Technische Universität Dresden

Fakultät Informatik
Institut für Systemarchitektur
Professur für Rechnernetze
Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill

Diplomarbeit

zur Erlangung des akademischen Grades
Diplomingenieur für Informatik

Geschäftstransaktionsbasierte Formulierung von Zugriffsrechten

Kerstin Gabsch
(Geboren am 28.Februar 1981 in Dresden)

Mat.-Nr.: 2845551
Betreuer: Dipl.-Medien-Inf. Eberhard Oliver Grummt

Dresden, 14.Januar 2009

TECHNISCHE UNIVERSITÄT DRESDEN
Fakultät Informatik, Institut für Systemarchitektur, Lehrstuhl Rechnernetze

Aufgabenstellung für die Diplomarbeit

Name, Vorname: Gabsch, Kerstin
Studiengang: Informatik
Matr.-Nr.: 2845551

Thema: Geschäftstransaktionsbasierte Formulierung von Zugriffsrechten

Einführung

In globalen Wertschöpfungsketten zeichnet sich der Trend zur Verwendung der RFID-Technologie ab, welche das automatische Erfassen (*Auto-ID*) von mit Transpondern versehenen Handelsgütern und Ladungsträgern ermöglicht. Die entsprechenden Informationen werden von jeder Firma in Form von Ereignissen in einem lokalen, so genannten Auto-ID-Repository gespeichert. Dabei repräsentiert ein Ereignis jeweils einen Vorfall, z. B. dass eine Lieferung ein Lager erreicht hat oder dass bestimmte Kartons auf einer bestimmten Palette verladen wurden.
Um eine detaillierte Sicht auf alle für ein Unternehmen relevanten Güterflüsse zu erhalten, ist der Zugriff auf die Repositorys von Handelspartnern (Zulieferer, Transportdienstleister, Abnehmer usw.) notwendig. Um die dafür benötigte Interoperabilität zu ermöglichen, existiert der neue Standard *EPC Information Services* (http://www.epcglobalinc.org/standards/epcis), der entsprechende Ereignistypen und Austausch-Schnittstellen spezifiziert. Offen ist hingegen, wie Zugriffskontrollsysteme realisiert werden können, die den Zugriff auf EPCIS Repositorys so begrenzen, dass Handelspartner nur die für sie vom Betreiber des entsprechenden Systems freigegebenen Ereignisse abrufen können.
Bei SAP Research wurde bereits untersucht, wie über Bedingungen, welche die Attribute von Ereignissen erfüllen müssen, Berechtigungen spezifiziert und wie diese durch SQL-Query-Rewriting effizient durchgesetzt werden können.
Dieser Ansatz unterstützt allerdings nicht das semantisch reichere Konzept der *Geschäftstransaktion*. Er nutzt somit nicht die Tatsache aus, dass eine Serie von Ereignissen durch eine Geschäftstransaktion wie z. B. eine Bestellung verknüpft wird und somit die Vergabe von Zugriffsrechten potenziell vereinfacht bzw. automatisiert werden kann.

Schwerpunkte

Ziel dieser Arbeit ist es, Ansätze und konkrete Lösungsvorschläge zu erarbeiten, um die vorgenannten Limitationen bei der Berechtigungsspezifikation zu beheben und dem Administrator eines EPCIS Repositorys eine komfortablere und abstraktere Form der Formulierung von Zugriffsrechten auf Basis von Geschäftstransaktionen zu ermöglichen.
Ausgehend von der Struktur der EPCIS-Ereignisse und den in der Spezifikation verankerten Möglichkeiten, Geschäftstransaktionen mit Ereignissen zu verknüpfen (*TransactionEvent* und *businessTransactionList*) sollen typische Arten von Transaktionen und die entsprechenden logischen Dokumente (wie Bestellung, Transportauftrag, ASN etc.) systematisiert werden. Danach soll untersucht werden, inwiefern diese Zuordnung unter der Annahme, dass jeder

Transaktion ein Handelspartner und damit ein eindeutiger Benutzeraccount des EPCIS zugeordnet werden kann, für die Zugriffskontrolle auf Ereignisse verwendet werden kann. Dabei ist zu beachten, dass die Formulierung von Berechtigungen zu einem möglichst einfach sein soll (z. B. „jeder Handelspartner soll nur die Ereignisse sehen, die über eine Bestellung mit ihm verknüpft sind“), zum anderen aber bestimmte Ausnahmen und Sonderfälle darstellbar sein müssen. So sollen bestimmte Ereignisse, die in internen Produktionsprozessen anfallen, einem Handelspartner nicht preisgegeben werden, auch wenn sie ein Produkt betreffen, das später von diesem Handelspartner gekauft wird. Weiterhin ist zu beachten, dass eine Transaktion auch beendet werden kann, z. B. wenn ein Partner seine Bestellung storniert. Zu betrachten sind auch Situationen, in denen Ereignisse mit mehreren Transaktionen und mehreren Handelspartnern (z. B. Abnehmer und Transportdienstleister) gleichzeitig verknüpft sind und potenziell Interessenskonflikte auftreten.
Nach Erarbeitung entsprechender Konzepte ist die konkrete technische Formulierung von Berechtigungen in Form einer XML-basierten Sprache zu spezifizieren. Anschließend ist zu konzipieren, wie diese Berechtigungen effizient durchgesetzt werden können, z. B. durch SQL-Query Rewriting oder Umwandlung in die bei SAP Research bereits entwickelte und oben erwähnte Berechtigungssprache. Folgende Teilaspekte sind zu bearbeiten:

- Systematisierung und Bewertung des relevanten Standes der Technik in den Gebieten Berechtigungsvergabe und Zugriffskontrolle für EPCIS
- Systematisierung, welche typischen Geschäftstransaktionen und Belegtypen anfallen
- Untersuchung, ob die Einteilung in (potenziell) öffentliche Informationen und generell interne Ereignisse sinnvoll ist
- Konzeption eines Modells für die Berechtigungsspezifikation auf Basis von Geschäftstransaktionen
- Umsetzung des Modells in einer XML-basierten Sprache
- Konzeption eines Durchsetzungs- oder Generierungsmechanismus für konkrete Berechtigungen
- Prototypische Implementierung ausgewählter Teile des eigenen Konzepts
- Bewertung der eigenen Konzeption und Implementierung

Benötigte Vorkenntnisse

- Grundlegendes Verständnis für Sicherheitstechnologien
- Gute Java-Kenntnisse

Betreuer:	Dipl.-Medien-Inf. Eberhard Grummt
Verantwortlicher Hochschullehrer:	Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill
Institut:	Institut für Systemarchitektur
Lehrstuhl:	Rechnernetze
Beginn am:	14.07.2008
Einzureichen am:	14.01.09

Unterschrift des verantwortlichen Hochschullehrers

Verteiler: 1 x Prüfungsamt, 1 x HSL, 1 x Betreuer, 1 x Student

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die von mir am heutigen Tag dem Prüfungsausschuss der Fakultät Informatik eingereichte Diplomarbeit zum Thema:

Geschäftstransaktionsbasierte Formulierung von Zugriffsrechten

vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Dresden, den 14. Januar 2009

Kerstin Gabsch

Inhaltsverzeichnis

1. Einleitung

Die Organisation EPCglobal hat es sich zur Aufgabe gemacht, die Abläufe von globalen Wertschöpfungsketten, die die RFID-Technologien verwenden, zu verbessern und eine komplette sowie technisch ausgereifte Spezifikation zur Verfügung zu stellen. Diese Spezifikation kann von Industriebetrieben unterschiedlicher Branchen angewendet werden. Eine auf globalen Standards basierende Technologie, das EPCglobal-Network, ist entstanden. Das EPCglobal-Network kombiniert die RFID-Tags mit einer existierenden Kommunikationsnetzwerkinfrastruktur und dem Electronic Produkt Code, kurz EPC, zur eindeutigen Identifikation eines Objektes [GS108].

Unternehmen, die am EPCglobal-Network teilnehmen, besitzen ihr eigenes EPCIS-Repository, in dem Informationen bezogen auf EPC-Objekte in Form von EPCIS-Events gespeichert werden.

EPC-Objekte sind Handelsgüter und Ladungsträger, die mit Transpondern gekennzeichnet sind. Unternehmen können diese Objekte durch RFID-Reader automatisch erfassen. Die erhaltenen Informationen, infolge der Objekterfassung in einem Unternehmen, werden in Form von Ereignissen, so genannten EPCIS-Events, in die lokale Datenbank des Unternehmens gespeichert. Ein Ereignis gibt an Was, Wann, Wo und Warum passiert ist. Zum Beispiel, die Kartons X,Y und Z wurde am 12.12.2008 um 10.00Uhr im Versandlager auf Palette W für den Transport zusammengefasst.

Der Austausch dieser Informationen zwischen Unternehmen trägt zur Verbesserung von Wertschöpfungsketten bei. Der EPC *Information Services*, kurz EPCIS, Standard unterstützt die Umsetzung und Realisierung der Behälterverfolgung, das Promotionmanagement und die Produktauthentifizierung. Des Weiteren ermöglicht die Spezifikation den elektronischen Lieferavis, die Überwachung und optimale Abwicklung der Produktionskette, sowie die Verbesserung des Retouremanagements. EPCIS bietet standardisierte Schnittstellen zu Erfassung und zum Austausch von produktrelevanten Informationen des jeweiligen Unternehmens einer Wertschöpfungskette. Der Informationsaustausch erfolgt mittel spezifizierter Ereignistypen (EPCIS-Eventtypen) [GS107].

Der EPCIS-Standard lässt jedoch offen, wie Zugriffskontrollsysteme realisiert werden können, um die unternehmensspezifischen EPCIS-Repositories von unerwünschten Zugriffen zu schützen. Für Handelspartner eines Unternehmens soll der Zugriff auf das EPCIS-Repository begrenzt sein, so das jeder Handelspartner nur Ereignisse abrufen kann, die vom System des Unternehmens freigegeben sind. In [GS07] und [Sch08] wurde die Policysprache XACML und deren Erweiterung zu aidXACML für die Berechtigungsdefinition in EPCIS-Systemen vorgestellt. Die Verwendung von aidXACML realisiert die attributgenaue und regelbasierte Eventfreigabe für EPCIS-Repositories in EPCIS-Systemen. Die erweiterte Policysprache biete ein große Flexibilität in der Definition von Zugriffsberechtigungen. Zur Zeit erfolgt die Erstellung von Zugriffsberechtigungen durch den Administrator des EPCIS-Repository manuell für jedes relevante Ereignis. Diese Art der Berechtigungsformulierung ist zum Einem zeitaufwendig und zum Anderen kostenintensiv. Um dem Administrator eine komfortablere und abstraktere Form für die Formulierung von Zugriffsberechtigungen zur Verfügung zu stellen und deren Vergabe im Kontext der Geschäftstransaktionen zu vereinfachen, werden im Kapitel 2. grundlegende

Standards, Berechtigungssprachen, bestehende Zugriffskontrollsysteme und XML-Transformationskonzepte vorgestellt. Sie dienen der Vorlage für die Erstellung des Konzeptes der Policy Instantiation Engine, kurz PIE. Anschließend an die Betrachtung der Grundlagen erfolgt im Kapitel 3. die Systematisierung von Geschäftstransaktionen und die Betrachtung des Zusammenhangs zwischen diesen Transaktionen und den Berechtigungen, sowie deren Vergaben und Entzug. Des Weiteren werden in diesem Kapitel kritische Punkte für die Automatisierung der Formulierung von Zugriffsberechtigungen aufgezeigt und daraus Anforderungen für die Konzeption der PIE gestellt, die in Kapitel 4. präsentiert wird. Dieses Kapitel enthält den konkreten Lösungsansatz zur Beseitigung der zuvor genannten Limitationen der Berechtigungsvergabe für EPCIS-Repositories. Die abstrakte Formulierung von Zugriffsberechtigungen wird durch XSLT-Stylesheets vorgenommen, die zur Generierung von Zugriffsberechtigungen verwendet werden. Für jede Geschäftstransaktion, die bei der Berechtigungsformulierung berücksichtigt wird, existiert ein entsprechendes XSLT-Stylesheet. Mit diesem XSLT-Stylesheet werden die unternehmens- und handelspartnerbezogenen, sowie produktspezifischen Formulierungen von Zugriffsberechtigungen realisiert. Die Umsetzung und Durchsetzung der Generierung nimmt die PIE anhand der einzuhaltenden Programmabläufe vor. Dieses Abläufe sind ausführlich in der Konzeption erläutert und dienen als Grundlage für die Implementierung. Ausgewählte Teile des entwickelten Konzepts wurden prototypisch implementiert. Im Kapitel 5. sind Aspekte der Implementierung dargelegt. Es wird auf die PIE-Datenbank und die EPC-Ermittlung im Besonderen eingegangen. Die Bewertung und Validierung des PIE-Konzeptes erfolg im Kapitel 6., anschließend wird mit Beispielen, das Konzept der PIE veranschaulicht.

Im Rahmen dieser Arbeit wurde ein vollständiges und funktionales Konzept für eine komfortablere und abstraktere Form der Formulierung von Zugriffsberechtigungen entwickelt, welches die Semantik von Geschäftstransaktionen für die Automatisierung nutzt.

2. Grundlagen

In dieser Diplomarbeit wird ein Konzept für die Formulierung von Zugriffsberechtigungen, basierend auf Geschäftstransaktion, im Rahmen der Zugriffskontrolle auf EPCIS-Repositories, entwickelt. Dieses Kapitel gibt einen Überblick über das EPCIS-System und über XML-basierte Berechtigungssprachen. Des Weiteren wird auf den Stand der Technik, zur Realisierung von Zugriffsberechtigungen, eingegangen. Anschließend werden verschiedene Ansätze und Standards präsentiert, die als Vorlage für das zu entwickelnde Konzept verwendet werden.

2.1. EPC Information Services

Der EPCglobal-Standard Electronic Product Code Information Services (EPCIS) [EPC07] spezifiziert Schnittstellen für den Austausch von EPC-bezogenen Daten und die Struktur der EPCIS-Events. Die EPCIS-Spezifikation definiert hingegen nicht, das Datenbankschema zur Speicherung von EPCIS-Daten für EPCIS-Repositories [GS106].

Das EPCIS ist ein wesentlicher Teil des EPCglobal-Networks (siehe Abbildung 2.1.1) und ermöglicht mehr Transparenz in der Bewegung, in der Lokalisierung und in der Verteilung, von mit EPC gekennzeichneten Objekten, in Wertschöpfungsketten (engl. Supply Chains).

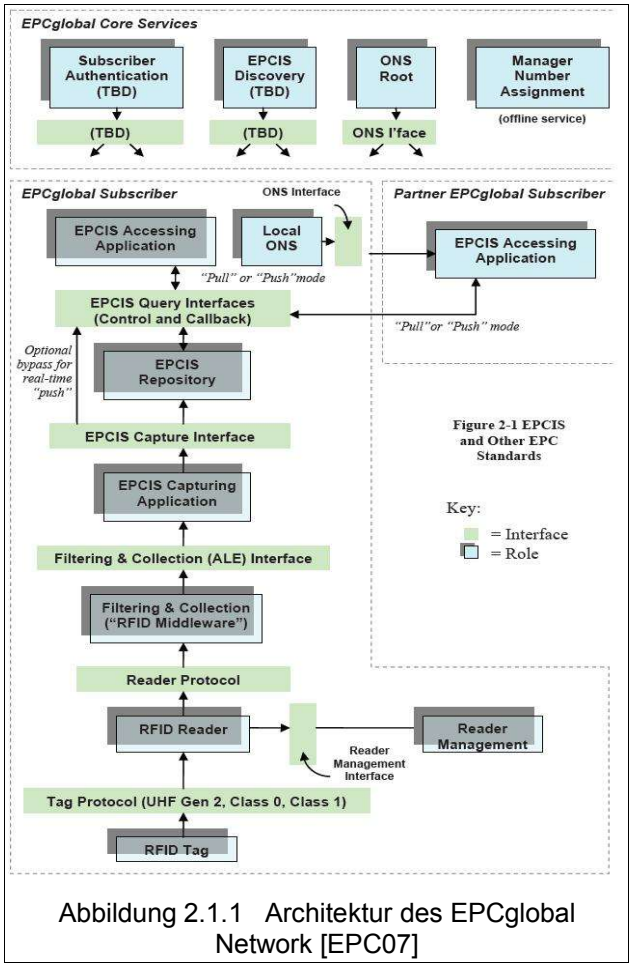

Abbildung 2.1.1 Architektur des EPCglobal Network [EPC07]

2.1.1. EPCIS-Komponenten und -Schnittstellen

Das EPCIS-System beinhaltet die EPCIS Capturing Application, das EPCIS-Repository und die EPCIS Accessing Application, sowie die EPCIS Interface Capture und EPCIS Query Interface.

