichen	Entity	Zeichen	Entity
&	&amp;	"	&quot;
<	&lt;	>	&gt;

Die folgende Tabelle zeigt die Darstellung von Sonderzeichen in XML-Dokumenten:

Zeichen	Entity	Zeichen	Entity	Zeichen	Entity
¡	&iexcl;	»	&raquo;	£	&pound;
¥	&yen;	|	&brvbar;	§	&sect;
©	&copy;	ª	&ordf;	«	&laquo;
®	&reg;	º	&deg;	±	&plusmn;
¹	&sup1;	²	&sup2;	³	&sup3;
¼	&frac14;	½	&frac12;	¾	&frac34;
Ä	&Auml;	Ö	&Ouml;	Ü	&Uuml;
ä	&auml;	ö	&ouml;	ü	&uuml;
ß	&szlig;	×	&times;	÷	&divide
È	&Egrave;	É	&Eacute;	Ê	&Ecirc;
Ë	&Euml;	è	&Egrave;	é	&Eacute;
ê	&Ecirc;	ë	&Euml;	Ø	&Oslach;

Sie können jedes beliebige Zeichen durch Angabe seines Zeichensatzcodes verwenden. Geben Sie hierfür den ASCII-Code des gewünschten Zeichens in Dezimalschreibweise wie folgt an: **&#<Dezimalcode des Zeichens>;**.
Um z. B. das Zeichen @ zu verwenden, geben Sie **@** im XML-Dokument an.

4.4 Erzeugen der Prozessinstanzfragmente

GRAPH-FORMAT

Beim Einlesen in das PPM-System werden die in der XML-Ausgabedatei enthaltenen Fragmentinstanzdaten eingelesen und Prozessinstanzfragmente erzeugt.

EVENT-FORMAT

Beim Einlesen in das PPM-System werden die in der XML-Ausgabedatei enthaltenen Instanzdaten der System-Events auf die Prozessfragmentdefinitionen der angegebenen Fragmentdatei abgebildet und Prozessinstanzfragmente erzeugt. Anschließend werden die in der Mapping-Datei angegebenen Attribute an die generierten Prozessinstanzfragmente kopiert.

Unabhängig vom zum Import der Fragmentinstanz verwendeten Format werden vor dem abschließenden Speichern eines Prozessinstanzfragments in der PPM-Datenbank Prozessschlüssel (siehe **Prozessfragmente zusammenführen** (Seite 90)) berechnet, damit die Prozessinstanzfragmente zu einer Prozessinstanz zusammengeführt werden können.

Warnung

Für jedes importierte Prozessinstanzfragment muss mindestens ein Prozessschlüssel berechnet werden. Prozessinstanzfragmente, für die kein Prozessschlüssel berechnet werden kann, werden nicht in der PPM-Datenbank gespeichert. Solche Fragmente gelten nicht als Bestandteil eines Gesamtprozesses.

Während des Imports werden auf der Konsole und optional in der angegebenen Protokolldatei entsprechende Statusmeldungen ausgegeben.

4.4.1 Erweiterung der Attributkonfiguration

Sie können in den XML-Importdateien vorhandene Attribute, die dem PPM-System nicht bekannt sind, automatisch identifizieren. Optional können Sie den neu hinzugekommenen Attributen einen Datentyp zuweisen und automatisch in die Attributkonfiguration Ihres PPM-Mandanten übernehmen.

Die automatische Attributerkennung unterstützt beide Datenimportformate.

Sie können folgende Möglichleiten verwenden, um die automatische Erweiterung der Attributkonfiguration zu aktivieren:

KOMMANDOZEILENARGUMENTE

Geben Sie an der Kommandozeile bei Aufruf von **runxmlimport** zusätzlich die Argumente **-autoextendattributes** und **-extractattributes** an.

Die Angaben der genannten Argumente wirken sich beim Import wie folgt aus:

Argument	Auswirkung
-extractattributes <Dateiname>	Unbekannte Attribute werden in der angegebenen Datei gespeichert. Es werden keine XML-Fragmentdaten importiert.
-autoextendattributes	Unbekannte Attribute werden in die Attributkonfiguration übernommen. Wenn die Option **automapping** eingeschaltet ist, wird die Konfiguration des Attribut-Mapping entsprechend erweitert. Anschließend werden die XML-Fragmentdaten importiert.
beide Argumente sind angegeben	Nacheinander werden die Aktionen wie unter -autoextendattributes und -extractattributes beschrieben ausgeführt.

DATENQUELLENDATEI

Weisen Sie dem Attribut **autoextentattributes** des XML-Elements **attributsettings** der verwendeten **datasource**-Datei den Wert **true** zu.

Beispiel

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

<datasource name="Events" type="EVENT">
...
  <attributesettings autoextendattributes="true">
...
  </attributesettings>
</datasource>
```

Bestimmte Attribute können Sie durch Mustererkennung von der automatischen Attributerkennung ausschließen. Geben Sie hierfür im XML-Element **excludepattern** die gewünschten Namensmuster der Attribute an, die Sie von der automatischen Erweiterung ausschließen möchten. Als Platzhalter können Sie die Zeichen **?** (einzelnes beliebiges Zeichen) und ***** (mehrere beliebige Zeichen) verwenden.

Beispiel

Im folgenden Beispiel werden alle System-Event-Attribute, deren Name mit **TEST** beginnt oder den Namen **USER** hat, von der automatischen Attributerkennung ausgeschlossen.

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

```
<datasource name="Events" type="EVENT">
...
  <attributesettings autoextendattributes="true">
    <excludepatterns>
      <excludepattern>TEST*</excludepattern>
      <excludepattern>USER</excludepattern>
    </excludepatterns>
...
  </attributesettings>
</datasource>
```

4.4.1.1 Datentyp unbekannter Attribute bestimmen

Sie können den Datentyp, der einem unbekannten Attribut zugewiesen werden soll, auf folgende Arten bestimmen lassen:

- Meta-Daten, die in der System-Event-Datei angegeben sind (nur bei Event-Importformat)
- Mustererkennung des Attributnamens, z. B. **TEXT_***
- Parsen des Attributwertes

Die gezeigt Reihenfolge entspricht der Priorisierung der automatischen Datentyperkennung.

Warnung

Der Datentyp bereits existierender PPM-Attribute kann nicht geändert werden.

META-DATEN

In den zu importierenden System-Event-Dateien können Sie für jedes Attribut eines System-Events einen PPM-Datentyp sowie optional ein Format (z. B. zur Erkennung von Zeitstempel) und Beschreibung eines Attributes angeben. Die im PPM-System noch unbekannten Attribute werden automatisch mit dem angegebenen Datentyp erzeugt.

Meta-Daten werden im XML-Element **metadata** angegeben:

```
<eventlist>
  <metadata>
    <attr_desc type="...">
      <ppmdatatype>...</ppmdatatype>
      <format>...</format>
      <description>...</description>
    </attr_desc>
...
  </metadata>

  <event>
...
  </event>
...
</eventlist>
```

Die XML-Tags des XML-Elements **metadata** haben folgende Bedeutung:

XML-Tag	Beschreibung
attr_desc	einzelne Meta-Datendefinition eines System-Event-Attributes, es können mehrer Definitionen angegeben werden.
type	Name des System-Event-Attributes
ppmdatatype	PPM-Datentyp, der dem erzeugten PPM-Attribut zugewiesen werden soll.
format (optional)	Format, das zum Parsen des Attributwertes verwendet werden soll. Wenn Sie zusätzlich automatisches Mapping aktiviert haben, wird das Format auch dazu verwendet, den Wert in das PPM-Zeitformat zu transformieren.
description (optional)	Beschreibung des erzeugten PPM-Attributs. Die angegebene Beschreibung wird in der Attributkonfiguration in der Standardsprache des PPM-Mandanten übernommen.

Beispiel

```
<eventlist>
  <metadata>
    <attr_desc type="SWWWIHEAD-WI_ID">
      <ppmdatatype>TEXT</ppmdatatype>
      <description>Work item ID</description>
    </attr_desc>
    <attr_desc type="SWWWIRET-WI_AED">
      <ppmdatatype>DAY</ppmdatatype>
      <format>yyyyMMdd</format>
      <description>End Date of Work Item</description>
    </attr_desc>
  </metadata>

  <event>
    <attribute type="SWWWIHEAD-WI_ID">000000525723</attribute>
    <attribute type="SWWWIRET-WI_AED">20011211</attribute>
  </event>
</eventlist>
```

Die für PPM erhältliche Extraktoren Process Extractor SAP-2-PPM und Process Extractor JDBC-2-PPM erzeugen automatisch das XML-Element **metadata**, so dass Sie ohne Zutun die erzeugten System-Event-Dateien direkt importieren können. Die im PPM-System noch unbekannten Attribute werden automatisch mit dem richtigen Datentyp erzeugt.

MUSTERERKENNUNG

Im XML-Element **datatypekeydetectionsettings** der Datenquellendatei können Sie eine Liste von Muster angeben (XML-Elemente **datatypedetectionpattern**), die zur Datentypzuweisung verwendet werden. Die Namen aller neu erkannten Attribute werden mit den Mustern verglichen. Das erste erkannte Muster der Liste bestimmt den Datentyp des Attributes.

Beispiel

Der folgende Auszug der datasource-Datei konfiguriert folgende Namensmuster zur Bestimmung des Datentyps neu erkannter Attribute:

- Einem System-Event-Attribut, dessen Namen mit **L_** beginnt oder auf **L** endet, wird der Datentyp **LONG** zugewiesen.
- Einem System-Event-Attribut, dessen Namen mit **D** beginnt und endet, wird der Datentyp **DOUBLE** zugewiesen.

```
...
  <attributesettings autoextendattributes="true">
    <datatypedetectionsettings>
      <datatypekeydetectionsettings >
        <datatypedetectionpattern datatype=”LONG”>*L
        </datatypedetectionpattern>
        <datatypedetectionpattern datatype=”LONG”>L_*
        </datatypedetectionpattern>
        <datatypedetectionpattern datatype=”DOUBLE”>D*D
        </datatypedetectionpattern>
      </datatypekeydetectionsettings>
    </datatypedetectionsettings>
  </attributesettings>
...
```

PARSEN DES ATTRIBUTWERTES

Wenn durch die Mustererkennung kein Datentyp bestimmt werden konnte, wird versucht, den Datentyp aus dem Wert des Attributes zu bestimmen. Folgende Datentypen werden erkannt: **BOOLEAN**, **LONG**, **DOUBLE**, **TIME**, **TIMEOFDAY**, **DAY** und **TEXT**. Die Aufzählung entspricht auch der Priorisierung der Datentyperkennung.

DATENTYP BOOLEAN

Unabhängig von Groß- Kleinschreibung werden die Attributwerte **true** und **false** dem Datentyp **BOOLEAN** zugeordnet. Andere Werte, z. B. **0** und **1** werden nicht als **BOOLEAN** erkannt.

DATENTYPEN LONG UND DOUBLE

Wurde Datentyp **BOOLEAN** nicht erkannt, wird versucht, numerische Werte zu erkennen. Bei der Erkennung numerischer Datentypen wird zwischen Ganzzahlen und Fließkommazahlen unterschieden. Im Attribut **doubleonly** des XML-Elements **datatypevaluedetectionsettings** können Sie mit Wert **TRUE** (Default-Wert) angeben, dass ganzzahlige Attributwerte dem Datentyp **DOUBLE** zugewiesen werden.

Die Erkennung des Datentyps **DOUBLE** erfolgt ohne Angabe eines Tausendertrennerzeichens. Als Dezimaltrenner wird ausschließlich der Punkt erkannt.

DATENTYPEN TIME, TIMEOFDAY UND DAY

Wurde kein numerischer Datentyp erkannt, wird versucht, den Attributwert als Zeitstempel (**TIME**), Tageszeit (**TIMEOFDAY**) oder Datum (**DAY**) zu erkennen. Die Aufzählung entspricht auch der Priorisierung der Datentyperkennung. Für die Datentypen Zeitstempel, Tageszeit und Datum können Sie jeweils optional in den XML-Elementen **timeformat**, **timeofdayformat** und

dayformat Formatzeichenketten angeben, die die Formate der angegebenen Attributwerte beschreiben. Wenn Sie keine Formatzeichenketten angeben, gelten folgende Standardformate:

Datentyp	Standardformat
TIME	dd.MM.yyyy HH:mm:ss
TIMEOFDAY	dd.MM.yyyy
DAY	HH:mm:ss

DATENTYP TEXT

Konnte bisher durch Parsen des Attributwertes kein Datentyp erkannt werden, wird dem Attribut der Datentyp **TEXT** zugewiesen.

VERHALTEN BEI MEHRDEUTIGEN DATENTYPEN

Im Attribut **numberofvaluestocheck** des XML-Elements **datatypevaluedetectionsettings** können Sie angeben, wie oft der Datentyp durch Parsen des Attributwertes bestimmt werden soll. Ist das XML-Attribut nicht angegeben, wird der Standardwert **100** verwendet. Werden bei der Bestimmung unterschiedliche Datentypen für ein Attribut erkannt, wird der zuvor erkannte Datentyp in einen allgemeinen Datentyp gewandelt. Dabei gilt folgende Tabelle:

Datentyp	Allgemeiner Datentyp
BOOLEAN	TEXT
LONG	DOUBLE, TEXT
DOUBLE	TEXT
TIME	TEXT
TIMEOFDAY	TEXT
DAY	TEXT
TEXT	TEXT

Beispiel

Der folgende Dateiauszug zeigt eine vollständige Konfiguration automatischer Datentypbestimmung durch Parsen des Attributwertes.

```
...
  <attributesettings autoextendattributes="true">
    <excludepatterns>
      <excludepattern>TEST*</excludepattern>
      <excludepattern>USER</excludepattern>
    </excludepatterns>
    <datatypedetectionsettings>
      <datatypekeydetectionsettings>
        <datatypedetectionpattern datatype = "LONG">LG_*
```

```
      </datatypedetectionpattern>
      <datatypedetectionpattern datatype = "LONG">*LONG*
      </datatypedetectionpattern>
      <datatypedetectionpattern datatype = "LONG">*_lng
      </datatypedetectionpattern>
      <datatypedetectionpattern datatype = "DOUBLE">DOUBLE_*
      </datatypedetectionpattern>
      <datatypedetectionpattern datatype = "DOUBLE">*_dbl
      </datatypedetectionpattern>
      <datatypedetectionpattern datatype = "DAY">DAY_*
      </datatypedetectionpattern>
      <datatypedetectionpattern datatype = "DAY">*_day
      </datatypedetectionpattern>
      <datatypedetectionpattern datatype = "TIMEOFDAY">TIMEOFDAY_*
      </datatypedetectionpattern>
      <datatypedetectionpattern datatype = "TIMEOFDAY">*_tod
      </datatypedetectionpattern>
      <datatypedetectionpattern datatype = "TIME">TIME_*
      </datatypedetectionpattern>
      <datatypedetectionpattern datatype = "TIME">*_tm
      </datatypedetectionpattern>
      <datatypedetectionpattern datatype = "TEXT">TEXT_*
      </datatypedetectionpattern>
      <datatypedetectionpattern datatype = "TEXT">*_txt
      </datatypedetectionpattern>
    </datatypekeydetectionsettings>
    <datatypevaluedetectionsettings
       doubleonly = "FALSE"
       numberofvaluestocheck = "100">
      <timeformat>dd.MM.yyyy HH:mm:ss</timeformat>
      <timeofdayformat>HH:mm:ss</timeofdayformat>
      <dayformat>dd.MM.yyyy</dayformat>
    </datatypevaluedetectionsettings>
  </datatypedetectionsettings>
  <attributeprefix>AT_</attributeprefix>
 </attributesettings>
...
```

4.4.2 Erweiterung der Mapping-Konfiguration

Wenn Sie zum Importieren das Event-Format verwenden, können Sie automatisches Mapping aktivieren, d. h. alle Attribute eines System-Event werden an die angegebenen Objekttypen der zugeordneten Fragmentdefinition übertragen.

Die Erweiterung der Mapping-Konfiguration wird in Attributen des XML-Elementes **automapping** konfiguriert.

XML-Attribut	Beschreibung
nodetype	bestimmt den Objekttyp, für den dieses **automapping** gilt. Mögliche Werte: **OT_FUNC** für Funktionen, **OT_EVT** für Ereignisse und **PROCESS** für Prozesse.

XML-Attribut	Beschreibung
graphid	bestimmt den Fragmentdefinitionsgaphen, für dessen Objekte dieses **automapping** gilt. Der angegebene Wert entspricht dem Attribut **id** des XML-Elements **graph** der Fragmentdefinition.
addmergeattributes (optional)	bestimmt bei Objekttyp **PROCESS**, ob die Attribute der Merger-Konfiguration hinzugefügt werden, so dass diese beim Zusammenführen von Instanzfragmenten als Prozessattribute erhalten bleiben (Wert **TRUE**) oder nicht (Wert **FALSE**). Für die Objekttypen **OT_FUNC** und **OT_EVT** wird die Angabe ignoriert. Für Objekttyp **PROCESS** muss **addmergeattributes** angegeben werden!

Wenn Sie automatische Mapping-Erweiterung für mehrere Objekttypen oder Fragmentdefinitionsgraphen angeben müssen, können Sie für jeden gewünschten Objekttyp bzw. Fragmentdefinitionsgraphen jeweils ein eigenes XML-Element **automapping** angeben. Das automatische Mapping berücksichtigt das im XML-Element **attributprefix** angegebene Präfix, z. B. **AT_**. Wenn Sie für bestimmte Attribute ein explizites Mapping angegeben haben, werden die zuvor durch automatisches Mapping übertragenen Attribute überschrieben.

Beispiel

Im folgenden Dateiauszug wird automatische Erweiterung des Mapping für Funktionen des Fragmentdefinitionsgraphen **FRG_CATCH_ALL** konfiguriert.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE eventmapping SYSTEM "eventmapping.dtd">
<eventmapping>
  <processfragmentmapping>
    <!--FRG_CATCH_ALL-->
    <processfragment graphid="FRG_CATCH_ALL">
...
    </processfragment>
...
  </processfragmentmapping>
  <attributemapping>
...
    <automapping
      nodetype="OT_FUNC"
      graphid="FRG_CATCH_ALL"
    />
  <attributemapping>
</eventmapping>
```