Die EPCIS Capturing Application wandelt relevante Daten durch Einbeziehung von Kontextinformationen in EPCIS-Events um und bereitet diese zur Speicherung im EPCIS-Repository vor. Das Capture Interface definiert die Übergabe der EPCIS-Events der EPCIS Capturing Applications in das EPCIS-Repository oder über den Bypass direkt an die Applikationen. Das EPCIS-Repository speichert EPCIS-Events von einer oder mehreren EPCIS Capturing Applications und hält sie für Abfragen von EPCIS Accessing Applications bereit. Das EPCIS Query Interface ist die Schnittstelle zwischen dem EPCIS-Repository und den EPCIS Accessing Applications, sowie den Teilnehmern des EPCglobal-Networks. Diese Schnittstelle legt die Datenabfrage auf das eigene EPCIS-Repository unternehmensintern und -übergreifend fest. Das Interface verfügt über einen synchronen und asynchronen Arbeitsmodus zur Datenabfrage. Im synchronen oder „on-demand“-Modus wird die Client-Anfrage durch das EPCIS Query Control Interface sofort beantwortet. Im asynchronen oder mit „standing request“ bezeichneten Modus, wird eine periodische Abfrage vereinbart. Das heißt, der Client bekommt in periodischen Abständen die Ergebnisse der Abfrage vom EPCIS Query Callback Interface übermittelt. Das EPCIS Query Callback Interface wird auch eingesetzt, um die vom Capture Interface entgegengenommenen EPCIS-Events direkt an EPCIS Accessing Applications oder an Teilnehmer des EPCglobal-Network weiter zu leiten. Durch EPCIS-Applikationen können EPCIS-Events abgefragt werden, die zur Steuerung und Überwachung von Geschäftsprozessen, sowie zum Rückverfolgen von EPC-Objekten genutzt werden [GS106], [EPC07].

Die Tabelle 2.1.1 enthält Bindungen, die die konkrete Umsetzung der Schnittstellen spezifizieren. Die angegebenen Bindungen basieren auf den Anforderungen des Datenmodells für EPCIS-Events und EPCIS-Interfaces [GS106].

Typ	Binding
Kernereignis, Kernabfrage	XML
EPCIS Capture Interface	Message Queue, HTTP
EPCIS Query Control Interface	SOAP over HTTP(WSDL), XML over AS2
EPCIS Query Callback Interface	XML over HTTP, XML over HTTPS, XML over AS2

Tabelle 2.1.1 Umsetzung der Schnittstellen [GS106]

2.1.2. EPCIS-Event

EPCIS-Events können durch interne oder externe Applikationen abgefragt werden. Für die Eventabfrage sind jedoch entsprechende Zugriffsrechte notwendig. Im Kapitel 2.2. wird die Realisierung der Zugriffskontrolle im EPCglobal-Network beschrieben.

Die Abbildung 2.1.2 zeigt ein EPCIS-Event in XML-Darstellung. Es gibt an, dass am 12.12.2008 um 0:25 Uhr, am *readPoint* mit der ID urn:epc:id:gid:3721.810.105, die in der *epcList* angegeben Objekte beobachtet wurden. Dieses EPCIS-Event ist ein ObjectEvent, welches einen bestimmten Typ von EPCIS-Events repräsentiert. Weitere Ereignistypen sind das AggregationEvent, das QuantityEvent und das TransactionEvent. Alle diese Eventtypen werden für verschiedene Anwendungsbereiche verwendet. Zur Übermittlung von EPCIS-Events kann das standardisierte EPCIS-Dokument genutzt werden. Das Schema des XML-basierenden EPCIS-Dokuments ist im Anhang 8.3. dargestellt.

```
<ObjectEvent>

   <eventTime>2008-12-12T00:25:130Z</eventTime>

   <eventTimeZoneOffset>+01:00</eventTimeZoneOffset>

   <epcList>
      <epc>urn:epc:id:sgtin:0057000.123780.7788</epc>
      <epc>urn:epc:id:sgtin:0057000.123780.7789</epc>
      <epc>urn:epc:id:sgtin:0057000.123780.7790</epc>
   </epcList>

   <action>OBSERVE</action>

   <bizStep>urn:epcglobal:epcis:bizstep:fmcg:shipped</bizStep>

   <disposition>
      urn:epcglobal:epcis:disp:fmcg:available for sale
   </disposition>

   <readPoint>
      <id>urn:epc:id:gid:3721.810.105</id>
   </readPoint>

   <bizLocation>
      <id>urn:epcglobal:loc:0614141073467.A23-49</id>
   </bizLocation>

   <bizTransactionList>
      <bizTransaction type="urn:epcglobal:fmcg:btt:po">
          http://demo.fosstrak.com/po/q3432q4324
      </bizTransaction>
   </bizTransactionList>

</ObjectEvent>
```

Abbildung 2.1.2 ObjectEvent

2.1.2.1. Dimensionen von EPCIS-Events

Das ObjectEvent der Abbildung 2.1.2 gibt Auskunft über *Objekte*, deren *Datum und Uhrzeit*, deren *Lokation* und deren *Geschäftskontext*. Diese Angaben spiegeln gleichzeitig die Dimensionen des Events wieder. Jeder EPCIS-Ereignistyp besitzt vier Dimensionen, die angeben WAS, WANN, WO und WARUM passiert ist, siehe Abbildung 2.1.3. Die „*Was*“ Dimension ist abhängig vom Eventtyp. Zum Beispiel umfasst diese Dimension für ein ObjectEvent oder ein AggregationEvent eine oder mehrere EPCs und für ein TransactionEvent die *bizTransactionList*. Die Dimensionen, *Lokation* und *Geschäftskontext,* geben den rück- und den vorausblickenden Aspekt an.

Die Dimension der Lokation gibt nicht nur den Ort der Erfassung (*readPoint*) an, sondern ebenfalls den Ort, wo sich das Objekt nach der Erfassung (*bizLocation*) befindet. In der Praxis stellt beispielsweise ein *readPoint* das Lagereingangstor und eine *bizLocation* das Lager dar.

Die Dimension des Geschäftskontext umfasst den Prozessschritt (*bizStep*) und den Dispositionsschritt (*disposition*). Für das EPCIS-Event der Abbildung 2.1.2 bedeutet dies, dass die beobachteten Objekte versendet worden sind und momentan zum Verkauf stehen. Die möglichen Vorgabewerte für die Aspekte des Geschäftskontextes wurden innerhalb der EPCglobal-Gremien erarbeitet [GS106].

	Rückblickend	Vorausblickend
Was?	EPC, EPC-Klasse + Menge (Quantitätsereignis) Liste der Geschäftstransaktionen (Transaktionsereignis)	
Wann?	Zeit der Aufzeichnung des Ereignisses (ISO-Format)	
Wo?	Kennung des Lesepunktes	Kennung der Geschäftslokation
Warum?	Kennung des Prozessschritts	Kennung des Dispositionsschritts

Abbildung 2.1.3 Schlüsseldimensionen [GS106]

2.1.2.2. EPCIS-Eventtyp

Jeder EPCIS-Eventtyp besitzt Attribute, die diesen definieren. Die Abbildung 2.1.4 stellt das EPCIS-Event und dessen Subklassen, welche die vier Eventtypen repräsentieren, sowie die Beziehung zur BizTransaction-Klasse dar.

Jeder Eventtyp kann ein *bizTransactionList*-Element besitzen, welches mehrere *BizTransaction*-Elemente beinhalten kann. Laut der EPCIS-Spezifikation ist das Feld der *bizTransactionList* von ObjectEvents, AggregationEvents und QuantityEvents optional. Enthält dieses Feld für diese Eventtypen eine Beschreibung, so zeigt diese, dass das Event innerhalb des Kontextes einer oder mehrerer spezieller Geschäftstransaktionen stattfindet. Bei TransactionEvents ist das *bizTransactionList* Feld nicht optional [EPC07], [BEA06].

In der Tabelle 2.1.2 sind die Attribute der fünf Eventklassen aufgelistet und näher erläutert.

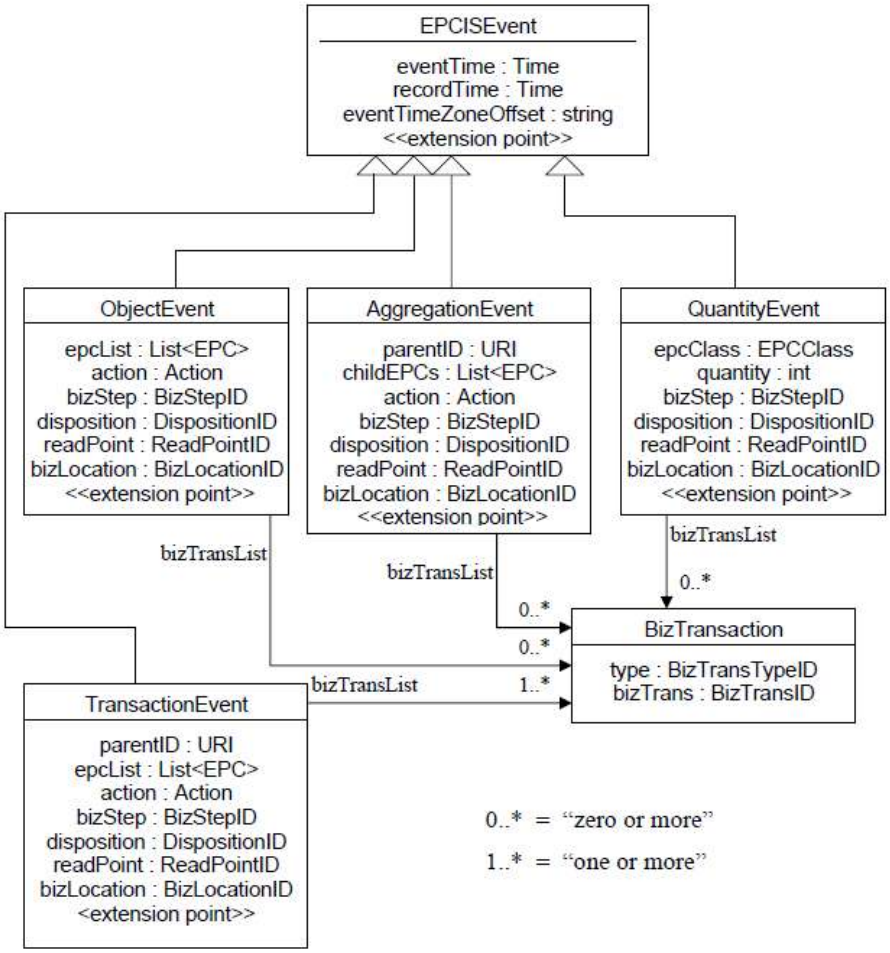

Abbildung 2.1.4 UML-Diagramm der Eventtypen [EPC07]

Attribut	**Beschreibung**	**O**	**A**	**Q**	**T**
eventTime	Datum und Uhrzeit des Ereignisses gemäß der Applikation zur EPCIS-Datenerfassung	R	R	R	R
recordTime	Datum und Uhrzeit, zu der das Ereignis im EPCIS-Repository gespeichert wird.	O	O	O	O
EventTimeZoneOffset	Gleicht die Verschiebung der Zeitzonen aus.	R	R	R	R
epcList	Eine Liste der EPCs der physischen Objekte, die zu diesem Ereignis gehören. Beim Transaktionsereignisses (Transaction Event) sind diese Objekte mit der Transaktion verknüpft.	R			R
parentID	Ein Ident der oberen Hierarchiestufe, bezogen auf die Liste der EPCs. (Der Status ist abhängig vom Eventtyp und der Nutzung des Attributes „action“.)		O		O
epcClass	Ein Ident, das die Objektklasse spezifiziert, welche zu dem Ereignis gehört.			R	
action	Diese Attribut beschreibt die Beziehung des Ereignisses, bezogen auf den Lebenslauf der enthaltenen EPCs. Das Attribut kann die Werte *Add*, *Observe* oder *Delete* annehmen.	R	R		R

Attribut	Beschreibung	O	A	Q	T
childEPCs	Die Klasse in der Produkte zusammengefasst werden.		R		
quantity	Die Anzahl von Objekte innerhalb einer Objektklasse, die das QuantityEvent beschreibt.			R	
bizStep	Der Prozessschritt, der dieses Ereignis beinhaltete.	O	O	O	O
disposition	Beschreibt den Zustand der zugehörigen Objekte, der bis zum Eintreten eines Folgeereignisses eingehalten wird.	O	O	O	O
readPoint	Der Lesepunkt, an dem das Ereignis stattgefunden hat.	O	O	O	O
bizLocation	Die Lokation, in der sich die zu den EPC zugehörigen Objekte nach dem Auslesen befinden.	O	O	O	O
type	Ein Ident, das anzeigt, welche Art von Transaktion stattgefunden hat.	O	O	O	O
bizTransaction	Ein Ident, das die spezifische Transaktion kennzeichnet.	O	O	O	R

Tabelle 2.1.2 EPCIS-Event-Attribute und deren Beschreibung nach [GS106][1]

ObjectEvent

Dieses Ereignis beinhaltet Informationen, die sich auf ein oder mehrere EPCs beziehen. Es dient zur Beobachtung von Objekten. In der Regel werden solche Events an readPoints erstellt, wenn Objekte registriert werden. Ein readPoint, der beispielsweise zwischen Wareneingang und Lager positioniert ist, nimmt ein Event auf, wenn ein Objekt vom Wareneingang ins Lager verschoben wird [GS106].

AggregationEvent

Das AggregationEvent wird verwendet, wenn einzelne Objekt zusammengefasst worden sind. Das ist dann der Fall, wenn es eine Anzahl von Objekten gibt, die in einer „übergeordneten“ Einheit aggregiert worden sind. Dieser Eventtyp wird beispielsweise angewendet, um Einzelprodukte in Kartons zu verpacken oder Kartons auf Paletten zu verladen, die im Weiteren als eine Einheit betrachtet werden. Ein Aggregationstyp wird nicht für „schwache“ Aggregations-beziehungen genutzt. Daraus folgt, das dieses EPCIS-Event nicht eingesetzt wird, um zwei Paletten, die zu einer Lieferung gehören, zusammen zu fassen. In diesem Fall wird das Transaktionsereignis verwendet [GS106].

QuantityEvent

Dieser Typ registriert Mengen bezogen auf eine Objektklasse und enthält kein *epcList*-Element. Das QuantityEvent wird im Bereich der permanenten Inventur oder Ermittlung von Abverkaufszahlen angewendet [GS106].

TransactionEvent

Dieser Eventtyp beschreibt die Verknüpfung (oder Trennung) von Objekten mit einer oder mehreren Transaktionen. Zum Beispiel könnte in einem solchen Ereignis die Referenz auf eine bestimmte Lieferung mit den darin angekündigten EPC-Objekten verknüpft werden [GS106].

1) O=optionales Feld; R= Pflichtfeld

2.2. Zugriffskontrolle

Die Zugriffskontrolle (auch: Rechteprüfung, Berechtigungsüberprüfung, Autorisierung) kontrolliert den Zugriff von Prinzipalen[1)] auf Ressourcen bzw. Objekte. Für die Realisierung der Zugriffskontrolle sind weitere Sicherheitsmechanismen, wie die Zugangskontrolle oder die Rechteverwaltung erforderlich. So muss die Zugangskontrolle (Authentifizierung), über die die Identität eines Prinzipals geprüft wird, vor der Zugriffskontrolle erfolgen. Die Administration der Zugriffsrechte innerhalb eines Systems, ist die Aufgabe der Rechteverwaltung. Sie vergibt und ändert die Zugriffsrechte für jedes Prinzipal auf zugriffsgeschützte Ressourcen bzw. Objekte und legt die Umstände fest, unter denen die Rechte wahrgenommen werden dürfen (zum Beispiel zu bestimmten Zeiten). Die Verwaltung von Rechten kann unter dem Vollständigkeitsprinzip (*complete mediation)* erfolgen. Das bedeutet, dass alle Subjekte und Objekte verwaltet werden. Diese Verwaltungsmethode ist sehr umfangreich. Eine effizientere und häufig verwendete Methode ist das Erlaubnisprinzip. Bei diesem Prinzip sind alle Aktionen verboten, die nicht explizit erlaubt worden sind. Der komplementäre Ansatz zu diesem, ist das Verbotsprinzip. Bei diesem sind alle Aktionen erlaubt, die nicht explizit verboten wurden [Leh07].

Weiterführende Informationen zu Authentifizierungs- und Autorisierungsprinzipen in verteilten und nicht verteilten Systemen werden in [Sch08] vorgestellt und bewertet. In [GS07] wurden Anforderungen an Zugriffskontrollsysteme in globalen Traceability-Networks und im Speziellen in Auto-ID-Repositories, sowie deren entsprechende Umsetzungen diskutiert (siehe Kapitel 2.3.).

Im Rahmen dieses Abschnittes wird die Policysprache XACML und deren Erweiterung durch das Einbetten der Ressourcenadressierung EAL zu aidXACML präsentiert.

2.2.1. eXtensible Access Control Markup Language

XACML (eXtensible Access Control Markup Language) [Mos05] ist eine OASIS-Spezifikation, die die XML-basierte Berechtigungspolicy und das request/ response-Format standardisiert, sowie eine Referenzarchitektur definiert.

2.2.1.1. Policy-Language Modell

Das Policy Language Model ist in der Abbildung 8.1.1 dargestellt. Die Hauptelemente der XACML-Policysprache zur Definition von Berechtigungen sind *PolicySet*, *Policy* und *Rule.* Die Abbildung 2.2.2 zeigt die hierarchische Struktur, die zwischen diesen Elementen des Policy-Modells existiert. Jedes der Hauptelemente enthält das *Target*-Element, das eine Sammlung von *Subjects, Resources, Actions* und *Environments* spezifiziert. Die Entscheidung, ob ein PolicySet, eine Policy bzw. eine Rule zutrifft, ist von dessen Target-Element abgängig. Ein PolicySet kann aus mehreren PolicySets oder Policies bestehen. Eine Policy enthält mindestens eine Rule. Eine anwendbare Rule liefert entweder Permit oder Deny als Ergebnis. Da mehrere Hauptelemente innerhalb des Policy-Modell möglich sind, ist die Referenzierung von Combining-Algorithmen in den Elementen PolicySet und Policy notwendig, für die Ermittlung eines Ergebnisses. Für anwendbare PolicySets und Policies liefert ein Combining-

1) Eine Entität,der eine eindeutige Indentität über die Zugangs- und Zugriffsberechtigungen zugeordnet werden können. Es kann sich um ein Subjekt, eine Person, eine Rolle, ein Terminal, eine Smartcard oder einen Prozess handeln.

Algorithmus entweder Permit, Deny, Indeterminate oder NotApplicable als Ergebnis [GS07].

Die XML-Struktur der Policy wird in Abbildung 2.2.1 anhand eines konkreten Beispiels verdeutlicht. Die abgebildete Policy ist eine Policy der Firma Medi Corp, die mit ihrem Domänennamen *medi.example.com* identifiziert wird. Die Firma Medi Corp legt in ihrer Zugriffspolitik fest, das Nutzer, deren E-Mailadresse den Namensraum *medi.example.com* beinhalten, jeglicher Zugriff auf alle Ressourcen erlaubt ist. Die zugehörige XACML-Policy besteht aus einer Headerinformation, einer optionalen Beschreibung der Policy, einer Gültigkeitsdefinition und einer Regel. Als Combining-Algorithmus ist der Rule-Combining-Algorithmus, d*eny-overrides,* angegeben. Er gibt nur Permit als Policy-Ergebnis zurück, wenn alle Regeln der Policy mit Permit bewertet wurden. Andere Algorithmen, die zur Verfügung stehen sind: *permit-overrides, first-applicable und only-one-applicable.* Weiterführende Informationen zu den Combining-Algorithmen und die detailierte Erläuterung zu diesem und anderen Beispielen sind in [Mos05] zu finden.

```
<?xml version="1.0" encoding="UTF-8"?>

<Policy
xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-
os.xsd"
PolicyId="urn:oasis:names:tc:example:SimplePolicy1"
RuleCombiningAlgId="identifier:rule-combining-algorithm:deny-overrides">
 <Description>
  Medi Corp access control policy
 </Description>
 <Target/>
 <Rule>
 RuleId= "urn:oasis:names:tc:xacml:2.0:example:SimpleRule1"
 Effect="Permit">
  <Description>
   Any subject with an e-mail name in the med.example.com
   domain can perform any action on any resource.
  </Description>
  <Target>
   <Subjects>
    <Subject>
     <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:
     function:rfc822Name-match">
      <AttributeValue
      DataType="http://www.w3.org/2001/XMLSchema#string">
       med.example.com
      </AttributeValue>
      <SubjectAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject
      -id" DataType="urn:oasis:names:tc:xacml:1.0:data-
      type:rfc822Name"/>
     </SubjectMatch>
    </Subject>
   </Subjects>
  </Target>
 </Rule>
</Policy>
```

Abbildung 2.2.1 Beispiel einer Policy [Mos05]

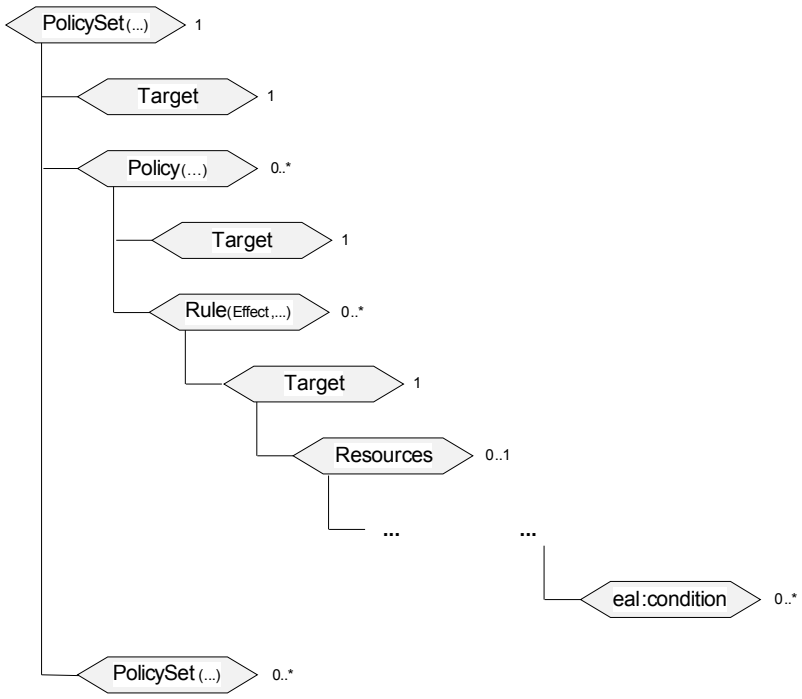

Abbildung 2.2.2 Hierarchie des PolicySet-Modells nach [Mos05]

2.2.1.2. Datenfluss- Modell

In der XACML-Spezifikation wird folgende Architektur (siehe Abbildung 2.2.3) für die Entscheidung und die Durchsetzung von Zugriffsrechten unter Verwendung von XACML vorgestellt.

Im Wesentlichen wird zwischen Policy Decision Point (PDP) und Policy Enforcement Point (PEP) unterschieden. Der PEP ist eine domänenspezifische Komponente und setzt Zugriffsentscheidungen auf Ressourcen durch, die von dem zuständigen domänenunabhängigen PDP anhand der Auswertung der anwendbaren XACML-Berechtigungsdefinition generiert worden sind. Einige der in Abbildung 2.2.3 dargestellten Datenflüsse werden durch die Verwendung eines Repositories erleichtert, wie zum Bespiel die Kommunikation zwischen Kontexthandler und PIP oder die Kommunikation zwischen PDP und PAP.

Im diesem Beispiel nimmt der PEP eine Zugriffsanfrage vom Client entgegen. Diese Anfrage wird über den Kontexthandler, der die Anfrage in den XACML-Request-Kontext überführt, an den zuständigen PDP gesendet. Der PDP fragt jedes Attribut beim Kontexthandler ab. Der Handler gibt dieses Request an den PIP weiter und kann optional die Ressourcen in die Abfrage einbeziehen. Er sendet die erbetenen Attribute und optional die Ressourcen an den PDP zurück. Basierend auf den Policies entscheidet der PDP und er generiert eine entsprechende XACML-Response mit der Autorisierungsentscheidung "Permit", "Deny", "Indeterminate" oder "NotApplicable". Diese Antwort sendet er zum PEP über den Kontexthandler, der die Umwandlung des XACML-Response in das naive Format des PEP vornimmt und weiter sendet. Entsprechend der PDP-Entscheidung über den Zugriff, wird diese vom PEP durchgesetzt. Ist die PEP-Anfrage nicht durch den PDP entscheidbar, signalisiert er dies dem PEP durch die Antwort "Indeterminate" oder "NotApplicable". Der PEP kann, infolge dieser Response, den Zugriff auf die Ressource verweigern [Mos05].

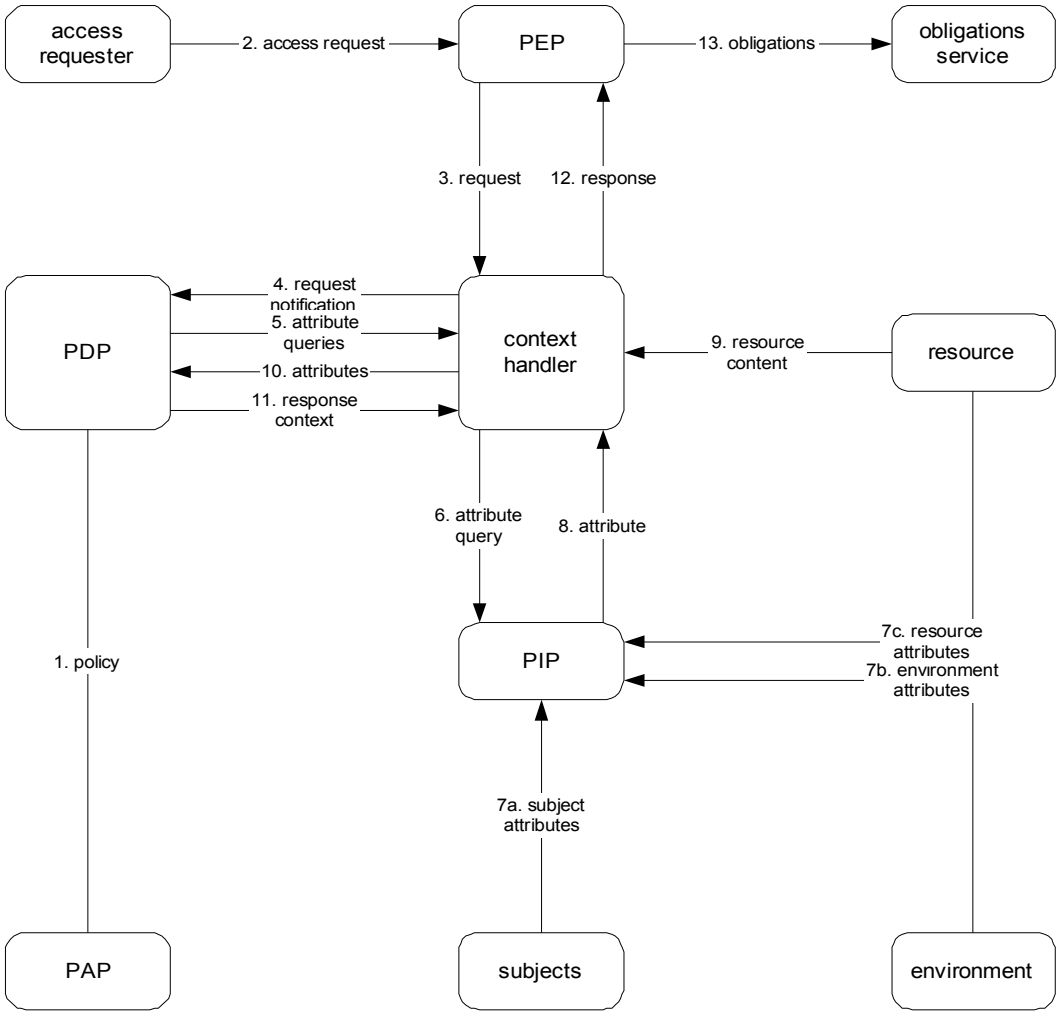

Abbildung 2.2.3 Datenflussdiagramm [Mos05]

2.2.1.3. Zusammenfassung

Der XACML-Standard ist eine umfassende und wichtige Spezifikation im Bereich der Zugriffskontrollsysteme. Er umfasst eine XML-Policy-Sprache und eine Auswertungslogik für Policies und PolicySets, sowie das XML-Request-Response-Format zwischen PEP und PDP, als auch eine Referenzarchitektur. Negativ ist zu bewerten, dass die Autorisierungsentscheidung des PDP nur "Permit" oder "Deny" lauten kann. Eine Anfrage teilweise zu erlauben ist, entsprechend der vorliegenden Berechtigungen, nicht vorgesehen.

So würde zum Beispiel eine Anfrage auf die EPCs "urn:epc:id:gid:100.[**200-300**].*" komplett abgelehnt werden, wenn die Berechtigung zu dieser Anfrage nur für urn:epc:id:gid:100.[**250-400**].* bestünde. Stattdessen sollten die Events mit urn:epc:id:gid:100.[**250-300**].* zurück zuliefert werden. Des Weiteren ist die verspätete Genehmigung von Anfragen, beispielsweise nach der manuellen Prüfung des Zugriffswunsches durch den Administrator, nicht vorgesehen. Auch die Delegation von Entscheidungskompetenzen wird nicht explizit unterstützt [GS07].

2.2.2. Auto-ID-XACML

Auf Basis von XACML wurde die XML-Sprache aidXACML, deren wesentlicher Bestandteil die neue Ressourcenadressierungssprache EAL (Event Addressing Language) ist, entwickelt [GS07]. Mit der Einbettung von EAL in das XACML-Resource-Element wird die attributgenaue und regelbasierte Adressierung von Ereignismengen möglich. Weiterhin werden die XACML-Schwachstellen, der teilweisen Genehmigung von Anfragen, sowie das Einbeziehen von externen Zugriffsentscheidungen, durch aidXACML adressiert.

EAL ist eine Sprache zur regelbasierten und feingranularen Adressierung von Ereignismengen. Des Weiteren definiert EAL Operatoren, mit deren Hilfe Zugriffs- und Delegationsentscheidungen getroffen werden können. Die EAL-Operatoren bilden die Basis für neue Matching- und Combining-Algorithmen in aidXACML. Eine effiziente Durchsetzung von Zugriffsregeln kann durch die Übersetzung von EAL in SQL und XQuery realisiert werden.

2.2.2.1. EAL-Instanzen

Die EAL-Instanz beschreibt eine Ergebnismenge, die durch eine oder mehrere eventSubsets oder durch ein permittedEvents bzw. ein deniedEvents spezifiziert ist. Die Abbildung 2.2.4 zeigt den strukturellen Aufbau einer EAL-Instanz grafisch.