4.4.3 Mehrwertige System-Event-Attribute

Wenn ein System-Event-Attribut mehrfach in einem System-Event vorkommt, ist nicht sichergestellt, dass der zuletzt gelesene Wert übernommen wird. Wenn Sie alle Werte von mehrfach im System-Event enthaltener Attribute übernehmen möchten, können Sie hierfür in der

Mapping-Konfiguration anstelle des XML-Elementes **attribute** das XML-Element **multieventattributetype** angeben. Alle Werte so gekennzeichneter System-Event-Attribute werden mit einem Trennzeichen (Standard ist Semikolon) konkateniert. Ein vom Standard abweichendes Trennzeichen geben Sie im optionalen XML-Attribute **delimiter** an.

Die Verwendung von mehrfachen System-Event-Attributwerten vereinfacht das Zusammenführen von parallelen Prozesspfaden, wenn Sie alle Merge-Schlüssel des Vorgängerfragments als **multieventattributetype**-Attribute in die System-Event-Datei schreiben.

Beispiel

Auszug System-Event-Datei

```
...
<event>
  <attribute type="EVENTTYP">Change customer order</attribute>
  <attribute type="THIS_KEY">3</attribute>
  <attribute type="PREV_KEY">1</attribute>
  <attribute type="PREV_KEY">2</attribute>
  <attribute type="USER">Team A</attribute>
</event>
...
```

Auszug Mapping-Datei

```
...
  <attributemapping>
...
    <!-- Mapping Event-Start BEGIN -->
    <objectattributes objectname="EVT_START" graphid="FRG_CATCH_ALL">
      <attribute ppmattributetype="AT_OBJNAME">
        <eventattributetype>EVENTTYP</eventattributetype>
        <value> to be done</value>
      </attribute>
      <attribute ppmattributetype="AT_ID">
        <eventattributetype>AT_PRCNO</eventattributetype>
      </attribute>
      <!-- Mulivalue Mapping -->
      <attribute ppmattributetype="AT_KEY">
        <multieventattributetype delimiter=";">PREV_KEY</multieventattributetype>
      </attribute>
...
    </objectattributes>

...
```

Das am Ereignis **EVT_START** erzeugt Attribut **AT_KEY** erhält den Wert **1;2**. Bei entsprechender Konfiguration des Merger werden für dieses Ereignis zwei Merge-Schlüssel erzeugt.

4.4.4 Direkter Import von Prozessattributen

Die zur Berechnung von Kennzahlen und Dimensionen auf Prozessinstanzebene benötigten Attribute werden in der Regel von Objekten der importierten Fragmentinstanzen an die Prozessinstanz kopiert. Dieser Vorgang kann durch Attributkopierregeln oder eine entsprechende Berechnungsvorschrift des Kennzahlenberechners konfiguriert werden.

Alternativ können Sie Prozessinstanzattribute direkt importieren (Graph-Format) oder durch Mapping von System-Event-Attributen an die Fragmentdefinition erzeugen (Event-Format).

Um Prozessattribute bereits existierender Prozessinstanzen zu überschreiben, können Sie Fragmentinstanzen ohne Objekte und Kanten, die ausschließlich Prozessattribute enthalten, direkt importieren (Graph-Format) oder durch Mapping von System-Event-Attributen an eine objektlose Fragmentdefinition erzeugen (Event-Format).

GRAPH-FORMAT

Beispiel

Graph ohne Objekte und Kanten

```
...
  <graph id="FRG_EMPTY">
    <attribute type="AT_ID">1</attribute>
    <attribute type="AT_SAP_BELEGNR">Beleg 2</attribute>
    <attribute type="AT_SAPCLIENT">R3</attribute>
  </graph>
...
```

EVENT-FORMAT

Bei Verwendung des Event-Format werden die System-Event-Attribute an den instanziierten Graphen einer objektlosen Fragmentdefinition übertragen.

Beispiel

Auszug objektlose Fragmentdefinition:

```
...
  <graph id="FRG_EMPTY">
     <attribute type="AT_ID">1</attribute>
  </graph>
...
```

Auszug Mapping-Regel:

```
...
<processattributes graphid="FRG_EMPTY">
  <attribute ppmattributetype="AT_SAP_BELEGNR">
    <value>Beleg </value>
    <eventattributetype>SAP_BELEGNR</eventattributetype>
  </attribute>
  <attribute ppmattributetype="AT_SAPCLIENT">
    <eventattributetype>SAPCLIENT</eventattributetype>
  </attribute>
</processattributes>
...
```

Warnung

Damit zu importierende objektlose Fragmente Prozessinstanzen zugeordnet werden können, müssen Sie darauf achten, dass gültige Prozessschlüssel auf Basis von Prozessattributen berechnet werden können.

Warnung

Damit direkt importierte Prozessattribute an der Instanz erhalten bleiben, müssen Sie jedes direkt importierte Attribut im XML-Element **mergeattributes** der Merger-Konfiguration angeben.

Beispiel merger_config.xml

```
...
   <processmerge>
     <mergeattributes>
       <attribute key = "AT_SAP_BELEGNR"/>
       <attribute key = "AT_SAPCLIENT"/>
     </mergeattributes>
   </processmerge>
...
```

4.4.5 Sonderfall skaliertes System

In einem skalierten System erfolgt der Datenimport an den Sub-Server. Alle Sub-Server müssen identisch konfiguriert sein. Die Optionen **autoextendattributes** und **addmergeattributes** würden die Konfiguration eines einzelnen Sub-Servers verändern und die einheitliche Konfiguration aller Sub-Server ginge verloren. Die Verwendung dieser beiden Optionen ist deshalb im skalierten System nicht möglich.

Falls Sie die automatische Erweiterung der Attributkonfiguration dennoch verwenden möchten, müssen Sie für jeden Sub-Server die möglichen Erweiterungen der Konfiguration exportieren und manuell am Master-Server einspielen, der die Erweiterungen anschließend an alle Sub-Server des Systems verteilen kann.

VORGEHEN

Bevor Sie Daten mit neuen Attributen an den Sub-Server importieren, führen Sie folgende Schritte durch:

1. Identifizieren Sie neue System-Event-Attribute und extrahieren Sie diese durch Parameter **-extractattributes <Dateiname>** des Kommandozeilenprogramms **xmlimport**.
2. Importieren Sie die neuen Attribute am Master-Server.
3. Identifizieren Sie neue Attribute, die beim Zusammenführen von Prozessfragmenten erhalten bleiben sollen, und extrahieren Sie diese durch Parameter **-extractmergeattributes <Dateiname>** des Kommandozeilenprogramms **xmlimport**.
4. Erweitern Sie manuell die Merger-Konfiguration des Master-Server, wie im folgenden Abschnitt **Hinzufügen der Merge-Attribute** beschrieben.
5. XML-Datenimport an den Sub-Server mit Option **automapping** durchführen.

HINZUFÜGEN DER MERGE-ATTRIBUTE

Wenn Sie automatische Erweiterung des Mappings für Prozesse angegeben haben (**<auotmapping nodetype="PROCESS" ...**), können Sie die neu hinzugekommenen Attribute, die beim Zusammenführen von Prozessfragmenten erhalten bleiben sollen, extrahieren, indem

Sie an der Kommandozeile den Parameter **-exctractmergeattributes <Dateiname>** angeben. Es werden keine XML-Daten importiert. Die erzeugt Datei enthält ausschließlich ein XML-Element **mergeattributes** mit einer Liste aller neuen Attribute und hat folgende Struktur:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<mergeattributes>
     <attribute key="..." />
...
</mergeattributes>
```

Exportieren Sie mittels Parameter **-export -merger <Dateiname>** die Merger-Konfiguration des Master-Server. Bearbeiten Sie die Merger-Konfiguration, indem Sie die Attribute (XML-Elemente **attribute** des XML-Elements **mergeattributes**) aller zuvor für jeden Sub-Server mittels **-exctractmergeattributes <Dateiname>** exportierten Merge-Attribute der exportierten Merger-Konfiguration des Master-Server hinzufügen.

Importieren Sie abschließend mittels Parameter **-import -merger <Dateiname>** die erweiterte Merger-Konfiguration des Master-Server.

Beispiel

Exportierte Merger-Konfiguration des Master-Server

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE mergerconfig SYSTEM "mergerconfig.dtd">
<mergerconfig>
  <mergehandling>
    <processmerge>
      <mergeattributes>
        <attribute key = "AT_SAPSYSTEM"/>
        <attribute key = "AT_SAP_BELEGNR"/>
      </mergeattributes>
    </processmerge>
    <eventmerge priority="1">
      <mode>
        <keymerge />
      </mode>
    </eventmerge>
  </mergehandling>
</mergerconfig>
```

Exportierte Merge-Attribute Sub-Server 1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<mergeattributes>
  <attribute key = "AT_SAPCLIENT"/>
</mergeattributes>
```

Exportierte Merge-Attribute Sub-Server 2

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<mergeattributes>
  <attribute key="AT_SAP_BSTYP" />
  <attribute key="AT_SAP_BSTYP" />
</mergeattributes>
```

Konsolidierte Merger-Konfiguration des Master-Server

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

```
  <mergehandling>
    <processmerge>
      <mergeattributes>
        <attribute key = "AT_SAPSYSTEM"/>
        <attribute key = "AT_SAP_BELEGNR"/>
  <!-- Merge-attributes sub-server 1 -->
        <attribute key = "AT_SAPCLIENT"/>
  <!-- Merge-attributes sub-server 2 -->
        <attribute key="AT_SAP_BSTYP" />
        <attribute key="AT_SAP_BSTYP" />
      </mergeattributes>
    </processmerge>
    <eventmerge priority="1">
      <mode>
        <keymerge />
      </mode>
    </eventmerge>
  </mergehandling>
</mergerconfig>
```

4.4.6 Archivierung der XML-Importdateien

Wenn Sie verhindern möchten, dass bereits importierte XML-Importdateien beim nächsten Aufruf nochmals importiert werden, können Sie angeben, dass die importierten Dateien umbenannt oder in ein anderes Verzeichnis verschoben werden.

Die Möglichkeit, Importdateien zu archivieren, konfigurieren Sie im XML-Element **archive** der verwendeten Datenquellendatei:

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

<datasource name="Events" type="EVENT">
...
  <archive>
    <directory>...</directory>
    <prefix>...</prefix>
  </archive>
</datasource>
```

XML-Element	Beschreibung
directory	bestimmt das Verzeichnis, in das die importierten XML-Importdateien verschoben werden. Nicht existierende Verzeichnisse des angegebenen Pfades werden automatisch erzeugt.
prefix	bestimmt das Präfix, mit dem die Namen der importierten XML-Importdateien versehen werden.

Wenn Sie beide XML-Attribute angeben, wird das Attribut **directory** bevorzugt ausgewertet, das Attribut **prefix** wird ignoriert.

4.5 Kommandozeilenprogramm runxmlimport

Der Import von XML-Importdateien erfolgt über das Kommandozeilenprogramm **runxmlimport**. Beim Import werden Prozessinstanzfragmente in den internen Puffer des PPM-Systems eingelesen und zunächst nicht weiterverarbeitet.

Warnung

Achten Sie darauf, dass vor dem XML-Import der PPM-Server gestartet wurde. Es kann immer nur ein Kommandozeilenprogramm aktiv sein.

Tritt während des Imports ein Fehler auf, erscheint eine Fehlermeldung auf der Konsole und der Rückgabewert des Programms ist ungleich **0**.

Der Aufruf des Programms ohne Parameter oder mit **-h** oder **-?** gibt die Online-Hilfe auf der Konsole aus. Die Hilfe beschreibt alle verfügbaren Optionen:

IMPORTIEREN VON PROZESSINSTANZEN IM XML-FORMAT

runxmlimport -user <username> -password <password> [-client <name>]
[-datasource <file>]
[-datasourcelist <file>]
[-i <file1>[,<file2>...]]
[-f <fragment file> -m <mapping file>]
[-extractattributes <attribute file>] [-autoextendattributes]
[-extractmergeattributes <mergeattribute file>]
[protocoloptions] [-language <ISO-code>] [-version]

-user <username>	Name des Benutzers
-password <password>	Kennwort des Benutzers
-client <name>	Name des Mandanten. Wird kein Mandant angegeben, wird der Importvorgang für den Standardmandanten durchgeführt.
-version	Versionsnummer
-datasource <file>	Datenquelle, die für den XML-Import verwendet werden soll.
-datasourcelist <file>	Liste mit Datenquellen, die für den XML-Import verwendet werden sollen.
-i <file1> [,<file2>...]	Importdateien. ZIP-Dateien werden unterstützt.

Erfolgt der Import im Ereignisformat, müssen -f und -m spezifiziert werden, falls keine Datenquelle oder Datenquellenliste angegeben ist.

[-f <fragment file>]	Fragmentdatei
[-m <mapping file>]	Mapping-Datei
[-extractattributes <attribute file>]	Generiert eine XML-Datei mit allen Attributtypen, die in der/den Eingabedatei(en) vorkommen, aber noch nicht definiert wurden.
[-autoextendattributes]	Vor dem XML-Import werden alle Attribut- typen, die in der/den Eingabedatei(en) vorkommen aber noch nicht definiert wurden, importiert.
[-extractmergeattributes <mergeattribute file>]	Generiert einen Auszug der Merge-Konfigurationsdatei mit allen neuen Attributen, die beim Zusammenführen von Fragmentinstanzen als Prozessattribute erhalten bleiben sollen.

Die Option **protocoloptions** kann aus folgenden Anweisungen bestehen:

-protocolfile <filename>	Protokollieren in Datei <filename>
-information {yes|no|default}	Protokollieren von Informationen
-warning {yes|no|default}	Protokollieren von Warnungen
-error {yes|no|default}	Protokollieren von Fehlern
-language <ISO-code>	Sprache der Protokollausgaben

4.5.1 Argumente von runxmlimport

-VERSION

Auf der Konsole werden die Version der PPM-Software und des Datenbankschemas ausgegeben. Andere Argumente werden ignoriert.

-USER <BENUTZERNAME> -PASSWORD <KENNWORT>

Hier geben Sie den Benutzernamen und das Kennwort des PPM-Benutzers an, der den Import ausführt. Der Benutzer muss das Funktionsrecht **Datenimport** haben.

-CLIENT <MANDANTENNAME>

Hier geben Sie den PPM-Mandanten an, in dessen Datenbankschema die importierten Prozessfragmente gespeichert werden. Wenn Sie diese Eingabe nicht vornehmen, wird der Standardmandant **Default** verwendet.

-I <NAME DER XML-DATEIEN>

Hier können Sie beliebig viele Dateien angeben, die importiert werden sollen. Die Namen dürfen auch Platzhalter enthalten. **?** steht für genau ein Zeichen, ***** für eine beliebige Anzahl von Zeichen. Der Dateiname darf nicht mit einem Minuszeichen beginnen.

Beispiel: Das Argument **-i <Pfad>\?ata*.xml** liest alle Dateien im Verzeichnis **<Pfad>** mit der Erweiterung **<.xml**, die mit einem beliebigen Zeichen beginnen, dem die Zeichenfolge **<ata** und danach eine beliebige Zeichenfolge folgt.

Handelt es sich bei den spezifizierten Dateien um ZIP-Archive, werden alle XML-Dateien im ZIP-Archiv unabhängig von der Verzeichnisstruktur des Archivs gelesen.

-M <NAME DER MAPPING-DATEI> -F <NAME DER FRAGMENTDATEI>

Hier legen Sie die Art des Imports fest. Wenn sie die Argumente **-f** und **-m** angeben, wird automatisch das PPM-System-Event-Format verwendet. Wenn Sie die Argumente nicht angeben, wird das Graphformat verwendet.

-DATASOURCE <DATEINAME>

Die in der Datei angegebenen XML-Elemente **data**, **fragments** und **mapping** spezifizieren die zu verwendenden XML-Dateien und ersetzen die Kommandozeilenparameter **-i**, **-f** und **-m**. Das Format der XML-Datei ist in der Dokumenttypdefinition **datasource.dtd** beschrieben. PPM Customizing Toolkit verwendet diese Datei zur Konfiguration und zum Auslesen von Datenquellen.

Die Datei liegt unter
<Installationsverzeichnis>\ppm\server\bin\agentLocalRepo\.unpacked\<Installationszeit>_ppm-client-run-prod-<Version>-runnable.zip\ppm\dtd

Die folgenden Beispiele veranschaulichen den Inhalt eine Datenquellendatei zum Datenimport im Graph- und System-Event-Format:

Datenquellendatei **datasource.xml** für Graph-Format

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE datasource SYSTEM "datasource.dtd">
<datasource name="Sales" type="GRAPH">
  <description name="default_description"
             language="de">Demo-Datenbankinhalt</description>
  <description name="default_description"
             language="en">Demo Database Content</description>

<data>C:\SoftwareAG\ppm\server\bin\work\data_ppm\custom\umg_en\data\umgsales_umg
_en.zip</data>
  <eventattributetypes />
</datasource>
```