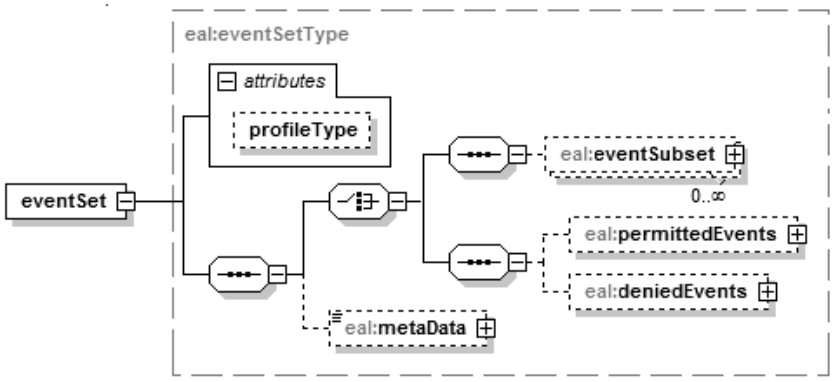

Abbildung 2.2.4 Schema EAL-Instanz [EAL08a]

Ein eventSubset, in Abbildung 2.2.5 dargestellt, benennt die Liste der sichtbaren Attribute. Die Basismenge, auf die sich das eventSubset bezieht, ist im Attribut basicSet beinhaltet. Die Attribute id und description können optional angegeben werden. Jedes eventSubset kann aus einem conditionSets-Element, sowie einem attributeSubsets-Element, bestehen. Ein conditionSets-Element kann mehrere conditionSet-Elemente enthalten, die konjunktiv verknüpftet werden. Mehrere AttributeSubset-Elemente können in einem AttributeSubsets-Element enthalten sein. Sie werden ebenfalls konjunktiv verknüpft.

Ein conditionSet, in Abbildung 2.2.6 dargestellt, verwendet die Elemente *condition* und *predicate* zur Beschränkung des im conditionSet angegebenen Attributes, das in einen bestimmten Datentyp vorliegt. Das conditionSet bezieht sich nur genau auf ein Attribut. Diese Einschränkung wurde getroffen, um EAL-Instanzen ohne Redundanzen logisch miteinander verknüpfen zu können. Mehrere c*ondition*-Elemente sind disjunktiv zu verknüpfen. Jede condition beinhaltet ein oder mehrere *predicate*-Elemente. Die Verknüpfung von

Prädikaten erfolgt konjunktiv. Ein Prädikat definiert die Beschränkung durch den Operator des Prädikates und den Operanden [GS07]. Mehrere Operanden werden mittels *literal*-Elementen innerhalb des Prädikates angegeben. Durch das *apply*-Element können arithmetische Operatoren, sowie der XPath-Operator in EAL auf weitere *apply*-Elemente bzw. Operanden im *literal*-Element angewendet werden [EAL08a].

Ein attributSubset, in Abbildung 2.2.7, entspricht der Struktur des eventSubsets. Es beinhaltet die Basismenge, die im basicSet-Attribut spezifiziert wird und die Liste der sichtbaren Attribute. Das attributSubset kann optional mit einer ID bezeichnet werden. Die Basismenge kann, wie beim eventSubset, durch das conditionSets-Element bzw. das attributeSubsets-Element beschränkt werden.

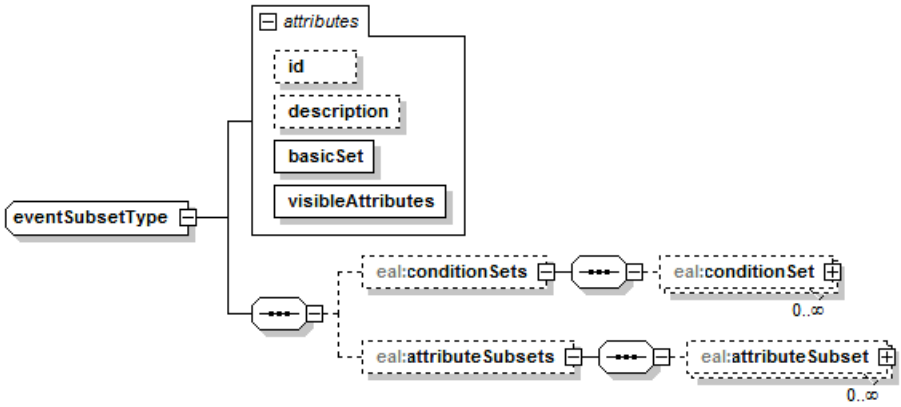

Abbildung 2.2.5 EAL-eventSubset [EAL08a]

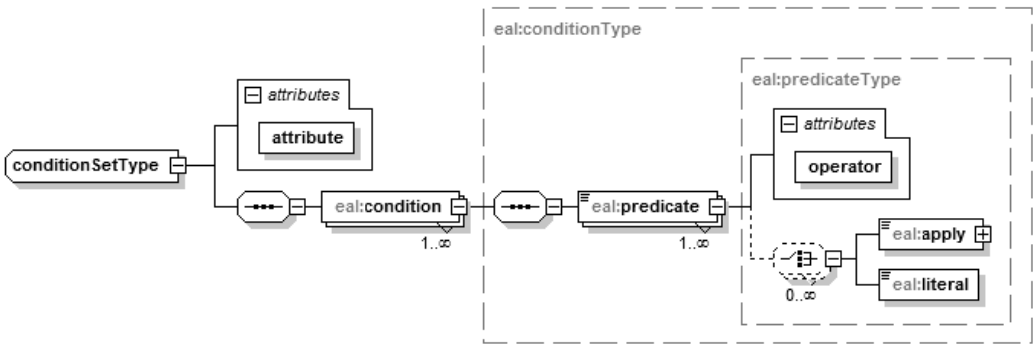

Abbildung 2.2.6 EAL-conditionSet [EAL08a]

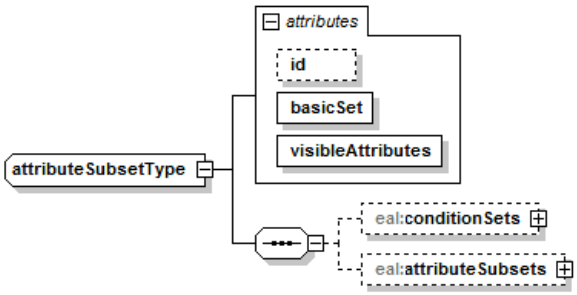

Abbildung 2.2.7 EAL-attributeSubset [EAL08a]

Abbildung 2.2.8 zeigt eine beispielhafte EAL-Instanz, die die Menge aller *ObjectEvents, deren readPoint*-Element den Wert *urn:epc:id:gid:100.200.300* besitzt und deren *epc*-Element einem der Parameter urn:epc:id:sgtin: 0057000.123780.7788 oder urn:epc:id:sgtin:0057000.123780.7789 enthält, als Ergebnis liefert. In der Ergebnismenge sind alle Elemente des *ObjectEvents* sichtbar, dies wird durch den Wert "*" des *visibleAttributes*-Attributes des eventSubSet-Elementes definiert.

```
<eal:eventSet>

   <eal:eventSubset visibleAttributes="*" basicSet="ObjectEvent">

      <eal:conditionSets>
         <eal:conditionSet attribute="readPoint">
            <eal:condition>
               <eal:predicate Operator="epc-match">
                  urn:epc:id:gid:100.200.300
               </eal:predicate>
            </eal:condition>
         </eal:conditionSet>
      </eal:conditionSets>

      <eal:conditionSets>
         <eal:conditionSet attribute="epcList.epc">
            <eal:condition>
               <eal:predicate operator=”epc-match”>
                  urn:epc:id:sgtin:0057000.123780.7788
               </eal:predicate>
            </eal:condition>

            <eal:condition>
               <eal:predicate operator=”epc-match”>
                  urn:epc:id:sgtin:0057000.123780.7789
               </eal:predicate>
         </eal:conditionSet>
      </eal:conditionSets>
   </eal:eventSubset>

</eal:EventSet>
```

Abbildung 2.2.8 Beispiel einer EAL-Instanz

2.2.2.2. aidXACML-Architektur

Das aidXACML-Konzept stellt eine Erweiterung des XACML-Frameworkes dar. Die wesentlichen konzeptionellen Bestandteile der aidXACML-Architektur werden in der Abbildung 2.2.9 gezeigt. Das Auto-ID-Repository repräsentiert den Policy Enforcement Point (PEP) und implementiert beispielsweise das EPCIS Query Control Interface, über das Events abgerufen werden können. Über Webservice-Schnittstellen werden PEP und PDP angesprochen.

Die weitere Vorstellung der aidXACML-Architektur erfolgt im Kontext des Abfrageverlaufes. Der Referenzmonitor formt die vom Client gestellte poll-Zugriffsanfrage (1) in das aidXACML-Request-Format um, in dem er die Anfrageparameter in EAL übersetzt und in aidXACML-Request einbettet. Anschließend sendet er den aidXACML-Request als Parameter einer Webservice-Operation an den PDP (2). Dieser wertet jede erhaltene Anfrage gegen die EAL-Definition der lokalen aidXACML-Policies, mittels Matching-Algorithmus, aus. Des Weiteren wird vom PDP die EAL-basierte Delegationsregel ausgewertet. Liegt eine Delegation vor, sendet der lokale PDP

den aidXACML-Request mit *delegieren*-EAL an externe PDPs (3), diese teilen ihm ihre aidXACML-Response Ergebnisse mit (4). Der lokale PDP kombiniert die erhaltenen aidXACML-Response Ergebnisse und das lokale Ergebnis per Combining-Algorithmus. Das kombinierte aidXACML-Response-Ergebnis wird vom PDP an den PEP zurückgesendet (5). Aus der aidXACML-Response des PDP wird die EAL-Definition extrahiert und in ein SQL-Statement oder XQuery-Statement übersetzt, um so die Anfrage auf das Repository auszuführen. Die Tupel der Datenbankanfrage werden als Response dem Client übermittelt (6) [GS07].

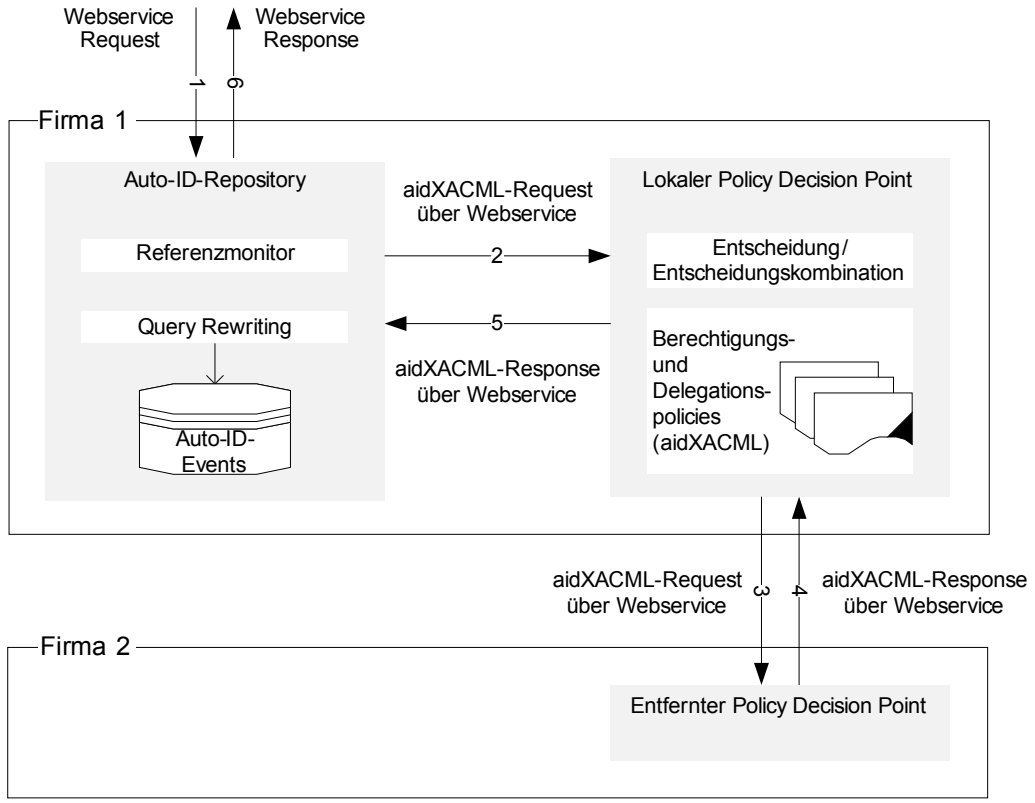

Abbildung 2.2.9 Ablauf einer Abfrage in aidXACML-Architektur nach [GS07]

2.2.2.3. Zusammenfassung

Die Erweiterung von XACML, mit Hilfe der neu entwickelten Ressoucen-adressierungssprache EAL, adressiert die Schwachstellen von XACML. Mit der Einbettung von EAL in das XACML-Resource-Element wird die attributgenaue und regelbasierte Adressierung von Ereignismengen, die teilweise Genehmigung von Anfragen, sowie das Einbeziehen von externen Zugriffsentscheidungen durch aidXACML möglich.

Durch die damit geschaffene Flexibilität, ist die Vergabe von Rechten innerhalb einer globalen Wertschöpfungskette technisch sehr umständlich geworden. Für jedes Objekt und jedes aggregierte Objekt dieser Wertschöpfungskette können unter Berücksichtigung der Unternehmensgrundsätze feingranulare Rechte für jeden Teilnehmer des EPCglobal-Networks gewährt werden. Dieses Berechtigungsvergabe erfordert einen hohen administrativen Aufwand, der durch Automatisierung reduziert werden soll.

2.3. Zugriffskontrollsystem im EPCISglobal-Network

In [Sch08] wurde ein Prototyp entwickelt, der ein verteiltes Zugriffskontrollsystem (Distributed Decision Service) auf Basis von aidXACML realisiert. Der Prototyp ist als EPCglobal-konformes EPCIS einsetzbar und nutzt ein eigenes EPCIS-konformes Event-Repository. Die EPCIS-Systeme kommunizieren untereinander über SOAP-Webservices und benutzen aidXACML als Policysprache für die Zugriffskontrolle. Die aidXACML-Policies können über mehrere Systeme verteilt sein, so dass eine Zugriffsentscheidung sich aus mehreren Teilentscheidungen der einzelnen System zusammensetzen kann. Durch Delegation von Zugriffsentscheidungen ist es Teilnehmern des EPCISglobal-Networks möglich, Einfluss auf die Zugriffsentscheidungen zu nehmen. So können beispielsweise Lieferbeziehungen geheim gehalten werden, um in Wettbewerb stehenden Teilnehmern nicht zu viel Einblick ins Unternehmensmanagement zu gewähren. Im Folgenden wird auf die Delegation von Zugriffsentscheidung nicht weiter eingegangen. Der Fokus wird auf die Struktur des Prototypen, die Realisierung der Durchsetzung von Zugriffsentscheidungen und das SQL-Query-Rewriting gelegt.

2.3.1. Struktur des Prototypen

Die Grundstruktur des Prototypen ist in Abbildung 2.3.1 dargestellt. Diese zeigt die Programmpakete und deren Beziehungen. Der EPC Information Service (EPCIS) und der Distibuted Decision Service (DDS) sind Anwendungen des Prototypen.