Datenquellendatei **datasource.xml** für Event-Format

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE datasource SYSTEM "datasource.dtd">
<datasource name="CustomerService" type="EVENT">
  <description name="default_description"
             language="de">Demo-Datenbankinhalt</description>
  <description name="default_description"
             language="en">Demo Database Content</description>
```

```
<data>C:\SoftwareAG\ppm\server\bin\work\data_ppm\custom\umg_en\data\CS\csdemodat
a.zip</data>

<fragments>C:\SoftwareAG\ppm\server\bin\work\data_ppm\custom\umg_en\xml\CS\Custo
merService_Fragments.xml</fragments>

<mapping>C:\SoftwareAG\ppm\server\bin\work\data_ppm\custom\umg_en\xml\CS\Custome
rService_Mapping.xml</mapping>
  <eventattributetypes />
</datasource>
```

Wenn Sie den Parameter **-datasource** auf der Kommandozeile angeben, werden die Parameter **-i**, **-f** und **-m** nicht beachtet.

Weitere Informationen zur Konfiguration von Datenquellen finden Sie in der Technischen Referenz **PPM Process Extractors**.

-DATASOURCELIST <DATEINAME>

Mit Hilfe des Arguments **-datasourcelist** können mehrere Datenquellen, außer Datenquellen des Typs **GRAPH**, auf einmal importiert werden. Der Import entspricht dem mehrfachen, nacheinander Ausführen des Imports mittels des Arguments **-datasource**.

Siehe Kapitel Mehrere Datenquellen importieren (Seite 63).

-AUTOEXTENDATTRIBUTES

Wenn Sie diesen Schalter an der Kommandozeile angeben, werden neue Attribute erkannt und die Attributkonfiguration des PPM-Systems entsprechend erweitert. Wenn Sie in der verwendeten Datenquellenkonfiguration die Option **automapping** eingeschaltet haben, werden neu erkannte Attribute an die angegebenen Objekte der Fragmentdefinition übertragen.

-EXTRACTATTRIBUTES <DATEINAME>

In den XML-Importdateien enthaltene Attribute, die dem PPM-System noch nicht bekannt werden, werden identifiziert und in die angegebene Datei geschrieben. Die erzeugte Datei ist DTD-konform und kann mittels Kommandozeilenprogramm **runppmconfig** importiert werden, um die PPM-Attributkonfiguration zu erweitern. Wenn Sie das Argument **-extractattributes** ohne Schalter **-autoexentattributes** verwenden, werden keine XML-Importdaten eingelesen.

-EXTRACTMERGEATTRIBUTES <DATEINAME>

Dieses Argument bewirkt, dass per Option **automapping** an Prozessinstanzen übertragene Attribute in die angegebene Datei geschrieben werden. Es werden keine XML-Importdaten eingelesen.

PROTOKOLLOPTIONEN

Mit diesen Argumenten können Sie die Protokollausgaben einschränken. Fehlermeldungen, die zum Programmabbruch führen, werden in jedem Fall auch auf der Konsole ausgegeben.

4.6 Mehrere Datenquellen importieren

Mit Hilfe des Arguments **-datasourcelist <file>** können mehrere Datenquellen gleichzeitig importiert werden (siehe Kapitel Kommandozeilenprogramm runxmlimport (Seite 59)). Zum Import mehrere Datenquellen steht eine Konfigurationsdatei zur Verfügung, in der eine Liste mit Datenquellen angegeben werden kann. Beim XML-Import werden die Daten der in der Konfigurationsdatei angegebenen Datenquellen nacheinander importiert, als würde der XML-Import mehrmals hintereinander mit dem Argument **-datasource <file>** aufgerufen. Die Reihenfolge des Imports der Datenquellen wird in der Konfigurationsdatei angegeben.

Die Konfigurationsdatei muss der DTD **datasourcelist.dtd** entsprechen, die folgendermaßen aussieht.

```
<!ELEMENT datasourcelist (datasource*)>
<!ELEMENT datasource (#PCDATA)>
<!ATTLIST datasource
      name ID #REQUIRED
      type (EVENT | MYSAP | JDBC | CSV | NIRVANA ) #REQUIRED
>
```

Zu jeder Datenquelle ist eine ID, d. h. im Normalfall der Name der Datenquelle, der auch in CTK verwendet wird, der Datenquellentyp und der Pfad zur Datenquellendatei anzugeben.

Eine XML-Datei kann dann zum Beispiel folgendermaßen aussehen.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE datasourcelist SYSTEM "datasourcelist.dtd">
<datasourcelist>
      <datasource name="CLEARING"
type="EVENT">M:/SoftwareAG/ppm/server/bin/work/data_ppm/custom/umg_en/xml/CLEARI
NG/CLEARING.xml</datasource>
      <datasource name="BILLING" type="MYSAP">
M:/SoftwareAG/ppm/server/bin/work/data_ppm/custom/umg_en/xml/BILLING/BILLING.xml
</datasource>
      <datasource name="SHIPMENT" type="JDBC">
M:/SoftwareAG/ppm/server/bin/work/data_ppm/custom/umg_en/xml/SHIPMENT/SHIPMENT.x
ml</datasource>
      <datasource name="MATERIAL_DOCUMENT"
type="CSV">M:/SoftwareAG/ppm/server/bin/work/data_ppm/custom/umg_en/xml/MATERIAL
_DOCUMENT/MATERIAL_DOCUMENT.xml</datasource>
      <datasource name="PURCHASE_PROCESS" type="NIRVANA">
M:/SoftwareAG/ppm/server/bin/work/data_ppm/custom/umg_en/xml/PURCHASE_PROCESS/PU
RCHASE_PROCESS.xml</datasource>
</datasourcelist>
```

FEHLERVERHALTEN

Wird der XML-Import mit einer gültigen Konfigurationsdatei aufgerufen, die aber keine Datenquellen enthält, so beendet er sich ohne eine Fehlermeldung.

Wird der XML-Import mit einer Konfigurationsdatei aufgerufen, die mehrere Datenquellen enthält und tritt beim Import einer Datenquelle ein Fehler auf, der zum Abbruch dieses einen Imports führt, so wird der Import mit der nächsten Datenquellendatei fortgeführt. D. h. der Abbruch des Imports einer Datenquellendatei führt nicht zu einem Komplettabbruch des Imports.

Tritt beim Import mindestens einer Datenquelle aus einer Konfigurationsdatei ein Fehler auf, der bislang, d. h. beim Import der einzelnen Datenquellen, zu einem Exit-Fehlerstatus (d. h. „-1")

geführt hat, so liefert auch der Import mit der Konfigurationsdatei diesen Exit-Fehlerstatus zurück.

4.7 Erneuter Import derselben Daten

Im PPM-System führt wiederholter Import derselben Quelldaten immer zu demselben eindeutigen Ergebnis.

4.7.1 Graphformat

Beim erneuten Import der Instanzdaten mittels Graphformat werden die Prozessinstanzen anhand des Attributes **AT_EPK_KEY** eindeutig identifiziert. Vorhandene Prozessinstanzen werden von importierten Prozessinstanzen mit demselben Attributwert überschrieben.

4.7.2 System-Event-Format

Beim erneuten Import der Instanzdaten mittels System-Event-Format werden identische Objekte automatisch überschrieben. Identische Objekte werden anhand des identischen internen Objektschlüssels erkannt, der während des Einlesens berechnet und im Objektattribut **AT_INTERNAL_OBJECT_KEY** gespeichert wird. Im XML-Element **internalobjectkeyrules** der Datei **keyrules.xml** werden Regeln zur Berechnung der Objektschlüssel angegeben. Bei identischen Objekten wird das zuletzt importierte Objekt in die Prozessinstanz übernommen.

BEISPIEL

Folgender Dateiauszug definiert Regeln zur Berechnung der Objektschlüssel von Funktionen und Ereignissen. Ereignisse werden als identisch erkannt, wenn die Werte der Attribute **AT_OBJNAME_INTERN** und **AT_MERGE_KEY_1** übereinstimmen. Funktionen werden als identisch erkannt, wenn die Werte der Attribute **AT_OBJNAME_INTERN** und **AT_END_TIME** übereinstimmen.

```
...
  <internalobjectkeyrule>
    <refobjects>
      <refobject objecttype="OT_EVT"/>
    </refobjects>
    <keyparts>
      <keypart attributetype="AT_OBJNAME_INTERN"/>
      <keypart attributetype="AT_MERGE_KEY_1"/>
    </keyparts>
  </internalobjectkeyrule>
  <internalobjectkeyrule>
    <refobjects>
      <refobject objecttype="OT_FUNC"/>
    </refobjects>
    <keyparts>
      <keypart attributetype="AT_OBJNAME_INTERN"/>
      <keypart attributetype="AT_END_TIME"/>
    </keyparts>
  </internalobjectkeyrule>
...
```

Warnung

Definieren Sie interne Objektschlüsselregeln, die sicherstellen, dass für identische Objekte der Prozessinstanz identische Objektschlüssel berechnet und diese Objekte somit beim erneuten Import überschrieben werden.

5 Import prozessinstanzunabhängiger Daten

Dieses Kapitel beschreibt die Importschnittstelle zum Einlesen von prozessinstanzunabhängigen Kennzahlen- und Dimensionswerten.

Prozessinstanzunabhängige Daten berücksichtigen Aspekte, die nicht prozessorientiert sind, z. B. betriebswirtschaftliche Fixkosten, Kundenzufriedenheit oder finanzwirtschaftliche Kennzahlen, die die finanzielle Sicht auf ein Unternehmen charakterisieren.

PROZESSINSTANZUNABHÄNGIGE KENNZAHLEN

Die Werte prozessinstanzunabhängiger Kennzahlen (siehe Kapitel **Prozessinstanzunabhängige Kennzahlen** (Seite 66)) sind Kennzahlenwerte, die nicht aus Prozessinstanzdaten berechnet werden. Sie werden direkt mit den referenzierten Dimensionswerten ohne Bezug zu Prozessinstanzdaten importiert.

Prozessinstanzunabhängige Kennzahlen können als Grundlage für benutzerdefinierte Kennzahlen verwendet und so mit prozessinstanzabhängigen Kennzahlen kombiniert werden.

PROZESSINSTANZUNABHÄNGIGE DIMENSIONSDATEN

Für ein-, zwei- und n-stufige Textdimensionen können Werte von Schlüsseln und Beschreibungen vorab, d. h. vor dem eigentlichen Datenimport, in ein PPM-System importiert werden (siehe Kapitel **Dimensionswerte** (Seite 79)).

5.1 Prozessinstanzunabhängige Kennzahlen

In den folgenden Kapiteln wird die Konfiguration der Importdatenformate (XML, CSV, XLS) prozessinstanzunabhängiger Kennzahlen sowie der Im- und Export von prozessinstanzunabhängigen Kennzahlenwerten beschrieben.

Voraussetzung dafür ist, dass die entsprechenden prozessinstanzunabhängigen Kennzahlen im PPM-System existieren. Wie Sie prozessinstanzunabhängige Kennzahlen definieren und anmelden, erfahren Sie in der Technischen Referenz **PPM Customizing**.

5.1.1 Datenimportformate

Importdaten prozessinstanzunabhängiger Kennzahlen können in den folgenden Formaten importiert werden:

- XML (Standard)
- CSV
- XLS

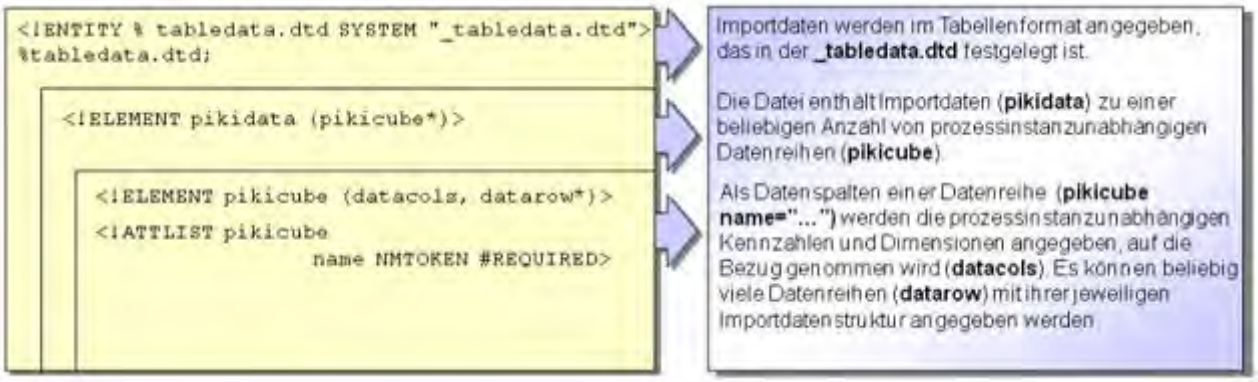
Unabhängig vom Format der Importdaten müssen immer alle Schlüsseldimensionen (Attribut **iskeydimension="TRUE"** des Elements **refdim** in der Definition der Datenreihe, siehe Technische Referenz **PPM Customizing**) und mindestens eine der prozessinstanzunabhängigen Kennzahlen der Datenreihe angegeben werden. Die Angabe weiterer,

prozessinstanzunabhängiger Kennzahlen bzw. der Nichtschlüsseldimensionen des betreffenden PIKI-Cube in der Importdatenstruktur ist optional.