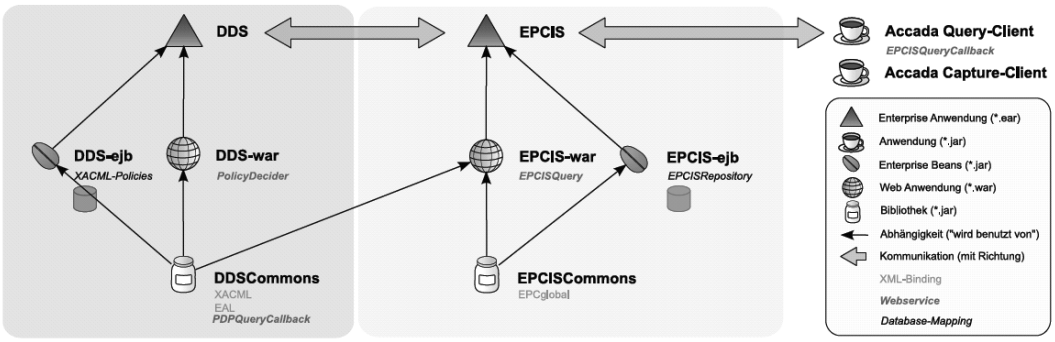

Abbildung 2.3.1 Teilobjekte und deren gegenseitige Abhängigkeit [Sch08]

Das EPCIS umfasst die Implementierung des Query Control Interface und des EPCIS Capture Interface. Das Query Control Interface befindet sich im Webservice *EPCglobalEPCISService* des Projekts EPCIS-war. Die Anfragenbearbeitung erfolgt über die EPCISQueryBean. Mittels eines Servlet ist das EPCIS Capture Interface im Projekt EPCIS-war implementiert. Dieses Servlet nimmt XML-Dokumente vom Typ EPCglobal-EPCIS Document entgegen und entspricht in der Funktionsweise dem Capture Interface von Accada. Der Policy Enforcement Point (PEP) schützt die Ereignisdaten des Event-Repository und ist im EPCIS enthalten.

Der Distributed Decision Service ist die Implementierung eines Policy Decision Points (PDP). Er beinhaltet die Komponenten der Policy-Verwaltung, wie dem Policy Administration Point (PAP). Der PDP bezieht seine Policies aus einer XML-Datei, die die XACML-PolicySets enthalten. Sie wird beim Start der

Anwendung einmalig geladen, die Policy-Änderungen sind erst nach einem Neustart des gesamten Systems verfügbar.

Beide Anwendungen, EPC Information Service und Distibuted Decision Service verwenden den gleichen Webservice, um die Kommunikation zwischen Policy Enforcement Point (PEP) und Policy Decision Point (PDP) zu realisieren. Dieser Webservice implementieren gleiche Interfaces aus dem gemeinsamen Projekt DDSCommons.

2.3.2. Durchsetzung der Zugriffsberechtigungen

Die Durchsetzung der Zugriffsberechtigungen für den Prototypen erfolgt mittels EAL Query Translation. Ein SimpleEventQuery, welches am Query Control Interface des EPCIS-System eintrifft, wird in die interne Ereignisadressierungssprache EAL übersetzt. Diese Übersetzung wird durch den Session-Bean EPCISQueryBean, der sich im Programmmodul EPCIS-ejb befindet, vorgenommen. Eine weitere Übersetzung, die EAL Query Translation, findet im letzten Schritt der Anfragebearbeitung im Policy Enforcement Point statt. Die Anfrage im EAL-Format wird in eine domänenspezifische Anfragesprache umgewandet. Im Prototyp wird das EAL-Dokument in EQL (EJB Query Language) überführt, da die Datenbank des Prototypen über EJBs angesprochen wird. Die Abbildung 2.3.2 zeigt das prinzipielle Vorgehen der Übersetzung, die im RepositoryQuery-Bean stattfindet [Sch08].

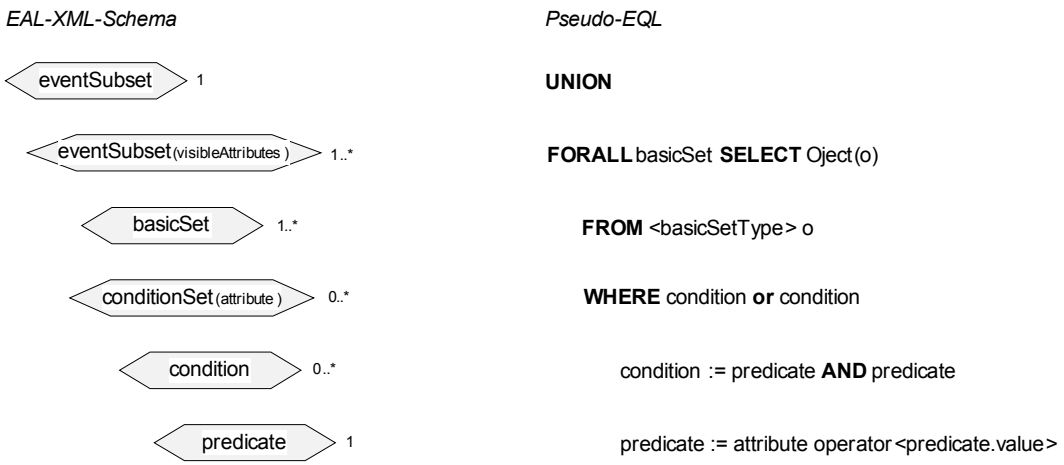

Abbildung 2.3.2 Übersetzung von EAL in EQL nach [Sch08]

2.3.3. Zusammenfassung

Der Prototyp verwendet, im Rahmen der Zugriffskontrolle, die Berechtigungssprache aidXACML. Alle Berechtigungen des Prototypen sind in einem globalen PolicySet definiert, welches in einem XML-Dokument im Prototyp gespeichert ist. Die Formulierung und die Verwaltung der Berechtigungen erfolgt durch den Systemadministrator manuell. Eine automatische Berechtigungsformulierung, die den administrativen Aufwand vermindert und potentiell vereinfacht, ist nicht vorhanden. Die Durchsetzung der Zugriffsberechtigung erfolgt mittels EAL Query Translation.

2.4. Weitere Ansätze

In diesem Abschnitt werden weitere Ansätze zur Zugriffskontrolle und zur Transformation von XML-Dokumenten vorgestellt. Anschließend werden diese Ansätze im Bezug auf die Umsetzung der geschäftstransaktionsbasierte Formulierung von Zugriffsberechtigungen auf EPCIS-Repositories bewertet.

2.4.1. Event Based Access Control

In konventionellen Zugriffskontrollsystemen werden Zugriffskontroll-Policies ausgewertet, nachdem ein Nutzer eine Zugriffsanfrage stellt. Im Bereich des E-Commerce ist dies ein ernster Nachteil, wenn Zugriffskontroll-Policies auf komplexen Entscheidungsfindungen, die sich beispielsweise auf Nutzeranalysen beziehen, basieren. Ein Beispiel für eine solche Policy ist: *„A gold user is allowed to access the stock analysis data for the last six months, if the user has done transactions worth $10,000/- in the last two months or if the cumulative sum of the transactions done by all the customers referred by the gold user is more than $ 50,000/-" [BGMP03].* Die Response-Time für Zugriffsanfragen dieser Art verlängert sich somit um die Ausführungszeit, die zur Policy-Bewertung benötigt wird. Dies kann zu entscheidenden Verzögerungen führen. Eine Lösung für diese Problematik stellt der Ansatz zur eventbasierten Zugriffskontrolle dar, der die dynamische Bewertung von komplexen Policies umfasst. Bei der eventbasierten Zugriffskontrolle werden die Zugriffsprivilegien dynamisch, in Abhängigkeit von bestimmten Datenbanken und zeitlichen Events, geändert. Dieser Ansatz wird im E-Commerce und in Anwendungen verwendet, in denen die Bewertung des Zugriffes von internen und externen Bedingungen abhängig und umfangreich ist [BGMP03].

Dieser Abschnitt stellt die EBAC-Architektur vor, die ein dynamisches Zugriffs-kontroll-Framework basierend auf Ereignissen realisiert und eine spezielle Event-Driven Architektur darstellt. Im Allgemeinen wird daher in diesem Abschnitt vorerst auf die Event-Driven Architektur eingegangen, und danach erfolgt die Erläuterung zur EBAC-Architektur.

2.4.1.1. Event-Driven Architektur

Eine ereignisgesteuerte Architektur, engl. *Event-Driven Architecture* und kurz EDA, ist eine Systemarchitektur, in der die Architektur-Systemkomponenten durch interne oder externe Ereignisse gesteuert werden. Die Reaktion des Systems auf Ereignisse ist abhängig von der Ereignisbehandlung *(event handling).* Ein Ereignis kann im System an einer oder mehreren Stellen Aktionen auslösen oder reaktionslos vom System verworfen werden.

Eine ereignisgesteuerte Architektur basiert auf den logischen Ebenen: Event Generator, Event Channel, Event Processing Engine und Downstream Event-Drive Activity. Im Weiteren findet eine Unterscheidung zwischen den Prozessen: Simple Event Processing, Event Stream Processing und Complex Event Processing statt. Das *Simple Event Processing* löst direkt spezifische und messbare Ereignisse aus. Wobei beim *Complex Event Processing* mehrere Ereignisse, die ursächlich, räumlich und zeitlich in Zusammenhang stehen, in einem gemeinsamen Prozess zusammengefasst werden. Es ist eine Struktur notwendig, die das entscheidende Event aus einer Sammlung identifiziert. Ein Beispiel für die Anwendung dieses Prozesses ist die Sperrung des E-Mail

Accountes eines Nutzers, nach dem die Authentifizierung des Nutzers dreimal fehlgeschlagen ist. Im *Event Stream Processing* finden einfache und bedeutende Events statt. Dieser Prozess wird eingesetzt, um zeitnah Informationen und Entscheidungen herbeizuführen, die für die Bearbeitung des Ereignisses benötigt werden [Mic06], [EDA08].

2.4.1.2. EBAC Architektur

Die Abbildung 2.4.1 zeigt die Systemarchitektur der eventbasierten Zugriffskontrolle. Das System besitzt ein User Interface, um Zugriffskontroll-Policies zu definieren, sowie auf Daten von der zugrunde liegenden Datenbank zu zu greifen. Der Policy Translator übersetzt die spezifizierten Policies in die XML-basierte Policysprache. Der External Event Handling Broker übernimmt die Registrierung und die Generierung des XML-Message Schemas. Mittels XML-Messages können entfernte Event-Detektoren über das Auftreten von Events im Bereich dieser Detektoren informieren. Der External Event Handling Broker stellt eine Schnittstelle zum Messaging System bereit, um Nachrichten von extern auftretenden Events übermitteln zu können. Die Access Validation Engine prüft die Zugriffsanfrage des Nutzers gegen dessen Zugriffskontrollprivilegien. Die Policy Execution Engine überwacht das Auftreten von Events, die in den Zugriffspolicies definiert wurden und ändert die Zugriffsprivilegien des Nutzers dynamisch. Sie wertet die Bedingungen aus, die nach dem Eventauftreten festgestellt wurden. Wenn die Bedingung mit wahr bewertet wurde, wird die Zugriffskontrollaktion in der Policy-Datenbank durchgesetzt. Die Policy-Datenbank speichert die Zugriffspriviliegen. Die Implementierung der Policy Execution Engine erfolgt durch Servlets. Das User Interface wurde unter Verwendung von JSP entwickelt. Dies ermöglicht eine systemexterne Administration, weil der Policy Maker die Policy über das Web definieren kann [BGMP03].

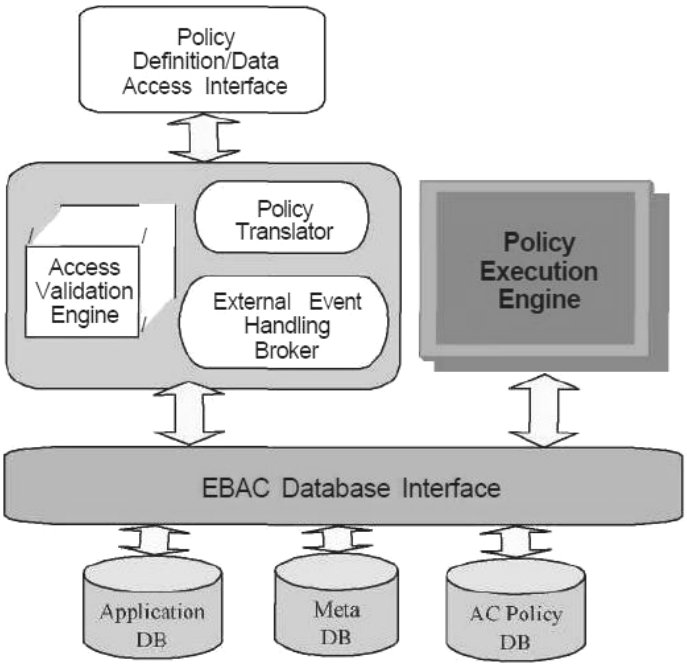

Abbildung 2.4.1 Architektur des eventbasierten Zugriffskontrollsystems [BGMP03]

2.4.2. Derived Access Control Specification for XML

In diesem Abschnitt wird ein XQuery-basierter Ansatz zur Herleitung von Zugriffskontrollregeln, beispielsweise für Dokumente und Datenbankinhalte, präsentiert. Dieser Ansatz umfasst drei Teile:

- Verwendung von XQuery zur Rückgabe von Privilegien, der Form <*subject, object, access-right*>
- Bereitstellung eines Prädikates *access(subjekt,objekt)*, das die Zugriffsrechte des Subjektes auf das Objekt zurückgibt
- Bereitstellung einer Funktionalität, um rekursiv den Zugriff, basierend auf anderen ableitbaren Privilegien, herzuleiten

Derived Access Control Spezifikation für XML (DACX) erreicht durch den Einsatz von XQuery eine neue Flexibilität. Die Formulierung von allgemeinen Expressions ist damit möglich, um Privilegien zu definieren. In anderen XML-basierten Ansätzen ist die Policy-Definition entweder für jedes Dokument einzeln oder im Hinblick auf das Schema vorzunehmen. Bei der Definition von Policy auf Schemaebene werden Privilegien auf Teile eines Dokumentes vergeben. Diese Policy-Definitionen spezifizieren keine kontextsensitive Beschränkungen auf bestimmte Dokumente oder Teildokumente. Der XQuery-basierter Ansatz hingegen unterstützt diese Fähigkeit. Er erlaubt eine Definition von Privilegien basierend auf dem Kontext von Dokumenten. Im Bereich der feingranularen XML-basierte Zugriffskontrolle, stellt dies einen mächtigen Ansatz zur Spezifizierung von Zugriffsrechten dar. Dies beinhaltet ebenfalls einen bedeutenden administrativen Vorteil [CGR03].

Im Weiteren wird anhand der XQuery-basierten Policy beispielhaft die Struktur von diesen Policies erläutert und die Zugriffsentscheidungen für die Beispielpolicy ermittelt.

2.4.2.1. XQuery-basierte Policy

Die Abbildung 2.4.2 stellt eine Policy dar, die die Zugriffsentscheidung innerhalb eines Krankenhauses trifft. Sie ermittelt welchen Zugriff ein Angestellter des Krankenhauses, in diesem Fall Arzt und Schwester, auf Patientendaten erhält. Die Struktur dieser Policy wird im Folgenden betrachtet.

Im ersten Teil der Policy werden die relevanten XML-Dokumente identifiziert. In diesem Beispiel werden zwei Dokumente referenziert. Zum Einen das Dokument (siehe Abbildung 8.4.1), in dem die Patientendaten und deren medizinische Daten spezifiziert sind, für die Zugriffsprivilegien vergeben werden sollen. Zum Anderen erfolgt eine Referenz auf das Dokument (siehe Abbildung 8.4.2), welches Informationen über das Krankenhauspersonal und dessen Verantwortlichkeiten enthält. Weiterhin beinhaltet es Informationen, die zur Ableitung von Privilegien benutzt werden. In den nachfolgenden drei for-Schleifen werden Variablen, mittels XPath, an bestimmte XML-Tags in dem gegebenen XML-Dokument gebunden. Die for-Schleifen ermöglichen die Iteration über die adressierten XML-Tags. Die erzeugten Privilegien werden in triple-form: <*Subjekt, Objekt, sowie das gewährte Zugriffsrecht des Subjektes auf das Objekt*>, siehe Abbildung 2.4.3, ausgegeben. Dabei ist *access(subject, object)* eine Funktion, die das Zugriffsrecht des Subjektes auf das Objekt zurückgibt.

```
let $hospital = document(”/hospital.xml”)
let $office = document(”/office.xml”)

for $a in $hospital/PatientRecords/Patient/Medical return
      for $b in $office/Staff/Employee return
            for $c in $office/Staff/Employee
                  where $b/@Name = $a/Doctor and
                        $c/@Name = $b//AccountableTo
                        return <$c/@Name,$a,access($b/@Name, $a)>
```

Abbildung 2.4.2 XQuery-basierte Policy [CGR03]

```
<Brian, hospital.xml/PatientRecords/Patient[@Name=Aaron]/Medical,
OVERWRITE >
<David, hospital.xml/PatientRecords/Patient[@Name=Christy]/Medical,
OVERWRITE >
<Fred, hospital.xml/PatientRecords/Patient[@Name=Emily]/Medical,
OVERWRITE >
<Greg, hospital.xml/PatientRecords/Patient/Medical, READ >

<!--abgeleitete Privilegien: -->

<David, hospital.xml/PatientRecords/Patient[@Name=Aaron]/Medical,
OVERWRITE>
<David, hospital.xml/PatientRecords/Patient[@Name=Emily]/Medical,
OVERWRITE>
```

Abbildung 2.4.3 Privilegen [CGR03]

Im Anhang befindert sich ein weiteres Beispiel für Policies (siehe Abbildung 8.4.3), welches dem Arzt den *Read*-Zugriff auf jede Patientenakte, im Speziellen auf den Personal/DoB und den Overwrite-Zugriff auf die medizinische Patientenakte erlaubt.

2.4.3. XSLT

Die Extensible Stylesheet Language Transformation, kurz XSLT, ist eine Template-basierte Sprache, die turing-vollständig [XSLT08] ist. Sie stellt eine leistungs-fähige Technik zur Transformation von XML-Dokumenten in andere Formate dar. Der Einsatzbereich von XSLT ist vielfältig, so wird XSLT beispielsweise als XML-Anfragesprache und zur Code-Generierung verwendet, sowie zur Dokumentation von SOAP-Diensten [Man06].

XSLT besitzt jedoch einige Beschränkungen, wie zum Beispiel das Variablen keine Variablen, sondern Konstanten sind. Das bedeutet, ist der Wert einer XSLT-Variable einmal festgelegt, so kann dieser nicht mehr geändert werden. Ein weiterer interessanter Aspekt ist das Zwischenspeichern von Werten, die zu einem bestimmten Zeitpunkt ausgelesen, gelöscht und wieder neu belegt werden. Durch die Einbindung von Java-Klassen in XSLT können Beschränkungen von dieser XML-Sprachausprägung aufgehoben werden und die gesamte Java Funktionalität steht nun in XSLT zur Verfügung [See08].

2.4.3.1. XSLT-Templates

Eine XSLT-Transformation besteht aus einer oder mehreren Transformations-regeln, Templates (dt. Schablonen) genannt. Ein Template beschreibt ein Muster, welches XPath als wichtige Ausdruckssprache für die XML-Verarbeitung integriert. XPath übernimmt in XSLT drei entscheidende Aufgaben. Zum Ersten wird XPath innerhalb von Templates genutzt, um Daten eines Dokumentes bei

der Transformation zu adressieren und zu extrahieren. Zum Zweiten dient die XPath-Syntax als Mustersprache in den Filterregeln für Templates. Drittens ermöglichen XPath-Operatoren und XPath-Funktionen einfache Berechnungen und Stringmanipulationen [Man06].

Ein XSLT-Dokument mit einem Template, das zur Transformation eines EPCIS-Dokument angewendet werden kann, wird als Beispiel in der Abbildung 2.4.4 gezeigt. Das dargestellte Template zählt die enthaltenen ObjectEvent-Tag eines Quelldokumentes iterativ und gibt diese Zahl innerhalb des Count-Tags für jeden ObjectEvent-Tag im transformierten Dokument aus. Das Zählen der ObjectEvent-Tags wird durch die Java-Erweiterung realisiert.

Für die Einbindung und Verwendung von Java-Klassen in XSLT sind folgende Schritte erforderlich [See08]:

- eine Java-Klasse, die idealerweise in einem separaten Paket vorliegt
- die Zuweisung des Klassennamens inklusive des Paketes zum Namensraum von XSLT
- das Erzeugen einer so genannten Xalan-Component, d.h. der konkreten Einbindungen von einer Java-Klasse
- die Initialisierung der Java-Klasse als Objekt.
- der Aufruf der Methoden aus der Java-Klasse als Funktion oder Element

Im Allgemeinen besteht ein XSLT-Dokument, wie in Abbildung 2.4.4, aus der einleitenden XML-Deklaration und dem Stylesheet-Element, dessen Attribute die verwendete XSLT-Version für den verarbeitenden XSLT-Parser und die Namensräume spezifizieren.

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

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xalan="http://xml.apache.org/xalan"
xmlns:java="http://xml.apache.org/xslt/java" exclude-result-prefixes="xalan
java"
xmlns:my-counter="xml.Counter"
extension-element-prefixes="my-counter">

<xsl:output method="xml" encoding="UTF-8" omit-xml-declaration="no"/>

<xalan:component prefix="my-counter" elements="init" functions="count
nextcount getcount index setindex setcount">
       <xalan:script lang="javaclass" src="xalan://xml.Counter"/>
</xalan:component>

<my-counter:init/>

<xsl:template match="/">
       <xsl:value-of select="my-counter:setcount(0)"/>

       <xsl:for-each select="//ObjectEvent">
       <xsl:variable name="objectevent" select="my-counter:count()"/>
       <count> ObjectEvent {$objectevent} </count>
       </xsl:for-each>