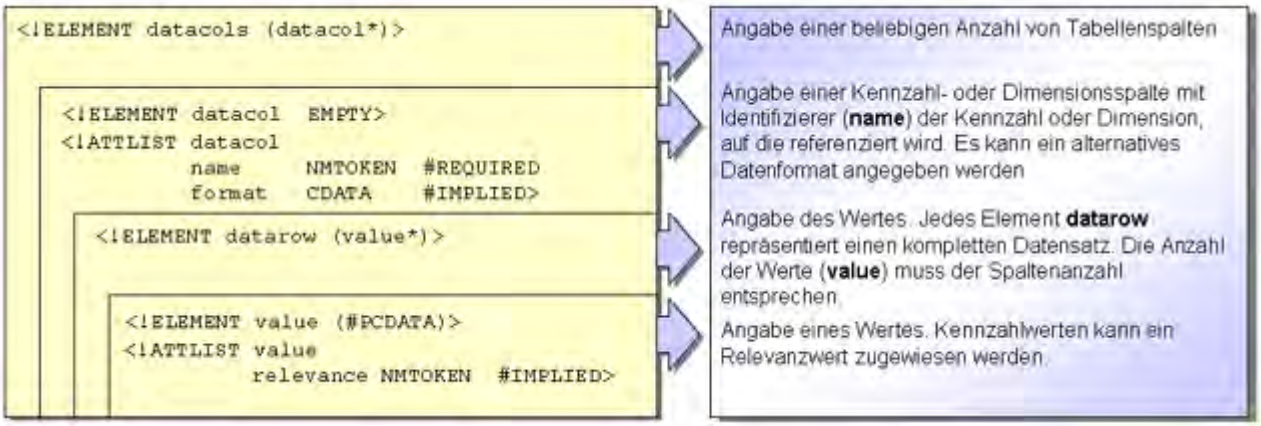
5.1.1.1 XML-Format

Das XML-Datenimportformat für prozessinstanzunabhängige Kennzahlen ist durch folgende Dokumenttypdefinitionen vorgegeben:

DTD **pikidata.dtd** (Referenzierung von Datenreihen als Importdaten)



DTD **_tabledata.dtd** (Tabellenformat für den Import prozessinstanzunabhängiger Daten):



Importdaten prozessinstanzunabhängiger Kennzahlen im XML-Format haben somit folgende, allgemeine Struktur:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE pikidata SYSTEM "pikidata.dtd">
<pikidata>
  <pikicube name="...">
    <datacols>
      <datacol name="..."/>
      ...
    </datacols>
    <datarow>
      <value relevance="...">...</value>
       ...
```

```
    </datarow>
    ...
  </pikicube>
  ...
</pikidata>
```

ELEMENT und ATTLIST pikidata	Beschreibung
pikidata	Liste von Importdaten zu beliebig vielen, prozessinstanzunabhängigen Datenreihen (PIKI-Cube)
pikicube	Datenreihe, in welche die angegebenen Daten importiert werden sollen
name	Identifizierer der Datenreihe. Muss mit dem in der Kennzahlenkonfiguration angegebenen Namen des PIKI-Cube (XML-Attribut **pikicube name**) übereinstimmen.

ELEMENT und ATTLIST datacols	Beschreibung
datacols	Importdatenstruktur (Tabellenspalten der Datenreihe)
datacol	Angabe einer Tabellenspalte. Für jede prozessinstanzunabhängige Kennzahl und jede gewünschte, referenzierte Dimension wird eine Spalte angegeben.
name	Identifizierer der Tabellenspalte. Muss mit den in der Kennzahlenkonfiguration angegebenen Namen der jeweiligen prozessinstanzunabhängigen Kennzahl (**pikidef name**) sowie den Namen der referenzierten Dimensionen (**refdim name**) übereinstimmen.
format	Optionales Importformat für Spaltenwerte. Unterstützte Formate sind Fließkommazahl, Zeit- und Uhrzeitwerte.
datarow	Konkrete Importdaten für die Datenreihe in der Form einer Datenzeile. Die Reihenfolge der Importwerte (**value**-Elemente) muss mit der Spaltenreihenfolge (**datacols**) übereinstimmen, so dass jedem Kriterium der Datenreihe (prozessinstanzunabhängige Kennzahl bzw. referenzierte Dimension) eindeutig ein Wert zugewiesen werden kann.

ELEMENT und ATTLIST datacols	Beschreibung
value	Zu importierender Spaltenwert einer Datenzeile (Wert einer referenzierten Dimension bzw. einer prozessinstanzunabhängigen Kennzahl)
relevance (optional)	Relevanzwert, der sich auf den betreffenden Wert einer prozessinstanzunabhängigen Kennzahl bezieht

Beispiel 1

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE pikidata SYSTEM "pikidata.dtd">
<pikidata>
  <pikicube name="PIKICUBE_COSTS">
    <datacols>
      <datacol name="OVERHEAD_COSTS"/>
      <datacol name="PROCESSTYPE"/>
      <datacol name="TIME" format="MMM yyyy"/>
      <datacol name="MATERIAL"/>
    </datacols>
    <datarow>
      <value relevance="100">1000 EUR</value>
      <value>Auftragsabwicklung\Automobilindustrie</value>
      <value>Juni 2000</value>
      <value>NaviSet B6{Navigationssystem}</value>
    </datarow>
    <datarow>
    ...
    </datarow>
    ...
  </pikicube>
  ...
</pikidata>
```

FESTLEGUNG DER IMPORTDATENSTRUKTUR (XML-ELEMENTE **DATACOLS** UND **DATACOL**)

Im XML-Element **datacols** werden die Identifizierer der prozessinstanzunabhängigen Kennzahlen und der referenzierten Dimensionen (**datacol name="..."**) angegeben, auf die sich der Datenimport für die jeweilige Datenreihe bezieht. Die Importdatenstruktur wird in einem einfachen Tabellenformat angegeben, in welchem jedes Kriterium der Datenreihe (prozessinstanzunabhängige Kennzahl bzw. referenzierte Dimension) eine Tabellenspalte darstellt.

DATENIMPORT

Die eigentlichen Importwerte werden aus den XML-Elementen **datarow** ausgelesen. Jedes dieser Elemente stellt eine Datenzeile dar, die importiert wird. Jedes angegebene **value**-Element korrespondiert dabei mit dem entsprechenden **datacol**-Element der festgelegten Importdatenstruktur.

Die Reihenfolge der Wertangaben (XML-Element **value**) einer Datenzeile (XML-Element **datarow**) muss mit der Reihenfolge der Spalten (im XML-Element **datacols**) in der Importdatenstruktur übereinstimmen.

Im XML-Attribut **format="MMM yyyy"** wird für die referenzierte Dimension **TIME** das Datenformat für den einzulesenden Wert festgelegt. Formatangaben sind optional.

Durch Angabe eines Beschreibungstextes in geschweiften Klammern **{ }** werden die Werte für die einstufige Dimension **MATERIAL** in der Form **<Identifizierer>{<Beschreibungstext>}** erwartet. Die Angabe eines Beschreibungstextes ist optional, da für den Dimensionsbezug nur der Identifizierer entscheidend ist.

Im Beispieldatensatz ist für die prozessinstanzunabhängige Kennzahl **OVERHEAD_COSTS** ein Relevanzwert angegeben. Wenn Sie keinen Relevanzwert angeben, wird als Vorgabe der Wert **relevance="1"** verwendet.

Wird für einen Datenwert keine Einheit angegeben, wird standardmäßig die Basiseinheit des Attributdatentyps angenommen, auf dem die prozessinstanzunabhängige Kennzahl oder referenzierte Dimension basiert.

Beispiel 2

Es gibt zwei verschiedene Formate für den Import von Werten zweistufiger Dimensionen:

- Wertdefinition in einer Zeile:

Die Wertangaben werden durch einen Rückwärtsschrägstrich (****) getrennt.

```
...
  <datacols>
    ...
    <datacol name="PRINCIPAL"/>
    ...
  </datacols>
  ...
  <datarow>
    ...
    <value>DE{Deutschland}\0000000001{Becker}</value>
    ...
  </datarow>
  ...
```

- Wertdefinition in zwei Zeilen:

Die Werte werden in zwei unmittelbar aufeinanderfolgenden XML-Elementen angegeben.

```
...
  <datacols>
    ...
    <datacol name="PRINCIPAL"/>
    <datacol name="PRINCIPAL"/>
    ...
  </datacols>
  <datarow>
    ...
    <value>DE{Deutschland}</value>
    <value>0000000001{Becker}</value>
    ...
  </datarow>
  ...
```

5.1.1.2 CSV-Format

Für den Import im CSV-Format gelten folgende Besonderheiten:

- Eine CSV-Datei kann nur Daten für eine Datenreihe enthalten.
- Die in einer **CSV-Datei** angegebenen Werte dürfen nicht das angegebene Datentrennzeichen enthalten. Das Datentrennzeichen wird über die Option **-csvchar "<Zeichen>"** des Kommandozeilenprogramms **runpikidata** angegeben.

Beispiel (mit Zeilennummern)

```
1 PIKICUBE_COSTS
2 OVERHEAD_COSTS(.,);OVERHEAD_COSTSNUM;TIME(MM.yyyy);↵
PROCESSTYPE;PROCESSTYPE
3 1.000,00 EUR;100;05.2000;Auftragsabwicklung;↵
Automobilindustrie
4 1.020,00 EUR;;06.2000;Auftragsabwicklung;↵
Automobilindustrie
...
```

Erläuterung

- Zeile 1: Name der Datenreihe (bspw. **PIKICUBE_COSTS**)
- Zeile 2: Angabe der Importdatenstruktur
  Die einzelnen Tabellenspalten der Datenreihe sind durch Semikolon voneinander getrennt. Formatangaben können nach dem Spaltennamen in runden Klammern angegeben werden. Die optionale Angabe der Relevanz muss unmittelbar auf die betreffende Kennzahl folgen. Der Relevanz-Spaltenname ergibt sich aus dem Namen der Kennzahl erweitert um die Zeichenfolge **NUM** (bspw. **OVERHEAD_COSTSNUM**).
- ab Zeile 3: Importdaten

Jede Zeile stellt eine zu importierende Datenzeile der Datenreihe dar. Die einzelnen Werte werden durch das angegebene Trennzeichen getrennt.
Für die Importwerte der prozessinstanzunabhängigen Kennzahl **OVERHEAD_COSTS** ist in der ersten Datenzeile (Zeile 3) ein Wert für die Relevanz (**OVERHEAD_COSTSNUM**) von 100 (**1.000,00 EUR;100**) angegeben, in der zweiten Datenzeile (Zeile 4) dagegen kein Relevanzwert (**1.020,00 EUR;;**). In diesem Fall wird automatisch der Relevanzwert **1** gesetzt.
Die Werte für zweistufige Dimensionen (im Beispiel: **PROCESSTYPE**) werden über zwei separate, aufeinander folgende, gleichnamige Spalten importiert. Die erste Spalte enthält den groben Wert, die zweite den feinen Wert.

Die optionale Formatangabe **MM.yyyy** für die referenzierte Dimension **Zeit** setzt sich aus einer ganzzahligen Monatsangabe gefolgt von einer vierstelligen Jahresangabe zusammen. Die einzelnen Feldwerte werden durch einen Punkt voneinander getrennt (siehe **Datenformate** (Seite 43)).

5.1.1.3 XLS-Format

Sie können Daten prozessinstanzunabhängiger Datenreihen aus Excel-Dateien ins PPM-System importieren. Für den Import im XLS-Format gelten folgende Besonderheiten:

- Eine Excel-Datei kann pro Arbeitsblatt nur Daten für eine Datenreihe enthalten. Alle Arbeitsblätter der Excel-Datei werden importiert.
- Beim Importieren eines Arbeitsblattes wird immer der gesamte Bereich des Blattes importiert. Spalten und Zeilen ohne Inhalt werden übersprungen.
- Die Namen der zu importierenden Arbeitsblätter können mit dem optionalen Kommandozeilenparameter **-sheet** angegeben werden. Enthält der Name Leerzeichen, müssen Sie den Namen in Anführungszeichen angeben. Wenn Sie mehrere Arbeitsblätter aus einer Datei importieren möchten, geben Sie die Namen durch Leerzeichen getrennt an.
- Das Zellenformat von Dimensionsspalten muss vom Datentyp **Text** sein. Das Format der Zellen von Kennzahlenspalten muss vom Typ **Text** oder **Zahl** sein. Formatvorgaben (z. B. Dezimal- und Tausender-Trennzeichen) können dabei nur für Kennzahlenwerte im Text-Format gemacht werden.

IMPLIZITE POSITIONSANGABE

Prinzipiell entspricht das Datenformat dem des CSV-Formats. Die Angaben im Arbeitsblatt werden in der folgenden Reihenfolge erwartet:

- Zelle A1: Name der Datenreihe
- Zelle A2 und folgende Zellen (nur Zeile 2): Definition der Datenstruktur
- Zelle A3 und folgende Zellen (alle Zeilen): Datenwerte

Beispiel: Dateiauszug aus MS Excel mit Standardpositionierung

	A	B	C
1	PC_CUSTSAT		
2	CUST_SATISFACT(,)	TIME(MM.yyyy)	PRINCIPAL
3	7.60	01.2004	DE{DEUTSCHLAND}
4	6.60	02.2004	US{USA}
5			
6			

2004 / 2005 / 2006 /

EXPLIZITE POSITIONSANGABE

Der Name der Datenreihe ist in Zelle A1 angegeben. In Zelle B1 ist der Bereich festgelegt, der die Angaben der Importdatenstruktur enthält und in Zelle C1 der Bereich, der die eigentlichen Importdaten enthält. Die Positionsangabe hat das allgemeine Format

<Position der Startzelle>[:< Position der Endzelle >]

Wenn Sie nur die Position der Startzelle angeben, z. B. C8, werden auch die Daten aller folgenden Spalten importiert. Die Angabe entspricht in diesem Fall einem unendlichen Tabellenbereich. Wenn Sie bei der Positionsangabe für den auszulesenden Bereich der Importdaten dieselbe

Zeilennummer angeben, z. B. C9:E9, werden die angegebenen Spalten bis zum Ende des Arbeitsblattes ausgelesen und damit eine unbegrenzte Anzahl an Datenzeilen importiert. Möchten Sie einen exakt begrenzten Wertebereich importieren, müssen Sie die erste und die letzte Datenzeile angeben, z. B. C9:E10.

Beispiel: Dateiauszug aus MS Excel mit expliziter Positionsangabe

	A	B	C	D	E
1	PC_CUSTSAT	C8:E8	C9:E10		
2					
3					
4					
5					
6					
7					
8			CUST_SATISFACT(,)	TIME(MM.yyyy)	PRINCIPAL
9			7.60	01.2005	DE{DEUTSCHLAND}
10			6.60	02.2005	US{USA}
11					

Die Importdatenstruktur der prozessinstanzunabhängigen Datenreihe **PC_CUSTSAT** ist in den Zellen C8 bis E8, die zu importierenden Werte selbst sind im Zellenbereich C9 bis E10 angegeben.

Sie können den Excel-Tabellenbereich, der außerhalb der Positionsangaben liegt, für Kommentare u. a. nutzen. Dieser Bereich wird beim Import ignoriert.

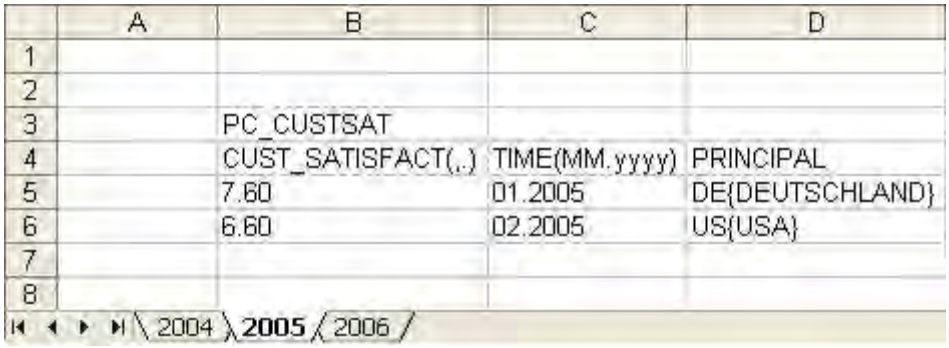
Warnung

Wenn Sie die explizite Positionsangabe verwenden möchten, müssen Sie die prinzipielle Reihenfolge beachten: Vor den Zeilen mit Angabe von Datenwerten müssen Sie die Zeile der Importdatenstruktur angeben.

RELATIVE STARTANGABE

Der Name der Datenreihe ist an von impliziter Positionsangabe (Zelle A1) abweichender Position angegeben. Die Position der Zelle im Arbeitsblatt wird mit dem Parameter **sheet** zusammen mit dem Namen des Arbeitsblattes angegeben. Die Syntax lautet
-sheet <Arbeitsblattname>:<Zelle>.

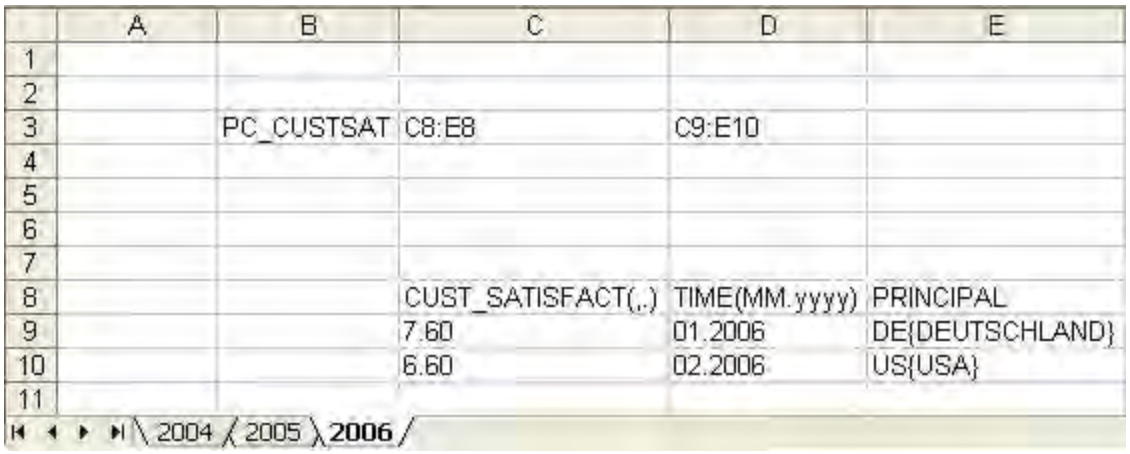
Beispiel 1: Dateiauszug aus MS Excel mit relativer Startangabe und impliziter Positionsangabe



	A	B	C	D
1				
2				
3		PC_CUSTSAT		
4		CUST_SATISFACT(,.)	TIME(MM.yyyy)	PRINCIPAL
5		7.60	01.2005	DE{DEUTSCHLAND}
6		6.60	02.2005	US{USA}
7				
8				

2004 **2005** 2006

Beispiel 2: Dateiauszug aus MS Excel mit relativer Startangabe und expliziter Positionsangabe



	A	B	C	D	E
1					
2					
3		PC_CUSTSAT	C8:E8	C9:E10	
4					
5					
6					
7					
8			CUST_SATISFACT(,.)	TIME(MM.yyyy)	PRINCIPAL
9			7.60	01.2006	DE{DEUTSCHLAND}
10			6.60	02.2006	US{USA}
11					

2004 2005 **2006**

Für beide Beispiele werden die Daten durch folgenden Befehl importiert:

```
runpikidata -user <username> -password <password> -mode import -format XLS -file excelpikidata.xls -sheet 2006:B3
```

5.1.2 Erneuter Datenimport

Der Import von Daten in prozessinstanzunabhängige Datenreihen arbeitet generell bzgl. neuer Datenzeilen **additiv** und bzgl. geänderter, bestehender Datenzeilen **überschreibend**.
Bei einem erneuten Datenimport werden bestehende Datenzeilen mit geänderten, aktuellen Werten der prozessinstanzunabhängigen Kennzahlen und referenzierten Dimensionen der Datenreihe überschrieben und bisher nicht vorhandene Datenzeilen in die jeweilige Datenreihe neu eingefügt. Das PPM-System erkennt bestehende Datenzeilen anhand der Werte der Schlüsseldimensionen einer Datenreihe.

Schlüsseldimensionen werden durch das Attribut **iskeydimension="TRUE"** des Elements **refdim** in der Definition der Datenreihe bestimmt. Weitere Informationen finden Sie im Dokument **PPM Customizing**.

Beispiel

In eine bestehende Datenreihe wurde bereits die folgende Datenzeile importiert:

D_LAND*	D_WERK*	D_ABTEILUNG*	D_ERFASSER	UMSATZ	KOSTEN
Deutschland	Hamburg	42	Schmidt	400000	

Die mit * gekennzeichneten, referenzierten Dimensionen seien die Schlüsseldimensionen der Datenreihe. Die Wertkombination **Deutschland; Hamburg; 42** ist daher der Identifizierer der gezeigten Datenzeile. Für die Datenreihe sind die beiden prozessinstanzunabhängigen Kennzahlen **UMSATZ** bzw. **KOSTEN** konfiguriert.

Erfolgt ein erneuter Datenimport bspw. mit den Werten **Deutschland; Hamburg; 42; Huber;;280000**, erkennt das PPM-System, dass diese Datenzeile bereits existiert und überschreibt zum einen den Wert der Dimension **D_ERFASSER** (**Schmidt**) mit dem geänderten Wert (**Huber**) und fügt zum anderen für die prozessinstanzunabhängige Kennzahl **KOSTEN** einen Wert ein:

D_LAND*	D_WERK*	D_ABTEILUNG*	D_ERFASSER	UMSATZ	KOSTEN
Deutschland	Hamburg	42	Huber	400000	280000

Der Wert der prozessinstanzunabhängigen Kennzahl **UMSATZ** wird unverändert beibehalten, da beim erneuten Datenimport kein neuer Wert importiert wurde.

5.1.3 Export von Werten prozessinstanzunabhängiger Datenreihen

Die Werte prozessinstanzunabhängiger Kennzahlen können mit Hilfe des Kommandozeilenprogramms **runpikidata** im XML-Format exportiert werden. Verwenden Sie hierfür das Kommandozeilenargument **-mode export** (siehe Kapitel **Kommandozeilenprogramm runpikidata** (Seite 76)).

Die Daten zu allen Datenreihen des angegebenen Mandanten werden in die mittels Argument **-file** angegebene Datei geschrieben. Wenn Sie die Werte bestimmter Datenreihen exportieren möchten, geben Sie die Namen der gewünschten Datenreihen durch Komma getrennt mit dem Argument **-pikicube** an.

Der ausführende PPM-Benutzer benötigt das Funktionsrecht **Datenimport**.

Beispiel

```
runpikidata -user system -password manager -client umg_de -mode export -file pikidata
-pikicube PC_MINITAB_CPK,PC_SALES_REVENUES
```

Die Werte der Datenreihen **PC_MINITAB_CPK** und **PC_SALES_REVENUES** des Mandanten **umg_de** werden im XML-Format in die Datei **pikidata.xml** exportiert.

5.1.4 Löschen von Werten prozessinstanzunabhängiger Datenreihen

Werte prozessinstanzunabhängiger Datenreihen im PPM-System können mit Hilfe des Kommandozeilenprogramms **runpikidata** gelöscht werden. Verwenden Sie hierfür die Kommandozeilenargumente **-mode delete** und **-pikicube <cubename>[,<cubename>,...]**, um die Datenreihen anzugeben, aus denen Sie Werte löschen möchten.

Standardmäßig werden alle Werte der angegebenen PIKI-Cubes gelöscht.
Sie können die zu löschenden Datenzeilen einer Datenreihe durch Angabe eines Paramsets, das bspw. einen Filter bzgl. einer referenzierten Dimension der Datenreihe enthält, einschränken. Geben Sie hierfür mittels Parameter **-ps** den Namen der Paramset-Datei an.

Enthält das Lösch-Paramset Filter bzgl. Dimensionen, die in der Definition der angegebenen Datenreihe nicht als referenzierte Dimensionen enthalten sind, wird bei Programmausführung eine Fehlermeldung ausgegeben. Der ausführende PPM-Benutzer benötigt das Funktionsrecht **Datenimport**.

Beispiel

Sie möchten Importwerte der Datenreihe mit dem internen Bezeichner **PC_TURNOVER** (PIKI-Cube Umsatz) des Zeitraums 1. Quartal 2009 löschen.
In der Analyse stellen Sie ein Lösch-Paramset mit dem entsprechenden Zeitfilter ein und exportieren dieses lokal in eine XML-Datei. Anschließend rufen Sie das Kommandozeilenprogramm **runpikidata** wie folgt auf:

```
runpikidata -client <ppmclient> -user <username> -password <password> -pikicube
PC_TURNOVER -mode delete -ps <deleteparamset>.xml
```

5.1.5 Kommandozeilenprogramm runpikidata

Der Datenimport bzw. -export prozessinstanzunabhängiger Datenreihen erfolgt mittels Kommandozeilenprogramm **runpikidata**:

```
runpikidata -user <username> -password <password>
   [-client <name>]
   -mode (import|export|delete)
   [-compatible4]
   -file <file1>[,<file2>...]
   [-format {XML|CSV|XLS} [-csvchar "<character>"
   [-encoding "<encodingname>"]]]
   -pikicube <cubename>[,<cubename>...]
   -ps <filename>
   [-sheet <name>[:<zelle>] [<name>[:<zelle>]]]
   [-version]
   [-language <ISO-code>] [protocoloptions]
   [-recoveryfile {yes|no}]

   -user <username>         Name des Benutzers
   -password <password>     Kennwort des Benutzers
   -client <name>          Name des Mandanten. Wird
                        kein Mandant angegeben, wird
```

```
                        der Server des
                        Standardmandanten verwendet.
    -language <ISO-code>     Importsprache
    -mode (import|export|delete) Importieren, Exportieren
                        oder Löschen von Daten
    -compatible4           Import von Dimensionswerten wie
                        bei PPM 4.x ohne Prüfung der
                        Verfeinerungsstufe
    -file <file1>[,<file2>...] Importdateien (-mode import)
                        bzw. Exportdateien
                        (-mode export).
                        ZIP-Dateien werden unterstützt.
    -format {XML|CSV|XLS}    Datenformat für den Import
                        (Default: XML)
    -csvchar "<character>"   Datentrennzeichen für das
                        Importformat "CSV"
                        (Default: Komma)
    -encoding "<encodingname>" Encoding der CSV-Datei
                        (Default:
                        Default-Encoding der
                        Workstation)
    -pikicube <cubename>     Datenreihe, aus der Daten
     [,<cubename>...]       gelöscht oder exportiert werden
                        sollen (-mode delete|export)
    -ps <filename>          Paramset zur Angabe der zu
                        löschenden Daten (-mode delete)
    -sheet <name>[:<zelle>]>  Vorgabe von Excel-Sheets für
                        das Importformat "XLS". Die
                        Zelle gibt die Position vor, ab
                        der die Daten importiert werden
                        sollen.
    -recoveryfile {yes|no}   Erstellt nach erfolgreichem
                        Abschluß des Imports die neuen,
                        für die Prozessanalyse relevanten
                        Wiederherstellungsdateien
                        Default: yes)

    -version              Versionsnummer der Applikation
                        und des Datenbankschemas

protocoloptions kann aus folgenden Anweisungen bestehen:
    -protocolfile <filename>  Protokollieren in Datei
                        <filename>
    -information {yes|no}    Protokollieren von
                        Informationen
    -warning {yes|no}       Protokollieren von Warnungen
    -error {yes|no}         Protokollieren von Fehlern
```

-VERSION

Auf der Konsole werden die Version der PPM-Software und des Datenbankschemas ausgegeben. Andere Argumente werden ignoriert.

-USER <BENUTZERNAME> -PASSWORD <KENNWORT>

Mit diesem Parameter geben Sie den Benutzernamen und das Kennwort des PPM-Benutzers an, der den Import ausführt. Der Benutzer muss das Funktionsrecht **Datenimport** haben.

-CLIENT <MANDANTENNAME>

Mit diesem Parameter geben Sie den PPM-Mandanten an, in dessen Datenbankschema die importierten Prozessfragmente gespeichert werden. Wenn Sie diese Option nicht verwenden, wird der Standardmandant verwendet.

-MODE IMPORT|EXPORT|DELETE)

Mit diesem Parameter geben Sie bspw. an, dass Sie eine Quelldatei (bzw. mehrere Quelldateien) mit Daten des gewählten Importformats (**-format {XML|CSV|XLS}**) importieren (**-mode import -file <file1>[,<file2>...]**) oder im PPM-System vorhandene Daten einer Datenreihe in eine XML-Datei exportieren (**-mode export -pikicube <cubename> -file <filename>**) oder Daten aus den angegebenen Datenreihen komplett (**-mode delete -pikicube <cubename>[,<cubename>...]**) oder teilweise löschen (**-mode delete -pikicube <cubename> -ps <filename>**) möchten.

-COMPATIBLE4

Mit dem optionalen Parameter geben Sie an, dass beim Import von Werten referenzierter, mehrstufiger Textdimensionen (**refdim**) wie in PPM der Versionen 4.x keine Prüfung der Verfeinerungsstufe (**refinement**) der Importdaten stattfinden soll. Dies ist erforderlich, wenn Dimensionswerte von prozessinstanzunabhängigen Kennzahlen-Importdaten für PPM ab Version 5 in einer anderen Verfeinerungsstufe als der in der Definition der Datenreihe Angegebenen vorliegen (siehe Technische Referenz **PPM Customizing**).

-FILE <FILE1>[,<FILE2>,...]

Mit diesem Parameter geben Sie den Pfad zur Importdatei (bzw. zu den Importdateien) und die Importdatei[en] selbst an. Die Quelldatei kann auch eine ZIP-Datei sein, die eine oder mehrere Importdateien vom gleichen Datenformat enthält.

-FORMAT {XML|CSV|XLS}

Mit diesem Parameter geben Sie das verwendete Datenimportformat an.

-CSVCHAR <ZEICHEN>

Mit diesem Parameter geben Sie das Trennzeichen für Datenfeldwerte des CSV-Importformats an (Vorgabewert ist das Komma).

-ENCODING "<ENCODINGNAME>"

Mit diesem Parameter geben Sie den Namen des CSV-Encodings an, das Sie optional verwenden möchten. Standardmäßig wird unter Windows das Encoding CP-1250 für CSV-Dateien verwendet.

-SHEET <NAME>[:<ZELLE>]>

Mit diesem Parameter geben Sie den Namen des Excel-Arbeitsblattes an, dessen Daten Sie importieren möchten. Sie können die Namen mehrerer Arbeitsblätter durch Leerzeichen getrennt angeben. Mit dem Argument **<zelle>** können Sie für jedes Arbeitsblatt die relative Startposition (Zelle, die den Namen der Datenreihe enthält) angeben, ab der Daten importiert werden sollen.

-RECOVERYFILE {YES|NO}

Mit diesem Parameter geben Sie an, ob nach dem Import oder Löschen von Werten prozessinstanzunabhängiger Datenreihen die für die Prozessanalyse relevanten Wiederherstellungsdateien des Analyseservers erstellt werden sollen. Vorgabewert ist **yes**.