</xsl:template>

</xsl:stylesheet>
```

Abbildung 2.4.4 Beispiel eines XSLT-Template

Das *exclude-result-prefixes*-Attribut teilt dem XSLT-Parser mit, dass im transformierten Dokument die angegebenen Präfixe nicht mit übertragen werden. Mit dem *extension-element-prefixes*-Attribut werden die Präfixe der Namensräume aufgelistet, die innerhalb des Stylesheet benötigt werden, um Elemente und Funktionen dieser Namensräume zu erkennen. Innerhalb des Stylesheet-Elements werden Ausgabeinformationen und Templates definiert. Des Weiteren können die Komponenten für Java-Klassen definiert und initialisiert werden. Die Counter-Klasse, die im nachstehende Beispiel verwendet wird, ist in der Abbildung 8.5.1 dargestellt. Weitere Erläuterungen zur Abbildung 2.4.4 oder zu anderen XSLT-Beispielen sind in [See08] und [Man06] zu finden.

2.4.4. Zusammenfassung

Die drei vorgestellten Ansätze werden noch einmal kurz zusammengefasst und anschließend werden diese Ansätze im Bezug auf die Entwicklung der geschäftstransaktionsbasierten Zugriffsformulierung bewertet.

Ein eventbasiertes Zugriffskontrollsystem ermöglicht eine bessere Response-Time für den Nutzer, als konventionelle Zugriffssysteme. Die Merkmale, wie zeitliche Zugriffskontroll-Policies, referenzieller Zugriff, zusammengesetzte Events, Eventausführungszeit oder externe Eventregistrierung und -behandlung, werden von diesem System unterstützt. Die Kosten für dynamische Änderungen von Zugriffsprivilegien, die zeitgleich mit jedem auftretenden Event innerhalb des EBAC-Systems erfolgen, können durch verschiedene Methoden reduziert werden. Zum Beispiel ist eine inkrementelle Bewertung von Policies denkbar [BGMP03].

Die dokument- bzw.datenbankbezogene Herleitung von Zugriffskontrollregeln, ist durch den Einsatz von XQuery möglich. Dieser Ansatz bietet eine neue Flexibilität in der Formulierung von Berechtigungsbedindungen. Es können kontextsensitive und feingranulare Beschränkungen definiert werden. Das ist vorteilhaft für die Administration von Zugriffsberechtigungen.

Die Template-basierte Sprache XSLT stellt eine leistungsfähige Technik zur Transformation von XML in das jeweils gewünschte Format dar. Diese Transformationsprache setzt die Kenntnis der Struktur des zu transformierenden XML-Dokumentes voraus. Um auch auf Daten eines XML-Dokumentes, dessen Schema nicht bekannt ist, zugreifen zu können, wurde XQuery entwickelt. XQuery ist eine standardisierte Abfragesprache [Pat03].

Im Rahmen der geschäftstransaktionsbasierten Formulierung von Zugriffs-berechtigungen ist die Entscheidung, welche der beiden XML-basierten Sprachen verwendet wird, zu treffen. Die Geschäftstransaktionen sind in EPCIS-Events referenziert. Die Struktur dieser EPCIS-Events ist bekannt. Die zu erstellenden Zugriffsberechtigungen sind in Form von aidXACML-PolicySets zu formulieren und in den Policy Access Point zu integrieren. Der Policy Access Point enthält die Zugriffsberechtigungen für das Zugriffskontrollsystem des EPCIS-Repository. Das eingehende EPCIS-Event ist zu einem entsprechenden PolicySet, das die gewünschte Berechtigung enthält, zu transformieren. Für die Umwandlung von Events zu PolicySets wird dem zu Folge XSLT verwendet. Um die Formulierung von Zugriffsberechtigung auszulösen und umsetzen zu können, wird eine ereignisbasierte Systemarchitektur benötigt. Im Kapitel 4. wird eine solche Systemarchitektur vorgestellt und auf die konkrete Berechtigungs-formulierung mittels XSLT-Stylesheet eingegangen.

2.5. Standardisierung von Geschäftsdokumenten

Die Kommunikation zwischen Geschäftspartnern erfolg heute vorrangig per Internet. Um den Austausch von Geschäftsdokumenten, wie Bestellungen, Rechnungen etc. zwischen verschiedenen Firmen zu ermöglichen und firmenintern die Daten dieser Dokumente verwenden zu können, sind einheitliche Standards notwendig. EDIFACT bzw. openTrans repräsentieren international gültige Spezifikation für elektronische Dokumente.

2.5.1. EDIFACT

EDIFACT steht für Electronic Data Interchange For Administration, Commerce und Transport und ist ein branchenübergreifender internationaler Standard, der elektronische Datenformate für Geschäftstransaktionen spezifiziert. Für jede Geschäftstransaktion wurde ein eigener Nachrichtentyp entwickelt, wie beispielsweise ORDERS für Bestellung oder INVOIC für Rechnung. In einzelnen Branchenbereichen reichten die durch EDIFACT definierten Datenformate nicht aus. Es entstanden branchenspezifische EDIFACT-Subsets, die die benötigten Formate umfassten. Beispielsweise für die chemische Industrie gibt es das EDIFACT-Subset CEFIC, für die Konsumgüterwirtschaft EANCOM, für Baumärkte EDIBDB, für die elektrische Industrie EDIFICE, sowie für die Möbelindustrie EDIFURN. In der Tabelle 2.5.1 sind die häufigsten EDIFACT-Nachrichtentypen aufgeführt. Die Datenumwandlung zwischen EDIFACT und unternehmensinternen Verarbeitungsformaten erfolgt durch sog. EDI-Konverter. Für die Übertragung der Daten steht der internationale Standard X.400 zur Verfügung [Lip07].

Nachrichtentypen für Geschäftstransaktionen	
PRICAT	Preisliste / Katalog
ORDERS	Bestellung
ORDRSP	Auftragsbestätigung
IFTMIN	Transportauftrag
DESADV	Lieferavis
IFTSTA	Transportstatus
IFTMAN	Ankunftsmeldung
RECADV	Wareneingang
INVOIC	Rechnung
REMADV	Zahlungsavis
PAYMUL	Zahlungsauftrag
CREMUL	Gutschrift

Tabelle 2.5.1 EDIFACT-Nachrichtentypen und deren Verwendungszweck [EDI08]

Die Abbildung 2.5.1 zeigt einen Kreislauf der elektronischen Geschäftsprozesse, die aus mehreren Transaktionen in Form von EDIFACT-Nachrichtentypen bestehen können. Derzeit wird in der Arbeitsgruppe von GS1 Germany ausgearbeitet, wie die gemeinsame Nutzung von EPCIS und EANCOM aussehen kann. So kann der Anwender bereits den elektronischen Lieferavis zur

Übermittlung von EPC-Identen in der Liefermeldung *(DESADV)* nutzen und das EPCIS ergänzend einsetzen [Kuh08].

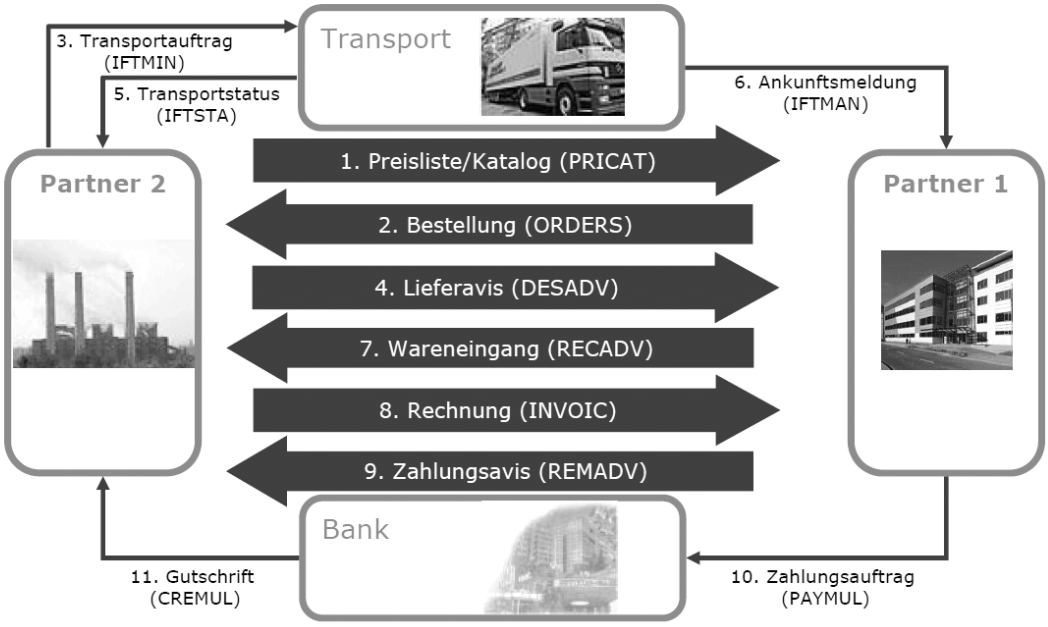

Abbildung 2.5.1 Kreislauf der elektronischen Geschäftsprozesse [EDI08]

2.5.2. openTrans

Der internationale Standard openTrans stellt einheitliche elektronische Dokumente auf Basis von XML für den zwischenbetrieblichen E-Commerce bereit. Somit können Geschäftsdokumente, wie Auftrag, Lieferschein oder Rechnung (siehe Tabelle 2.5.2), zwischen den Geschäftspartnern und elektronischen Marktplätzen ausgetauscht werden. Die Grundlage für openTrans bildet die BMEcat-Spezifikation für den elektronischen Produktdatenaustausch. Die Prozessgestaltung, welche Art von openTrans-Dokumente verwendet und in welcher Reihenfolge und Richtung diese ausgetauscht werden, liegt im Ermessen der Geschäftspartner. Durch die Verwendung von XML zur Spezifikation von Geschäftsdokumenten wird die gleichzeitige Kodierung von Strukturen und Daten in einem Geschäftsdokument möglich. Für jedes Dokument wurde ein entsprechendes XML-Schema definiert [SKO08].

Nachrichtentypen für Geschäftstransaktionen	
RQF	Angebotsaufforderung
QUOTATION	Angebot
ORDER	Bestellung
ORDERRESPONSE	Auftragsbestätigung
ORDERCHANGE	Bestellungsänderung
DISPATCHNOTIFICATION	Lieferavis
RECEIPTACKNOLEDGEMENT	Wareneingangsbestätigung
INVOICE	Rechnung

Nachrichtentypen für Geschäftstransaktionen	
INVOICELIST	Rechnungsliste
REMITTANCEADVICE	Zahlungsavis

Tabelle 2.5.2 openTrans-Nachrichtentypen und deren Verwendungszweck [SKO08]

In der Abbildung 2.5.2 wird eine Möglichkeit zur Gestaltung eines Geschäftsprozesses für die katalogbasierte Beschaffung dargestellt.

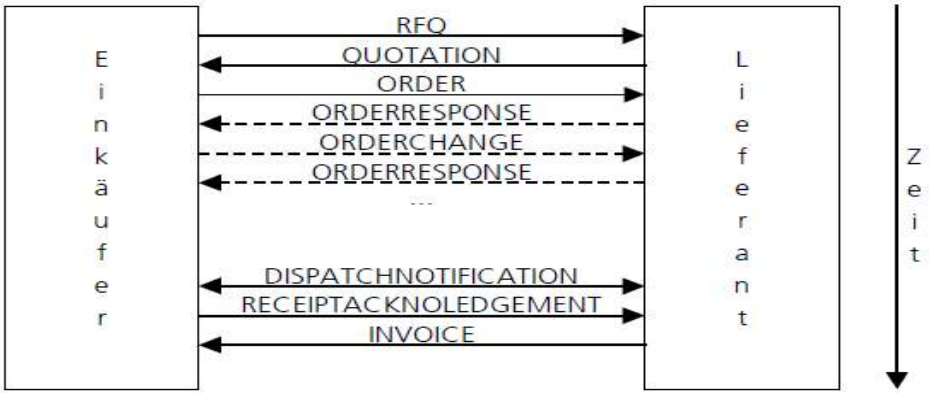

Abbildung 2.5.2 Beziehung zwischen Einkäufer und Lieferant [SKO08]

2.5.3. Vergleich EDIFACT und openTrans

Die Industriestandards, EDIFACT und openTrans, spezifizieren standardisierte Dokumente für die einheitliche Kommunikation zwischen Geschäftspartnern.

EDIFACT bietet eine Vielzahl branchenspezifischer Dokumente an. Die Bearbeitung von Daten, die unter Verwendung von EDIFACT-Dokumenten übermittelt worden sind, kann nicht direkt vorgenommen werden. Es ist zuvor eine Konvertierung des Dokumentes notwendig. Anders dagegen ist es bei Dokumenten die mittels OpenTrans übermittelt worden sind, diese können direkt durch XML-unterstützende Systeme verarbeitet werden.

OpenTrans stellt XML-basierte Geschäftsdokumente bereit, die zwischen Lieferant und Einkäufer ausgetauscht werden. Dieser Industiestandard wurde von Fraunhofer IAO, in Kooperation mit der Universität Duisburg-Essen, sowie Vertretern aus der Industrie 2001 entwickelte. Mit der zweiten Version des openTrans 2.0 sind 10 Geschäftsdokumente für die E-Commerce-Kommunikation definiert wurden [OT08].

In der Tabelle 2.5.3 erfolgt die Gegenüberstellung der Dokumenttypen beider Standards. EDIFACT besitzt im Vergleich zu openTrans sehr viele Dokumenttypen. Die Tabelle zeigt nur einige ausgewählte EDIFACT Dokumenttypen, die zum Kontext der Konsumgüterindustrie (EANCOM) gehören. Die openTrans Dokumenttypen sind hingegen branchenunabhängig. Beide vorgestellten Standards sind hersteller- und anbieterneutral. Sie sind mit Infrastrukturen, wie Internet oder Intranet, austauschbar.

Die Betrachtung dieser Standards gibt für die weitere Arbeit Aufschluss, über existierende Geschäftstransaktionen, die im Rahmen des EPCIS-Systems auftreten können.

EDIFACT	*openTrans*
RRICAT	-
-	RQF
-	QUOTATION
ORDERS	ORDER
ORDRSP	ORDERRESPONSE
-	ORDERCHANGE
IFTMIN	-
DESADV	DISPATCHNOTIFICATION
IFTSTA	-
IFTMAN	-
RECADV	RECEIPTACKNOLEDGEMENT
INVOIC	INVOICE
-	INVOICELIST
PAYMUL	-
REMADV	REMITTANCEADVICE
CREMUL	-

Tabelle 2.5.3 Gegenüberstellung der Geschäftsdokumente ([EDI08], [SKO08])

3. Problem- und Anforderungsanalyse

In diesem Kapitel werden Geschäftstransaktionen innerhalb einer expliziten Wertschöpfungskette untersucht. Im Rahmen der Systematisierung dieser Geschäftstransaktionen und relevanter Anwendungsfälle werden bestehende Probleme aufgezeigt, aus denen Anforderungen für die Automatisierung von Zugriffsberechtigungen definiert werden. Im Kapitel 4. wird ein Konzept, basierend auf dieser Analyse, erstellt.

3.1. Wertschöpfungskette

Die Idee und die Konzeption der Wertschöpfungskette wurde erstmals 1985 von dem Wirtschaftswissenschaftler Michael E. Porter vorgestellt.

Wertschöpfungsketten, engl. Supply Chains, beschreiben verschiedene sequenzielle Aufgaben zur Leistungserbringung. Die Darstellung reicht meist von der Entwicklung, über die Beschaffung und Produktion, über den Vertrieb bis zum Inkasso und nachlaufenden Serviceleistungen. Das Geschäftsmodell und die integrierten Kooperationspartner bestimmen den Aufbau von Wertschöpfungsketten. [Kaa07]

3.1.1. Spezielle Wertschöpfungskette

Die Abbildung 3.1.1 zeigt eine Wertschöpfungskette, die aus den Teilnehmern Hersteller, Händler, Transportunternehmen und Kunde besteht. Im konkreten Fall ist der Händler ein Internetversandhaus, welches Waren von verschiedenen Herstellern und eigene Produkte vertreibt. Wegen der Übersichtlichkeit umfasst die dargestellte Wertschöpfungskette nur einen Hersteller, der zwei verschiedene Produktionslinien besitzt. Die Produkte werden mittels Transportunternehmen dem Internethändler, in Folge seiner getätigten Bestellung, vom Hersteller zugesandt. Die vom Hersteller bezogenen und die eigenen Produkte des Händlers können über das Internet bestellt werden. Die Kundenbestellung wird durch den Händler bearbeitet, die gewünschten Produkte werden gegebenenfalls beim Hersteller nachbestellt, zusammengestellt, verpackt und anschließend über ein Transportunternehmen an den Kunden geliefert.

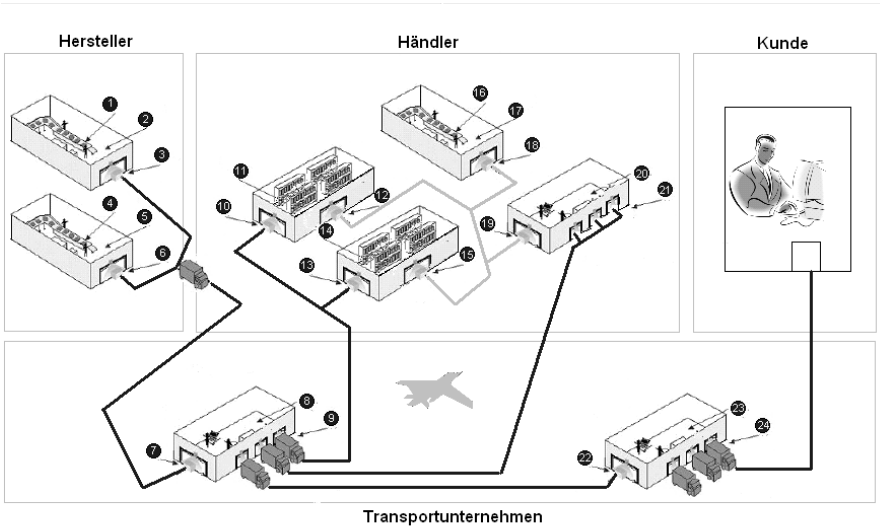

Abbildung 3.1.1 Wertschöfungskette (siehe vergrößerte Abbildung 8.6.1)

Das Transportunternehmen in der vorgestellten Wertschöpfungskette ist sehr abstrakt dargestellt. In der Praxis ist ein solches Unternehmen sehr komplex. Es umfasst meist mehrere Zwischenlager. Die Transportgüter können auf verschiedenen Transportwegen befördert werden. So kann ein bestelltes Produkt vom Händler bis zum Kunden verschiedene Lager passieren und die Zulieferung per Schiff, per Flugzeug, per LKW etc. oder eine Kombination aus verschiedenen Transportmitteln erfolgen.

Der Fokus dieser Arbeit liegt auf die Kommunikationsbeziehung zwischen den Teilnehmer der Wertschöpfungskette. Somit wird auf die Teilnehmer selbst nur abstrakt eingegangen.

3.2. Systematisierung von Geschäftstransaktionen

3.2.1. Geschäftstransaktionen

Eine Geschäftstransaktion ist die Übermittelung eines Geschäftsdokumentes von einem Geschäftspartner (des Erzeugers) zu einem anderen Geschäftspartner (dem Empfänger) [SKO08].

In der Regel ist ein Geschäftsdokument, wie Bestellung oder Rechnung, ein standardisiertes Dokument, beispielsweise des Dokumenttyps EDIFACT oder openTrans.

Im Rahmen dieses Arbeit werden eigene Dokumenttypen eingeführt, um unabhängig von bestehenden Standards die Geschäftstransaktionen als Auslöser für Berechtigungen zu betrachten. Der Bezug zwischen eigenen Dokumenttypen, EDICFACT- und openTrans-Dokumenttypen wird in Tabelle 3.2.1 dargestellt, so das die Entscheidung, welches Format zu verwenden ist, dem Endnutzer offen gelassen wird.

EDIFACT	***openTrans***	***eigene***
	RQF	RQ
	QUOTATION	QU
ORDERS	ORDER	PO
ORDRSP	ORDERRESPONSE	ORP
	ORDERCHANGE	POC
IFTMIN		IT
DESADV	DISPATCHNOTIFICATION	DAD
IFTSTA		TS
INVOIC	INVOICE	IV
PAYMUL		PAO
REMADV	REMITTANCEADVICE	PA
CREMUL		CR

Tabelle 3.2.1 EDIFACT-, openTrans-Dokumenttypen und eigene Dokumenttypen

3.2.1.1. Geschäftsdokumente in der Wertschöpfungskette

Dieser Abschnitt beschreibt ein Kommunikationsszenario zwischen den Geschäftspartnern der zuvor vorgestellten Wertschöpfungskette (siehe Abbildung 3.1.1). Die Kommunikation im Szenario erfolgt unter Verwendung der eigenen Dokumenttypen, die in der Tabelle 3.2.1 in Beziehung zu den internationalen Standards EDIFACT und openTrans gesetzt wurden.

Die grafische Darstellung des Szenarios ist in der Abbildung 3.2.1 zu sehen. Neben den Geschäftspartnern, Kunde, Händler, Hersteller und Transportunternehmen, ist ein weiterer Partner, die Bank, in diesen Kommunikationsprozess eingebunden worden.

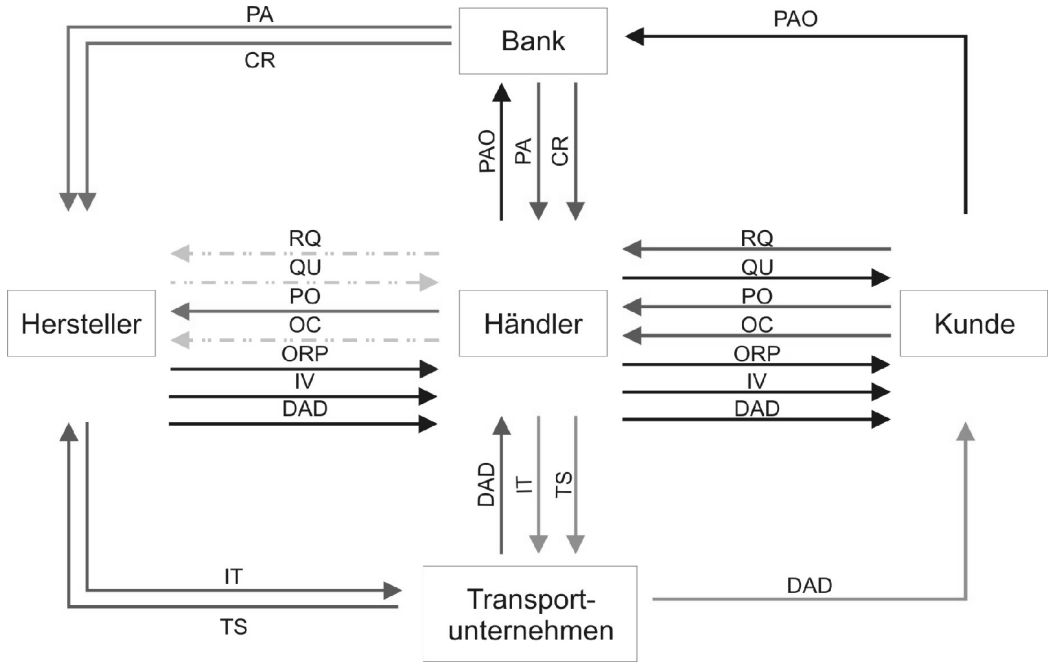

Abbildung 3.2.1 Austausch von Geschäftsdokumenten zwischen Teilnehmer der Wertschöpfungskette

Ein Kunde hat Bedarf an unterschiedlichen Produkten und orientiert sich per Internet. Um die Preise und die Verfügbarkeit der gewünschten Produkte vom Internethändler zu erfahren, stellt er eine Angebotsaufforderung *(RQ)* an diesen. Der Händler erstellt für den Kunden ein entsprechendes Angebot *(QU)*. In Folge des Angebotes entscheidet sich der Kunde die Bestellung der Produkte auszulösen *(PO)*. Nach dem Versenden des elektronischen Bestellformulars fällt ihm ein Fehler in der getätigten Bestellung auf. Er storniert einen Teil der getätigten Bestellung *(POC)* und bestellt gleichzeitig noch ein weiteres Produkt. Die endgültige Bestellung wird vom Händler bestätigt *(ORP)* und eine entsprechende Rechnung *(IV)* wird dem Kunden zugesandt. Nachdem der Kunde seine Bank angewiesen hat, den Rechnungsbetrag an den Händler zu überweisen *(PAO)*, erhält der Händler eine Nachricht über den Eingang des Zahlungsauftrages *(PA).* Der Händler löst daraufhin den Transportauftrag *(IT)* aus und benachrichtigt den Kunden, sobald die Waren vom Transportunternehmen abgeholt worden sind *(DAD)*. Der Händler kann den Transportstatus seiner versendeten Waren beim Transportunternehmen verfolgen bzw. abfragen. Sobald die Ware beim Kunden zugestellt wurde, erhält

der Händler die Nachricht (TS) mit dem Transportstatus „Ware ausgeliefert“. Die Bestellung ist somit abgeschlossen. Das Transportunternehmen selbst kann dem Kunden eine Nachricht über die voraussichtliche Ankunft der bestellten Ware übermitteln *(DAD).* Die Gutschrift *(CR)* für die bestellten Produkte ist zwischenzeitlich erfolgt.

Das bisher beschrieben Szenario repräsentiert nur den rechten Teil, des in der Abbildung 3.2.1 gezeigten Kreislaufes von Geschäftstransaktionen. Das Teilszenario zwischen Hersteller und Händler entspricht dem zwischen Händler und Kunde, wobei der Hersteller die Rolle des Händlers und der Händler die Rolle des Kunden einnimmt. Die Transaktionen für Angebotsaufforderung *(RQ),* Angebot (*QU*) und Bestellungsänderung (*POC*) werden jedoch eher selten in der Kommunikationsbeziehung zwischen Hersteller und Händler angewendet. Des Weiteren kann die Rechnungslegung und der Versand der bestellten Ware gleichzeitig erfolgen, da ein Vertrauensverhältnis zwischen beiden Geschäftspartnern besteht und somit die Ware beispielsweise erst während oder nach deren Lieferung bezahlt wird.

Eine komplette Änderung in der Kommunikationsbeziehung, zwischen Hersteller und Händler, ist durch Outsourcen der Lagerverwaltung möglich. Das bedeutet, dass der Händler den Bestand seines Lagers durch die Hersteller eigenständig bestücken lässt. Der Händler gibt dabei die Anzahl bzw. die Menge der Produkte vor, die mindestens im Lager vorhanden sein müssen. Um dies zu realisieren benötigen die Zulieferer des Händlers die entsprechende Berechtigungen, um den Lagerbestand für die zu liefernden Produkte abfragen zu können. Das Outsourcen der Lagerhaltung von Herstellern, Vendor Managed Inventory, wird in dieser Arbeit nicht weiter vertieft.

3.2.1.2. Geschäftstransaktionen im EPCIS-Eventtyp

Ein TransactionEvent ist ein bestimmter EPCIS-Eventtyp (siehe Kapitel 2.1.2.2.), der die Assoziation oder Disassoziation von EPC-Objekten mit einer oder mehreren Geschäftstransaktionen beschreibt.

Im TransactionEvent sind Geschäftstransaktionen, innerhalb des *bizTransactionList*-Element, referenziert. In den EPCIS-Eventtypen, ObjectEvent, QuantityEvent und AggregationEvent, wird dieses Element optional verwendet, um den Kontext in dem das EPCIS-Event steht anzugeben. Das *bizTransactionList*-Element enthält die Identifier, *bizTransaction* und *type*, beide sind vom Typ anyURI [BEA06]. Die Abbildung 3.2.2 zeigt das *bizTransactionList*-Schema, welches die Beziehungen zwischen *bizTransactionList* und den zwei Identifier verdeutlicht.

Ein *bizTransactionList*-Element enthält ein oder mehrere *bizTransaction*-Elemente. Jedes *bizTransaction*-Element sollte das *type*-Attribut besitzt. Wird dieses Attribut weggelassen, so ist die Information über den Typ der Geschäftstransaktion nicht verfügbar [BEA06].

Die Abbildung 3.2.3 zeigt ein Beispiel für die Identifizierung einer Bestelltransaktion mittels *bizTransactionList*-Element. In den Masterdatentypen werden mit *bizTransaction* die Business Transaction und mit *type* der Business Transaction Type assoziiert. Der Substring *po* ist in beiden Elementen enthalten und kennzeichnet die Bestellung.