5.2 Dimensionswerte

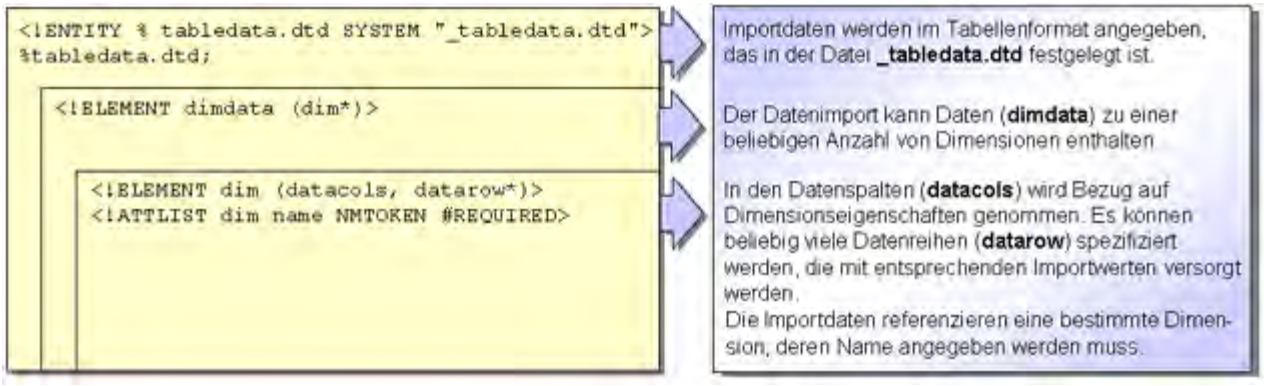
Für ein-, zwei- und n-stufige Textdimensionen können Dimensionswerte, vor allen Dingen umfangreiche Stufenbeschreibungen, vor dem eigentlichen PPM-Import in das PPM-System eingelesen werden. Die Dimensionswerte setzen sich aus einer obligatorischen ID und einer optionalen Beschreibung zusammen.

Vorteile:

- Sie können Planwertdefinitionen in PPM erstellen, bevor Prozessinstanzen importiert werden.
- Das Datenvolumen der zu importierenden Prozessinstanzen während des eigentlichen PPM-Imports kann deutlich reduziert werden, wenn Sie auf die Angabe der bereits vorab importierten Stufenbeschreibungen verzichten.

5.2.1 XML-Format

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



Warnung

Die Namen der Datenspalten von Schlüssel und Beschreibung einer Dimensionsstufe müssen immer mit dem Präfix **LEVEL** beginnen und mit dem Suffix ID (Schlüssel) bzw. DESC (Beschreibung) enden. Verwenden Sie zusätzlich eine fortlaufende Nummerierung beginnend mit 1 für die Angabe der Stufe. Andernfalls wird der Import mit einer Fehlermeldung abgebrochen.

Die folgende Tabelle veranschaulicht die Zuordnung der Datenspalten zu den Dimensionswerten (**<n>** ist die fortlaufende Nummerierung der Stufenzahl):

dimdata-Konfiguration	Dimensionskonfiguration	Beispiel
LEVEL<n>_ID	ID der n-ten Dimensionsstufe	**LEVEL5_ID**
LEVEL<n>_DESC	Beschreibung der n-ten Dimensionsstufe	**LEVEL5_DESC**

Beispiel

Für die zweistufige Dimension **Auftraggeber** (**PRINCIPAL**) werden drei Werte importiert:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE dimdata SYSTEM "dimdata.dtd">
<dimdata>
  <dim name="PRINCIPAL">
    <datacols>
      <datacol name="LEVEL1_ID"/>
      <datacol name="LEVEL1_DESC"/>
      <datacol name="LEVEL2_ID"/>
      <datacol name="LEVEL2_DESC"/>
    </datacols>
    <datarow>
      <value>DE</value>
      <value>Deutschland</value>
      <value>0000000003</value>
      <value>Becker</value>
    </datarow>
    <datarow>
      <value>FR</value>
      <value>Frankreich</value>
      <value>0000000092</value>
      <value>Leclerc</value>
    </datarow>
    <datarow>
      <value>EN</value>
      <value>Großbritannien</value>
      <value>0000000027</value>
      <value>Crichton</value>
    </datarow>
  </dim>
</dimdata>
```

Warnung

Beim Importieren von Werten für mehrstufige Textdimensionen müssen Sie den Schlüssel der ersten Dimensionsstufe (**LEVEL1_ID**) immer angeben.

5.2.2 CSV-Format

Für den Import im CSV-Format gelten folgende Besonderheiten:

- Eine CSV-Datei kann nur Daten für eine Dimension enthalten.
- Die in einer **CSV-Datei** angegebenen Dimensionswerte dürfen nicht das angegebene Datentrennzeichen enthalten. Das Datentrennzeichen wird über die Option **-csvchar "<Zeichen>"** des Kommandozeilenprogramms **rundimdata** angegeben.

Beispiel (mit Zeilenummern):

```
1 PRINCIPAL
2 LEVEL1_ID;LEVEL1_DESC;LEVEL2_ID;LEVEL2_DESC
3 DE;Deutschland;0000000003;Becker
4 FR;Frankreich;0000000092;Leclerc
5 UK;Großbritannien;0000000027;Crichton
...
```

Erläuterung

- Zeile 1: Identifizierer der Dimension
- Zeile 2: Definition der Datenstruktur
  Die einzelnen Spalten sind durch Semikolon getrennt. Die Spaltennamen sind Schlüsselwörter, die nicht verändert werden dürfen.
- ab Zeile 3: Datenwerte

Jede Zeile enthält einen Datensatz. Die einzelnen Werte werden durch das angegebene Trennzeichen getrennt. Die Wertangaben müssen jeweils den in der Attributtypkonfiguration festgelegten Datentypen entsprechen. Die Spalteneinträge stellen sich von links nach rechts wie folgt dar: Landeskürzel, Name des Landes, Auftraggeber-ID, Auftraggebername

5.2.3 Standard- und Ersatzwerte

Wenn Sie in der Kennzahlenkonfiguration für den Schlüssel bzw. die Beschreibung einer Dimensionsstufe Ersatz- bzw. Standardwerte angegeben haben und diese beim Import mit **rundimdata** verwenden möchten, lassen Sie die entsprechenden **value**-Elemente in der XML-Importdatei bzw. Spaltenwerte in der CSV-Datei leer.
Näheres zur Verwendung von Standard- und Ersatzwerten in Textdimensionen finden Sie in der Technischen Referenz **PPM Customizing**.

5.2.4 Erneuter Datenimport

Der Import prozessinstanzunabhängiger Dimensionswerte (**rundimdata -mode import**) arbeitet prinzipiell additiv, d. h. noch nicht vorhandene Dimensionswerte werden im PPM-System neu hinzugefügt (mit Ausnahme der zweistufigen Textdimension **PROCESSTYPE**, hier ist ein additiver Import nicht möglich).

Geben Sie zusätzlich die Kommandozeilenoption **-overwrite** an, werden bestehende Beschreibungen von Dimensionsstufen mit den importierten, geänderten Beschreibungen überschrieben.

Wenn Sie die Beschreibungen bestimmter Dimensionsstufen löschen möchten, importieren Sie leere Beschreibungen für diese Dimensionsstufen im Überschreibungsmodus mit **-mode import -overwrite**.

5.2.5 Löschen von Dimensionswerten

Zum Löschen von Dimensionswerten verwenden Sie die Option **-replace** des Kommandozeilenprogramms **rundimdata**. Hierdurch werden alle Dimensionswerte, die nicht in der Importdatei enthalten sind, zum Löschen markiert. Um die Daten endgültig zu löschen, muss der Analyseserver vollständig neu initialisiert werden, standardmäßig werden daher nach dem Import die für die Prozessanalyse relevanten Wiederherstellungsdateien des Analyseservers neu erstellt. Am Ende des Imports gibt das Kommandozeilenprogramm eine entsprechende Meldung aus.

Wenn Sie alle Dimensionsdaten einer Dimension löschen möchten, importieren Sie eine semantisch leere Datei.

Die Option **-replace** zum Ersetzen bzw. Löschen von Dimensionswerten impliziert die Option **-overwrite**. Deshalb darf die Option **-overwrite** bei Verwendung der Option **-replace** nicht an der Kommandozeile angegeben werden.

5.2.6 Kommandozeilenprogramm rundimdata

Der Import prozessinstanzunabhängiger Dimensionswerte im XML- und CSV-Format erfolgt über das Kommandozeilenprogramm **rundimdata**:

```
rundimdata -user <username> -password <password> [-client <name>]
    -mode import
    -file <file1>[,<file2>...]
    [-format {XML|CSV} [-csvchar "<character>"]]
    [-overwrite |-replace]
    [-recoveryfile {yes|no}]
    [-version]
    [-language <ISO-code>][protocoloptions]

    -user <username>         Name des Benutzers
    -password <password>     Kennwort des Benutzers
    -client <name>           Name des Mandanten. Wird kein Mandant angegeben,
                         wird der Server des Standardmandanten verwendet.
    -language <ISO-code>     Importsprache
    -mode import             Importieren der Daten
    -recoveryfile {yes|no}    Nach Abschluss des Imports in den Analyseserver
                         werden die für die Prozessanalyse relevanten
                         Wiederherstellungsdateien neu erstellt.
    -file <file1>[,<file2>...] Importdateien. ZIP-Dateien werden unterstützt.
    -format {XML|CSV}        Datenformat für den Import (Default: XML)
    -csvchar "<character>"    Datentrennzeichen für das Importformat "CSV"
                         (Default: Komma)
    -overwrite              Überschreiben von Dimensionsbeschreibungen
    -replace               Löschen von bestehenden Dimensionswerten, die
                         durch einen Datenimport für Dimensionen geändert
                         oder ergänzt wurden.
    -encoding "<encodingname>" Encoding der CSV-Datei (Default:
                         Default-Encoding der Workstation)
```

```
    -version                Versionsnummer der Applikation und des
                           Datenbankschemas

protocoloptions kann aus folgenden Anweisungen bestehen:
    -protocolfile <filename>     Protokollieren in Datei <filename>
    -information {yes|no|default} Protokollieren von Informationen
    -warning {yes|no|default}    Protokollieren von Warnungen
    -error {yes|no|default}      Protokollieren von Fehlern
```

-VERSION

Auf der Konsole werden die Version der PPM-Software und des Datenbankschemas ausgegeben. Andere Argumente werden ignoriert.

-USER <BENUTZERNAME> -PASSWORD <KENNWORT>

Mit diesem Parameter geben Sie den Benutzernamen und das Kennwort des PPM-Benutzers an, der den Import ausführt. Der Benutzer muss das Funktionsrecht **Datenimport** haben.

-CLIENT <MANDANTENNAME>

Mit diesem Parameter geben Sie den PPM-Mandanten an, für den Sie die importierten Dimensionswerte speichern möchten. Wenn Sie diese Option nicht verwenden, wird der Standardmandant verwendet.

-MODE IMPORT

Mit diesem Parameter geben Sie an, dass die Quelldateien mit prozessinstanzunabhängigen Daten importiert werden.

-FILE <FILE1>[,<FILE2>...]

Mit diesem Parameter geben Sie die Importdatei(en) an. Die Quelldateien können auch ZIP-Dateien sein, die eine oder mehrere XML-Datei(en) vom gleichen Datenformat enthalten.

-RECOVERYFILE {YES|NO}

Mit diesem Parameter geben Sie an, ob nach dem Import prozessinstanzunabhängiger Dimensionswerte die für die Prozessanalyse relevanten Wiederherstellungsdateien des Analyseservers erstellt werden sollen. Vorgabewert ist **yes**.

Sie können die Performance mehrerer, hintereinander ausgeführter Dimensionsdatenimporte steigern, indem Sie für alle vorangegangenen Importe das Erstellen der Wiederherstellungsdateien des Analyseservers unterdrücken. Geben Sie dazu die Option **-recoveryfile no** an. Lediglich der zuletzt ausgeführte Import erstellt durch Fehlen der Option (oder Angabe von **-recoveryfile yes**) die für die Prozessanalyse relevanten Wiederherstellungsdateien des Analyseservers. Bitte achten Sie darauf, dass in jedem Fall nach Abschluss des Importes die für die Prozessanalyse relevanten Wiederherstellungsdateien des Analyseservers erstellt werden. Das Erstellen aller Wiederherstellungsdateien können Sie auch durch den Kommandozeilenaufruf **runppmadmin** mit Option **-recoveryfile force** erzwingen.

-FORMAT {XML|CSV}

Mit diesem Parameter geben Sie das verwendete Datenformat an.

-CSVCHAR "<ZEICHEN>"

Mit diesem Parameter geben Sie das Trennzeichen für Datenfeldwerte des CSV-Importformats an (Vorgabewert ist das Komma).

-OVERWRITE

Mit diesem Parameter bestimmen Sie, dass bestehende **Beschreibungen** von Textdimensionsstufen mit geänderten (auch leeren) Werten überschrieben werden. Die **Schlüssel** (IDs) von Textdimensionen können nicht überschrieben werden.

-REPLACE

Mit diesem Parameter können Sie bereits importierte Dimensionsdaten überschreiben, nicht in der Importdatei enthaltene Dimensionen werden gelöscht. Dieser Parameter impliziert den Parameter **-overwrite**. Nach dem Import werden die für die Prozessanalyse relevanten Wiederherstellungsdateien des Analyseservers neu erstellt. (Standardverhalten).

5.3 Data Analytics

Mit Hilfe von Data Analytics können zusätzlich zur Prozess-, Funktions- und Interaktionsanalyse komplett prozessunabhängige Daten ausgewertet werden. Data Analytics ermöglicht die Analyse umfangreicher Datensätze, die in tabellarischer Form vorliegen und aus mehreren, miteinander verknüpften Tabellen bestehen. Die Analysekriterien, d. h. Dimensionen und Kennzahlen, ergeben sich in Data Analytics durch die Tabellenstruktur der Datenbasis, wobei jede Tabellenspalte ein Analysekriterium darstellt.

Informationen zum Import von Data-Analytics-Daten erhalten Sie in der technischen Dokumentation **PPM Data Analytics**.

6 Behandlung großer EPKs

6.1 Import großer EPKs

Wenn beim Importieren großer EPKs die maximal zulässige Anzahl von Funktionen pro EPK überschritten wird, wird eine Fehlermeldung angezeigt. Der Import wird jedoch nicht abgebrochen.

Der Schwellwert für die maximal zulässige Anzahl von Funktionen in einer EPK wird durch den Konfigurationsparameter **KI_EPC_FUNCTION_COUNT_THRESHOLD** kontrolliert, der in der Datei **EpkImport_settings** definiert wird. Standardwert für diesen Parameter ist **500**. Dieser wird herangezogen, wenn der Parameter komplett fehlt oder ein Wert **<= 0** eingetragen ist. Fällt der Parameter auf den Standardwert zurück, wird eine entsprechende Warnung in das Protokoll geschrieben.

Sobald sich eine große EPK in der Datenbank befindet, sind XML-Import, PPM-Import und Prozessimport im Modus **DEFAULT** blockiert. Das heißt, es wird eine entsprechende Fehlermeldung ausgegeben und der Import bricht ab. Der Prozessimport im Modus **RECOVERIMPORT** ist nicht betroffen.

Um die Blockade aufzuheben, kann man den Konfigurationsparameter auf einen ausreichend hohen Wert einstellen. Dies kann zu höherem Hauptspeicherbedarf des Systems führen. Alternativ kann man die problematische EPK löschen (Siehe folgender Abschnitt).

6.2 Löschen großer EPKs

Man kann eine große EPK mit dem Kommandozeilenprogramm **runppmdelete** löschen. Vorausgesetzt man hat für die Anzahl von Funktionen in einer Prozessinstanz eine Prozesskennzahl definiert. In dem Fall definiert man als Eingabe für das Kommandozeilenprogramm ein ParamSet wie folgt.

- Das ParamSet enthält die Anzahl von Funktionen **nur** als Filter
- Das ParamSet enthält die Prozessanzahl als Kennzahl
- Das ParamSet enthält einen Zeitfilter
- Das ParamSet enthält einen Prozesstyp- oder Prozesstypgruppenfilter

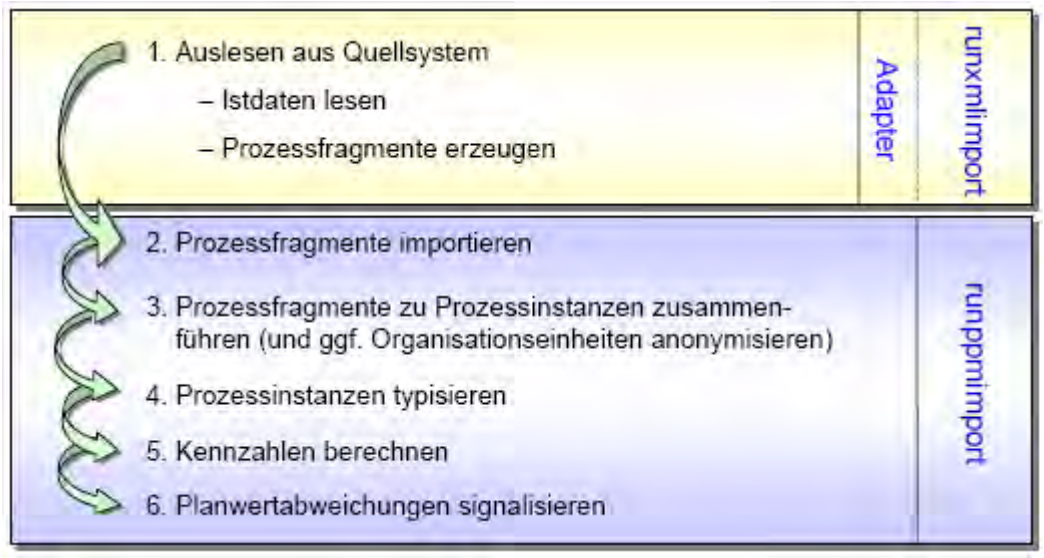
Die Kennzahl kann auch nachträglich (d. h. wenn die Importe bereits blockiert sind) definiert werden. In diesem Fall muss man die EPKs mit dem Kommandozeilenprogramm **runppmimport –keyindicator new** neu berechnen.

7 Anhang

7.1 Aufbau eines Process Warehouse

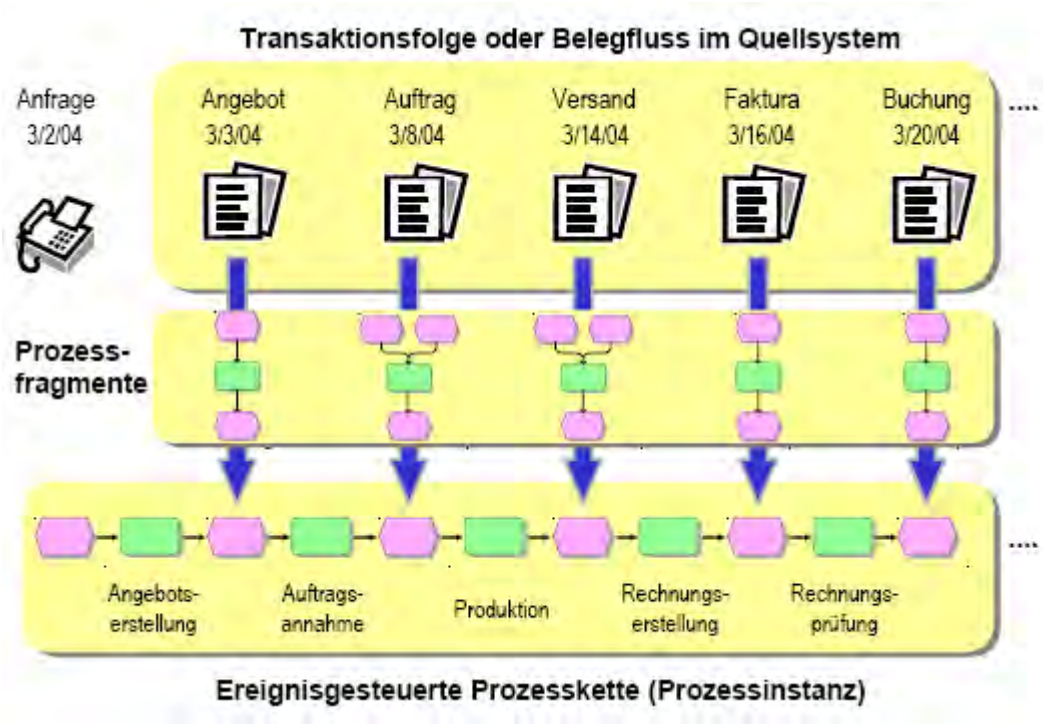
Dieses Kapitel gibt einen Überblick über den Aufbau eines Process Warehouse in PPM und erläutert anschließend jeweils kurz die einzelnen Schritte. Die notwendige Installation und Konfiguration des PPM-Systems wird als gegeben vorausgesetzt.

Der Aufbau eines Process Warehouse umfasst die folgenden Schritte:



1. Zunächst werden aus dem Quellsystem die Istdaten ausgelesen und dem weiteren Import in Form von Prozessfragmenten zur Verfügung gestellt.
2. Die Prozessfragmente werden importiert.
3. Im nächsten Schritt werden in allen importierten Daten die zu jeweils einem Geschäftsvorfall gehörenden Fragmente gesucht und zu einer Prozessinstanz zusammengeführt. Dabei werden Objektattribute an die Prozessinstanz kopiert. Beim Zusammenführen der Prozessfragmente können Informationen über die tatsächlichen Bearbeiter anonymisiert werden.
4. Anschließend werden die generierten Prozessinstanzen klassifiziert: Gleichartige Prozessinstanzen werden Prozesstypen zugeordnet, welche wiederum zu Prozesstypgruppen zusammengefasst werden.
5. Für jede Prozessinstanz werden die definierten Kennzahlen berechnet und in Info Cubes performant abgelegt.
6. Planwertüberschreitungen werden überprüft und gegebenenfalls signalisiert.

Beim Aufbau des Process Warehouse werden somit aus Transaktionsfolgen und Belegflüssen der Quellsysteme Prozessfragmente erzeugt und zu Prozessinstanzen zusammengeführt. Diese Prozessinstanzen dienen als Bezugsobjekte für Analysen und Auswertungen der Prozessleistung. Sie können in PPM als Prozessmodelle angezeigt werden.



Als Darstellung für eine Prozessinstanz wird die Ereignisgesteuerte Prozesskette (EPK) verwendet, die aus einer Verkettung von Objekten besteht. Sowohl die Objekte als auch die Prozessinstanz selbst können Attribute tragen, in denen die Instanzdaten gespeichert sind. Anhand dieser Daten werden die Prozessinstanzen typisiert und die Kennzahlen berechnet. Die berechneten Kennzahlen werden wiederum in Attributen der Objekte und der Prozessinstanzen gespeichert. Damit sind die Attribute die eigentlichen Informationsträger des PPM-Systems. Die verfügbaren Attribute werden in der PPM-Konfiguration definiert. Die Konfiguration setzt sich aus Standardattributen, wie z. B. **Prozessidentifizierer** und **Endzeit** sowie frei definierbaren, systemspezifischen Attributen zusammen.

EXKURS EPK:

Eine ereignisgesteuerte Prozesskette (EPK) ist ein von Prof. Scheer entwickelter Modelltyp zur grafischen Beschreibung des zeitlich-logischen Ablaufs eines Leistungserstellungsprozesses. Sie beruht auf folgenden Annahmen:

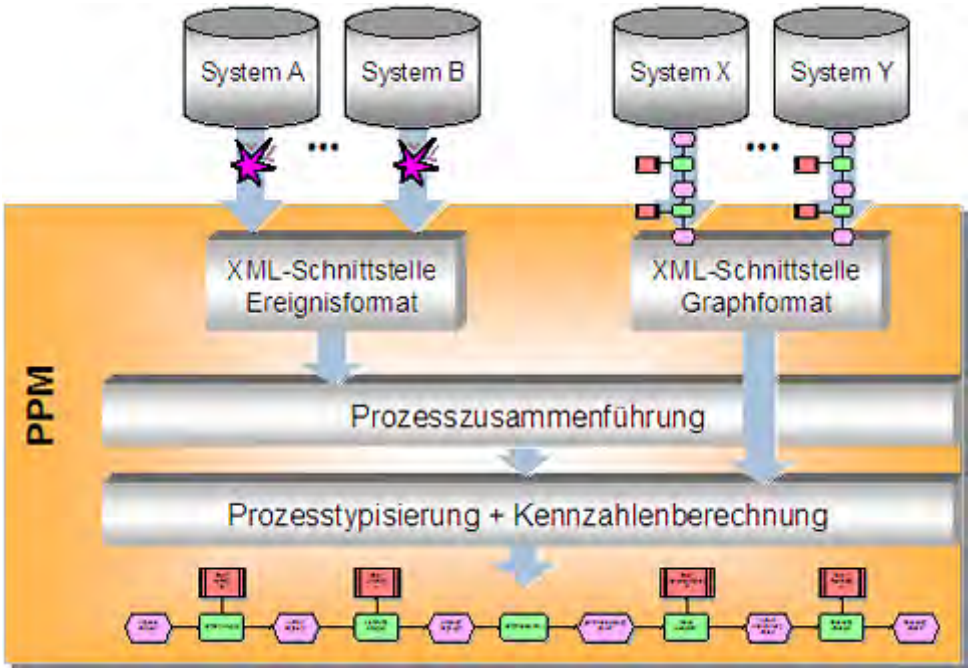
- Jede Aktivität innerhalb eines Prozesses wird durch eine betriebswirtschaftlich relevante Zustandsänderung eines Informationsobjekts bewirkt. Jede Aktivität kann als Ergebnis eine betriebswirtschaftlich relevante Zustandsänderung eines Informationsobjektes mit sich bringen.
- Der Zustand eines betriebswirtschaftlich relevanten Informationsobjekts wird grafisch durch ein Objekt des Typs **Ereignis** beschrieben.

- Zur grafischen Darstellung von Aktivitäten werden Objekte des Typs **Funktion** verwendet. Durch Hintereinanderschalten von Ereignissen und Funktionen und das Verbinden dieser Objekte durch gerichtete Kanten wird der Kontrollfluss des Prozesses grafisch dargestellt.
- Da ein Ereignis mehrere Funktionen auslösen und eine Funktion wiederum mehrere Ereignisse als Ergebnis haben kann, werden an solchen Verzweigungen Und-, Oder- bzw. Exklusiv-Oder-Verknüpfungen eingefügt. Sie verdeutlichen die logische Beziehung, die zwischen den aufeinanderfolgenden Objekten besteht.
- Organisationseinheiten beschreiben die Gruppen von Bearbeitern, welche eine Funktion ausführen.

7.1.1 Prozessfragmente generieren

Prozessdaten können auf unterschiedliche Weise aus den Anwendungssystemen gewonnen werden.

Schema: Datenextraktion



Bei SAP R/3 greifen spezielle Adapter online auf operative R/3-Belegdaten zu und transformieren die SAP-Belegflüsse in Prozessbeschreibungen. Diese Adapter werden hier nur der Vollständigkeit halber erwähnt.

Bei allen anderen Anwendungssystemen werden Ablaufdaten durch eine generalisierte XML-Importschnittstelle offline mittels einer Datei ins PPM-System importiert.

Die XML-Importschnittstelle kann zwei verschiedene Typen von XML-Dateien verarbeiten: XML-Dateien im PPM-Graphformat und im PPM-System-Event-Format.

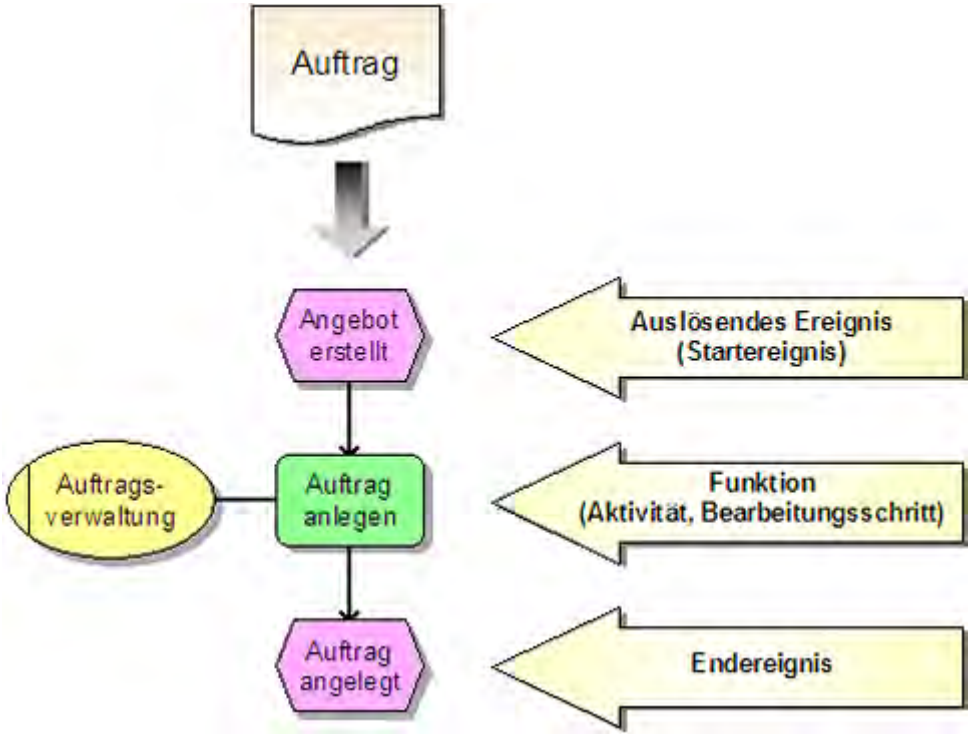
GRAPHFORMAT

Das Graphformat wird verwendet, um bereits strukturierte Prozessdaten aus prozessorientierten Anwendungssystemen (z. B. Workflow-Systemen) zu übertragen. Der anwendungssystemspezifische Adapter generiert XML-Dateien, in denen Prozessinstanzen einschließlich ihrer Ablauflogik im PPM-Graphformat beschrieben werden.

SYSTEM-EVENT-FORMAT

Das System-Event-Format wird für alle tätigkeitsorientierten Anwendungssysteme eingesetzt, bei denen die prozessbildende Information (Ablauflogik) nicht ausgelesen werden kann. Die System-Events werden als Prozessfragmente interpretiert.

Beispiel: Zuordnung eines Prozessfragments zu einem System-Event Auftrag angelegt



Ein Prozessfragment beschreibt einen Teil eines Gesamtprozesses. Es enthält mindestens eine Funktion mit ihren auslösenden und erzeugten Ereignissen. Ein Prozessfragment kann als ein einzelner Bearbeitungsvorgang, eine Aktivität oder eine Transaktion innerhalb eines Gesamtprozesses interpretiert werden. Neben dem zeitlich-logischen Ablauf kann ein Prozessfragment auch Informationen über den Bearbeiter einer Funktion in Form von Organisationseinheiten beinhalten.

Beim Auslesen aus den Quellsystemen wird jedem System-Event anhand der Mapping-Definition ein Prozessfragment zugeordnet. Dabei werden die Instanzdaten des System-Event als Attribute an die Objekte des Prozessfragments geschrieben.

Anhand der Attributwerte der Fragmentereignisse werden die einzelnen Prozessfragmente anschließend zu Prozessinstanzen zusammengesetzt.

Prozessfragmente werden nur dann in die PPM-Datenbank importiert, wenn sie bestimmten Prozessinstanzen zugeordnet werden können.

7.1.2 Prozessfragmente zusammenführen

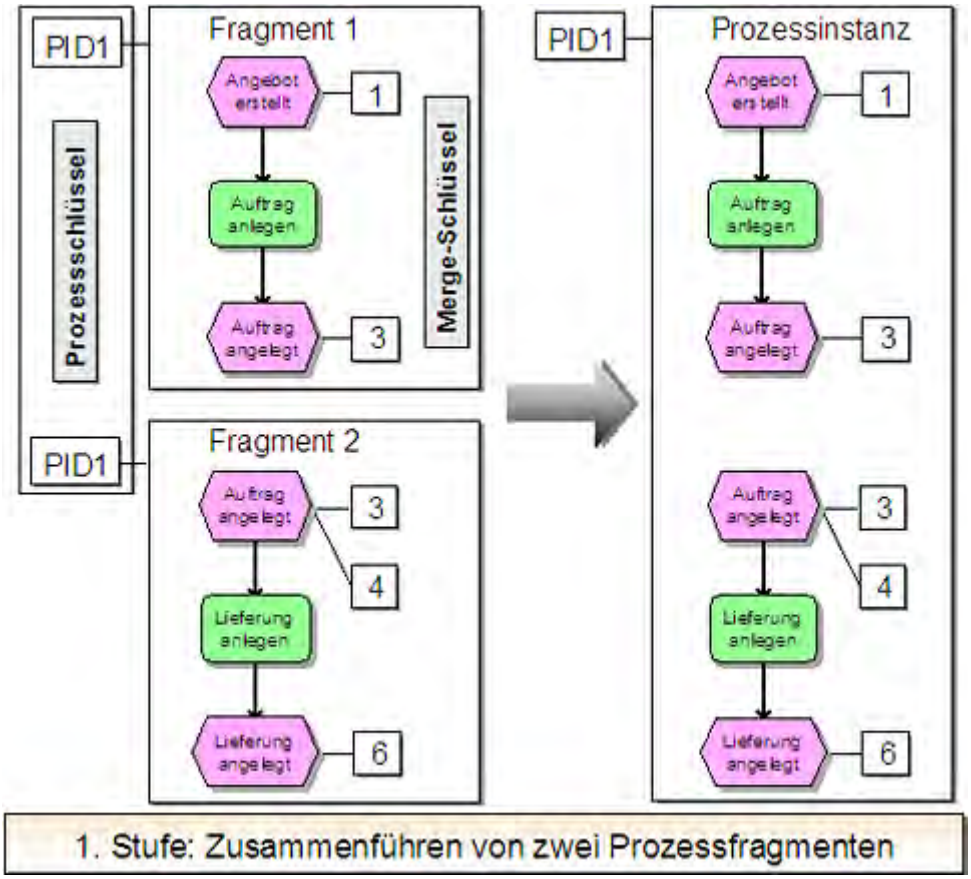
Dieses Kapitel beschreibt das Zusammenführen der importierten Prozessfragmente zu Prozessinstanzen. Zusammengeführte Prozessfragmente entsprechen Geschäftsprozessen, die tatsächlich durchlaufen wurden. Sie werden Prozessinstanzen genannt und werden analog zu den Fragmenten in der bereits aus ARIS bekannten Notation der EPK dargestellt.

Beim Zusammenführen werden Attribute an die Prozessinstanzen kopiert und personenbezogene Bearbeitungsdaten anonymisiert.

Der Merge-Vorgang läuft in zwei Stufen ab:

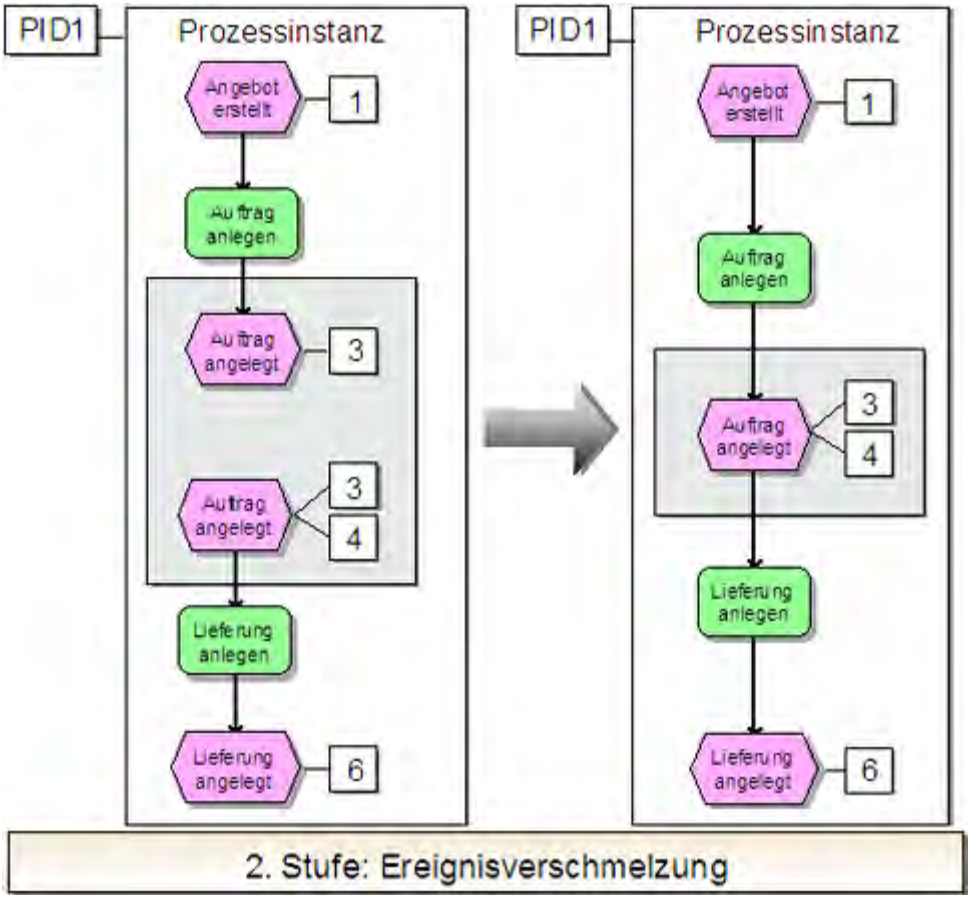
Im ersten Schritt werden mit Hilfe von Prozessschlüsseln die Prozessfragmente, die zu derselben Prozessinstanz gehören, identifiziert und in eine Prozessinstanz kopiert.

Beispiel: Schritt 1 des Prozess-Merge



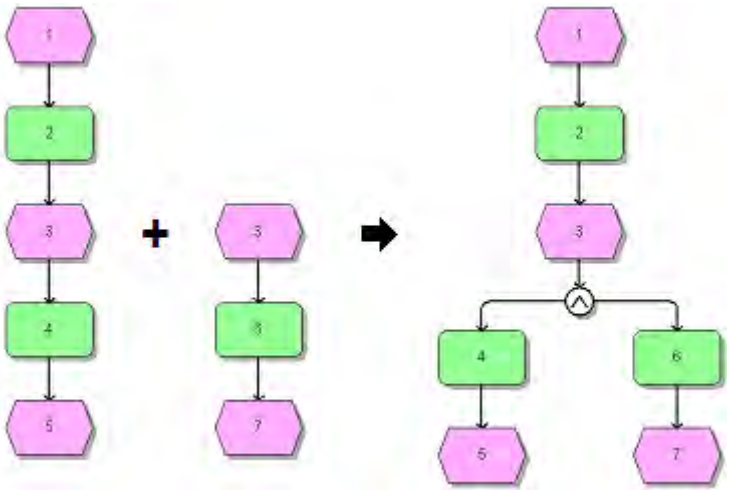
Im zweiten Schritt werden die unverbundenen Prozessfragmente durch Verschmelzen der Merge-Ereignisse miteinander verkettet. Merge-Ereignisse sind Ereignisse, für die Merge-Schlüssel berechnet wurden.

Beispiel: Schritt 2 des Prozess-Merge



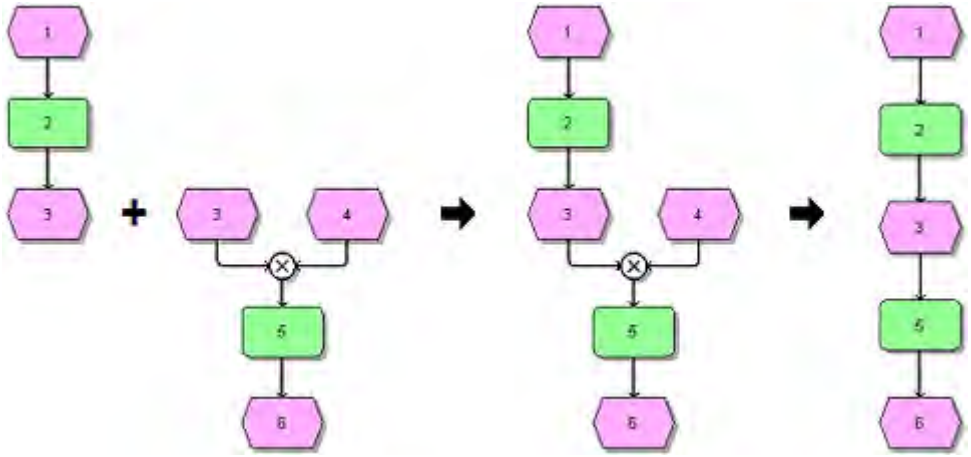
Die eventuell durch den Merge-Vorgang entstandenen Verzweigungen werden durch Regeln erweitert bzw. nicht erforderliche Regeln werden gelöscht.

Beispiel: Merge mit UND-Regel



In folgendem Beispiel wird beim Merge eine zusammenführende XOR-Regel gelöscht, da Prozessinstanzen reale Geschäftsvorfälle darstellen und in ihnen keine XOR-Regeln vorkommen können. Es werden die XOR-Regel und das Ereignis gelöscht, das keine vorangehende Funktion besitzt (Ereignis 4).

Beispiel: Merge mit XOR-Regel



Da beim Zusammenführen der importierten Daten nicht zwischen den einzelnen Quellsystemen unterschieden wird, können Prozesse mit PPM systemübergreifend betrachtet werden.

Wie Prozess- und Merge-Schlüssel berechnet werden, wird in der XML-Konfiguration mandantenspezifisch über Regeln eingestellt.

7.1.2.1 Kopieren der Prozessinstanzattribute

Beim Zusammenführen von Prozessfragmenten zu Prozessinstanzen werden Attribute von Objekten an die Prozessinstanzen kopiert. Prozessinstanzattribute bilden die Grundlage für die Berechnung der einzelnen Kennzahlen in Abhängigkeit von den Dimensionen.

Beim Import von vollständigen Prozessinstanzen im Graphformat entfällt der Merge-Vorgang. Diese Prozessinstanzen tragen bereits Prozessinstanzattribute.

7.1.2.2 Anonymisieren der Organisationseinheiten

Auf Instanzebene sind die tatsächlichen Bearbeiter einer Funktion bekannt. Da die Bearbeiter aus datenschutzrechtlichen Gründen oft nicht angezeigt werden sollen und unerheblich für die Kennzahlenberechnung sind, besteht die Möglichkeit, die Bearbeiter zu anonymisieren. Dazu müssen in PPM alle Mitarbeiter einer Organisationseinheit zugeordnet werden. Beim Datenimport werden dann im Zuge der Prozesszusammenführung die Mitarbeiter durch die zugehörige Organisationseinheit ersetzt. Die Information über den tatsächlichen Bearbeiter geht dabei verloren.

7.1.3 Prozesse typisieren

Nach dem Zusammenführen der importierten Prozessfragmente zu Prozessinstanzen müssen diese klassifiziert werden, um eine sinnvolle Kennzahlenauswertung zu ermöglichen. Dazu werden sie in eine frei definierbare zweistufige Hierarchie eingeordnet: Die Prozessinstanzen werden Prozesstypen zugeordnet, die wiederum zu Prozesstypgruppen zusammengefasst werden. Eine Prozesstypgruppe kann somit mehrere Prozesstypen enthalten, die wiederum mehrere Prozessinstanzen beinhalten. Eine Prozessinstanz kann jedoch nur genau einem Prozesstyp und damit auch nur einer Prozesstypgruppe zugeordnet werden.
Die Regeln zur Typisierung sind quellsystemspezifisch frei definierbar. Die Informationen zur Typisierung werden in Prozessinstanzattributen gespeichert.

Die Zuordnung der Prozesstypen und Prozesstypgruppe wird in PPM im Prozessbaum dargestellt. In der Konfiguration des Prozessbaums wird darüber hinaus festgelegt, welche Kennzahlen und Dimensionen für die einzelnen Prozesstypen zur Verfügung stehen.

Prozessinstanzen stellen die tatsächlich abgelaufenen Geschäftsvorfälle dar und werden aus den importierten Prozessinstanzfragmenten zusammengesetzt. Gleichartige Prozessinstanzen werden zu Prozesstypen zusammengefasst, die wiederum einer Prozesstypgruppe zugeordnet sind. Der Begriff Prozess, wie er aus der ARIS-Modellierung bekannt ist, ist im Rahmen von PPM nicht spezifiziert und sollte somit nicht verwendet werden.

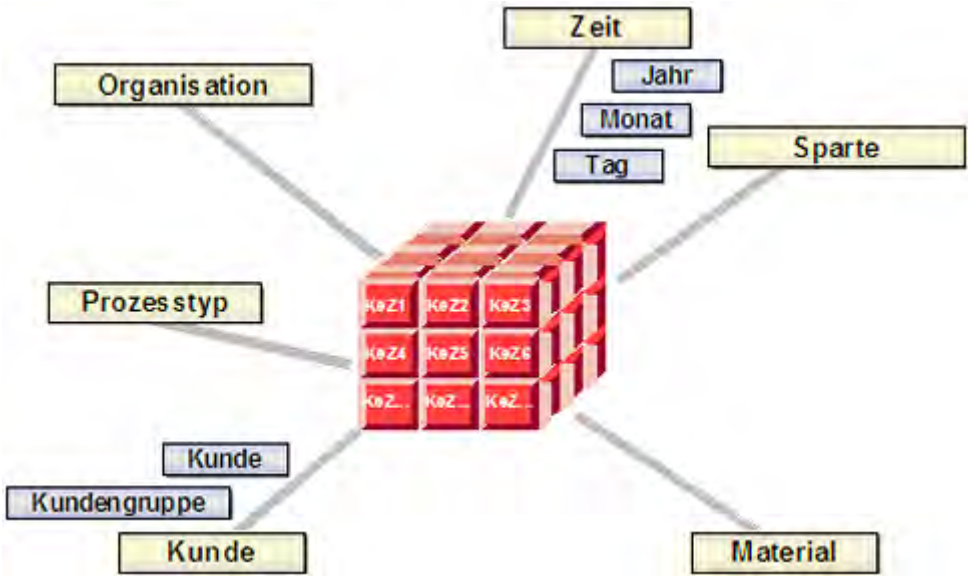
7.1.4 Kennzahlen berechnen

Die Analyse einer Prozessinstanz wird auf Basis von berechneten Kennzahlen durchgeführt. Unter Kennzahlen versteht man die aus Messgrößen berechneten Eigenschaften eines Prozesses oder einer Funktion. Es wird zwischen Funktions- und Prozesskennzahlen unterschieden. Neben vordefinierten Standardkennzahlen, zu denen die Kennzahlen **Prozessanzahl** und **Prozesslaufzeit** gehören, können in der Konfiguration beliebige Kennzahlen mit der zugehörigen Berechnungsvorschrift definiert werden.

Dimensionen sind Kriterien, nach denen die Kennzahlen von Prozessinstanzen und Funktionen differenziert werden können, z. B. der Prozesstyp oder der Ausführungsort.

Für die im Prozessbaum enthaltenen Prozesstypen werden die angegebenen Kennzahlen in Abhängigkeit von den Dimensionen berechnet. Das Ergebnis wird in der PPM-Datenbank dauerhaft gespeichert. Dabei werden die Werte in sogenannten Datenwürfeln (**Data Cubes)** abgelegt, um performante Abfragen zu gewährleisten.

Beispiel: Data Cube



7.1.5 Planwerte überprüfen

Das PPM-System bietet zur Prozessüberwachung die Definition von Plan- und Alarmwerten an. Planwerte beziehen sich auf eine Menge von Prozessinstanzen, Alarmwerte auf eine einzelne Prozessinstanz. Es kann zu kritischen Alarmwertüber- bzw. -unterschreitungen einzelner Prozessinstanzen kommen, obwohl die Planwerte der zugehörigen Menge der Prozessinstanzen eingehalten werden.

Die prozesstypbezogene Überprüfung der Planwertüberschreitung bzw. -unterschreitung einzelner Kennzahlen schließt den Datenimport des PPM-Systems ab. Die berechneten Kennzahlen werden mit den in der PPM-Oberfläche definierten Planwerten verglichen und die dort angegebenen Aktionen, wie z. B. das Senden einer E-Mail an den Prozessverantwortlichen (Planwertnachricht), ausgeführt.