```
<!-- bizTransactionList -->
<xsd:element name="bizTransactionList"
   type="epcis:BusinessTransactionListType" minOccurs="0"/>

<!-- bizTransactionListType -->
<xsd:complexType name="BusinessTransactionListType">
  <xsd:sequence>
    <xsd:element name="bizTransaction"
       type="epcis:BusinessTransactionType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<!-- BusinessTransactionType -->
<xsd:complexType name="BusinessTransactionType">
  <xsd:simpleContent>
    <xsd:extension base="epcis:BusinessTransactionIDType">
      <xsd:attribute name="type"
        type="epcis:BusinessTransactionTypeIDType" use="optional"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Abbildung 3.2.2 bizTransactionList-Schema

```
<bizTransactionList>
    <bizTransaction type="urn:epcglobal:fmcg:btt:po">
       http://transaction.acme.com/po/54847
    </bizTransaction>
</bizTransactionList>
```

Abbildung 3.2.3 Referenz zur Geschäftstransaktion Bestellung

Probleme	***Anforderungen***	
Der Identifier *type* ist optional.	Eine eindeutige Identifizierung von Geschäftstransaktionen im *bizTransaction*-Element.	Q1
Mehreren Geschäftstransaktionen sind in einem TransactionEvent referenzierbar.	Jede referenzierte Geschäftstrans-aktion eines TransactionEvents ist bei der Formulierung von Berechtigungen zu prüfen.	Q2

Tabelle 3.2.2 Probleme und Anforderungen zu Geschäftstransaktionen im EPCIS-Event

3.2.1.3. Geschäftstransaktionen und Geschäftsdokumente

Bei Geschäftstransaktionen werden standardisiere, elektronische Dokumente von einem Geschäftspartner zum Anderen übertragen. Wie die Übertragung erfolgt und wie die eingehenden Nachrichten im System der Geschäftspartner verarbeitet werden, steht in dieser Arbeit außerhalb der Betrachtung. Es wird die Annahme getroffen, das die eingehenden Geschäftsdokumente am EPCIS Capture Interface in Form von EPCIS-TransactionEvents eintreffen.

Im vorherigen Abschnitt wurde gezeigt, dass Geschäftstransaktionen innerhalb von TransactionEvents durch die Elemente, *bizTransaction* und *type,* identifiziert sind. Die *Vocabularies*[2] der beiden Elemente wurden nicht betrachtet, dies soll in diesem Abschnitt vorgenommen werden. Die Recherche zum Vocabulary der

2) Vocabulary eines Masterdatentypes bestehen aus Vocabulary-Elementen, die zur Definition von EPCIS-Event-Elementen und -Attributen verwendet werden. Das EPCIS-Event beschreibt mittels Vocabulary-Elementen einen bestimmtes Ereignis.

Masterdatentypen dieser Elemente hat ergeben, das es für die in dieser Arbeit verwendeten Geschäftstransaktionen nur unzureichende *VocabularyElemente* zur Verfügung stehen. Ein Auszug der Recherche ist im Kapitel 8.2. zu finden.

Für die weiterführende Arbeit wurden daher eigene *VocabularyElemente* verwendet, die in der Tabelle 3.2.3 aufgeführt sind. Diese Tabelle enthält Geschäftsdokumente und deren Referenzen durch die beiden Identifier. Alle Transaktionen des Szenario im Kapitel 3.2.1.1. sind beinhaltet.

Geschäfts-dokument	***Identifier***	
	bizTransaction	*type*
RQ	`http://.../rq/...`	`urn:...:rq`
QU	`http://.../qu/...`	`urn:...:qu`
PO	`http://.../po/...`	`urn:...:po`
ORP	`http://.../orp/...`	`urn:...:orp`
POC	`http://.../poc/...`	`urn:...:poc`
IT	`http://.../it/...`	`urn:...:it`
DAD	`http://.../dad/...`	`urn:...:dad`
TS	`http://.../ts/...`	`urn:...:ts`
IV	`http://.../iv/...`	`urn:...:iv`
PAO	`http://.../pao/...`	`urn:...:pao`
PA	`http://.../pa/...`	`urn:...:pa`
CR	`http://.../cr/...`	`urn:...:cr`

Tabelle 3.2.3 Geschäftsdokumente und deren Identifizierung im TransactionEvent

Annahme	
Eingehende Geschäftsdokumente werden am EPCIS Capture Interface, in Form von EPCIS-TransactionsEvents, übergeben.	A1
Geschäftsdokumente werden wie in Tabelle 3.2.3 referenziert.	A2

Tabelle 3.2.4 Annahme zu Geschäftstransaktionen und Geschäftsdokumenten

3.2.1.4. Geschäftstransaktionen und Berechtigungen

Eine Geschäftstransaktion, referenziert im TransactionEvent, soll die Vergabe oder den Entzug von Berechtigungen auslösen. Welche Bedingungen dafür erfüllt sein müssen und bei welchen Transaktionen die Vergabe und der Entzug von Berechtigungen möglich bzw. sinnvoll ist, wird im Weiteren untersucht. Für diese Untersuchung wird das Schema des TransactionEvents und die Geschäftstransaktionen der Abbildung 3.2.1 herangezogen.

Das TransactionEvent-Schema ist in der Abbildung 3.2.4 dargestellt. Ein TransactionEvent (siehe Kapitel 2.1.2.2.) ist durch das *TransactionEvent*-Element gekennzeichnet. Es erbt Elemente vom EPCIS-Event und umfasst die Elemente *bizTransactionList, EPCList und action*, sowie weitere optionale Elemente siehe Tabelle 2.1.2 .

Das *action*-Element nimmt einen der vorgegebenen Werte, ADD, OBSERVE oder DELETE an. Die Bedeutung der einzelnen Wertbelegungen wird am Beispiel eines TransactionEvents, welches eine Geschäftstransaktion referenziert, erläutert.

Der *action*-Wert ADD wird verwendet, wenn eine Transaktion zum ersten Mal aufgetreten ist, oder wenn neue EPCs zu einer schon existierenden Transaktion hinzugefügt werden [EPC07].

Der Wert DELETE hingegen zeigt an, das eine EPC-Teilmenge von einer Transaktion getrennt wird, oder das diese Geschäftstransaktion beendet ist. Bleibt die EPC-Liste eines TransactionEvents leer, wurden alle EPCs von der Transaktion getrennt und diese ist beendet oder wurde abgebrochen [EPC07].

Wird der *action*-Wert des TransactionEvents mit OBSERVE angegeben, ist das TransactionEvent ganz ähnlich dem ObjectEvent, welches eine nicht leere Menge von *bizTransaction*-Elementen besitzt. In der Praxis muss durch den Endnutzer klar definiert werden, wann welches der beiden Events genutzt wird [EPC07]. Es wurde die Annahme getroffen, das ausschließlich das ObjectEvent statt dem TransactionEvent eingesetzt wird.

```
<xsd:complexType name="TransactionEventType">
 <xsd:annotation>
  <xsd:documentation xml:lang="en">
   Transaction Event describes the association or disassociation of
   physical objects to one or more business transactions.
  </xsd:documentation>
 </xsd:annotation>

 <xsd:complexContent>
  <xsd:extension base="epcis:EPCISEventType">
   <xsd:sequence>

    <xsd:element name="bizTransactionList"
    type="epcis:BusinessTransactionListType"/>
    <xsd:element name="parentID" type="epcis:ParentIDType" inOccurs="0"/>
    <xsd:element name="epcList" type="epcis:EPCListType"/>
    <xsd:element name="action" type="epcis:ActionType"/>
    <xsd:element name="bizStep" type="epcis:BusinessStepIDType"
    minOccurs="0"/>
    <xsd:element name="disposition" type="epcis:DispositionIDType"
    minOccurs="0"/>
    <xsd:element name="readPoint" type="epcis:ReadPointType"
    minOccurs="0"/>
    <xsd:element name="bizLocation" type="epcis:BusinessLocationType"
    minOccurs="0"/>
    <xsd:element name="extension"
    type="epcis:TransactionEventExtensionType" minOccurs="0"/>
    <xsd:any namespace="##other" processContents="lax" minOccurs="0"
    maxOccurs="unbounded"/>

   </xsd:sequence>
   <xsd:anyAttribute processContents="lax"/>
  </xsd:extension>
 </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="ActionType">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="ADD"/>
  <xsd:enumeration value="OBSERVE"/>
  <xsd:enumeration value="DELETE"/>
 </xsd:restriction>
</xsd:simpleType>
...
```

Abbildung 3.2.4 TransactionEvent-Schema [EPC07]

Bezogen auf den Kontext der Berechtigungen bedeutet das, dass prinzipiell eine Berechtigungsvergabe auf Basis einer Geschäftstransaktion möglich ist, wenn es zu dieser Geschäftstransaktion ein entsprechendes TransactionEvent gibt, welches das *action*-Element mit dem Wert ADD besitzt. Es ist nicht sinnvoll, für jede Geschäftstransaktion Berechtigungen zu vergeben. Die Betrachtung dazu erfolgt in der Tabelle 3.2.5 .

Es ist wichtig, dass die auf diese EPCs vergebenen Berechtigungen auch wieder entzogen werden, wenn EPCs von einer Geschäftstransaktion getrennt werden. Bei der teilweisen Stornierung von bestellten Produkten ist dies der Fall. Ein weiterer Berechtigungsentzug von bereits vergebenen Berechtigungen ist nach der Beendigung einer Transaktion möglich, um nicht mehr benötigte Berechtigungen aus dem System zu entfernen. Dieser Entzug kann sofort nach Beendigung einer Transaktion oder erst nach einer definierten Zeitspanne erfolgen. Eine Änderung der vergebenen, speziellen Berechtigung in eine globale Berechtigung wäre ebenfalls denkbar. Eine globale Berechtigung gibt dem Handelspartner (Subjekt der Berechtigung) beispielsweise nur noch Auskunft über die Herkunft des Produktes. Somit sind spezielle Informationen zum Produkt, wie die Bereitstellungszeit des Produktes im Lager, nicht mehr abfragbar. Die Strategie, nach welchen Kriterien Berechtigungen entzogen werden, ist vom Besitzer des Repositories festzulegen. Prinzipiell kann jedoch gesagt werden, dass ein Berechtigungsentzug auf Basis der Geschäftstransaktion immer dann möglich ist, wenn es zu einer Geschäftstransaktion ein entsprechendes TransactionEvent gibt, welches das *action*-Element mit den Wert DELETE besitzt.

Im Weiteren soll nun die Betrachtung folgen, für welche Geschäftstransaktionen es sinnvoll ist Berechtigung zu vergeben bzw. zu entziehen. Dazu stellt die Tabelle 3.2.5 die Beziehung zwischen Geschäftstransaktionen der Abbildung 3.2.1 und deren Berechtigungen dar. Sie zeigt, für welche Transaktionen es sinnvoll ist, Berechtigungen auf bestimmte EPCIS-Eventtypen zu vergeben. Des Weiteren beinhaltet diese Tabelle, für welche Transaktionen eine Delegation von Berechtigungen erstellt werden kann und auf welches Repository sich die Berechtigungen beziehen. Sie enthält jedoch nicht die explizite Berechtigungsvergabe auf konkrete Elemente der EPCIS-Eventtypen. Diese Vergabe ist abhängig von der Sicherheitspolitik des Unternehmens für das EPCIS-Repository. Die Berücksichtigung der Sicherheitsbedürfnisse aller EPC-Teilnehmer ist im EPCglobal-Netzwork von zentraler Bedeutung. Besonders sensibel ist der Umgang mit Firmendaten bei im Wettbewerb stehenden EPC-Teilnehmern. Beispielsweise ist es für ein Unternehmen wichtig, dass seine Lieferbeziehungen und die internen Produktionsabläufe des Unternehmens vor dem Einblick von Wettbewerbern geschützt ist.

Die Tabelle 3.2.5 zeigt den Zusammenhang zwischen Vergabe und Entzug von Berechtigungen. Es ist sinnvoll Berechtigungen, die auf Basis von Geschäftstransaktionen vergeben wurden, auch wieder zu entziehen, wenn diese Berechtigungen nicht mehr benötigt werden bzw. überflüssig sind. Dieser Entzug ist vom Besitzer des EPCIS-Repository zu definieren.

Im Rahmen der Automatisierung ist ein Berechtigungsentzug, der erst nach einer bestimmten Zeitspanne erfolgt, von einem Berechtigungsentzug, der infolge eines Abbruchs der Transaktion erfolgt, nicht unterscheidbar. In diesem Fall, werden weitere Informationen oder die Entscheidung des Systemadministrators benötigt, um den Entzugszeitpunkt der Berechtigung zu bestimmen.

Das nachfolgende Beispiel beschreibt eine Berechtigung, die in Rahmen einer Angebotsaufforderung (*RQ*) gewährt wird. Die Angebotsaufforderung (*RQ*) eines Kunden, bezogen auf bestimmte Produkte, löst die Formulierung einer Berechtigung für den Kunden auf die Elemente *childEPCs* und *quantity* des QuantityEvents, der entsprechenden Produktklasse im EPCIS-Repository des Händlers, aus. Der Kunde erhält somit die Möglichkeit, die Verfügbarkeit der gewünschten Produkte abzufragen. Die Berechtigung wird mit einer Gültigkeitsdauer beschränkt, um eine permanente Abfrage des Kunden auf bestimmte Produktbestände zu unterbinden. Bestätigt der Kunde das Angebot (*QU*), welches ihm in Folge seiner Angebotsaufforderung (*RQ*) unterbreitet wurde, mit einer verbindlichen Bestellung, so wird ebenfalls die Berechtigung, die ihm zuvor in diesem Zusammenhang gewährt worden ist, entzogen.

Im Weiteren wird der Fokus auf drei Anwendungsfälle, die Bestellung von Produkten und deren Beendigung, sowie der Bestellungsänderung gerichtet. Die Bestellungsänderung stellt einen Sonderfall dar, der die Berechtigungsvergabe bzw. den Entzug von Berechtigungen, oder beides umfassen kann. Anhand dieser Anwendungsbeispiele werden weitere Probleme und Anforderungen betrachtet.

Vor dieser Betrachtung wird das zeitliche Auftreten von Bestelltransaktionen zu unterschiedlichen Produktionsstufen der bestellten Produkte, in Abhängigkeit zu Berechtigungen, untersucht.

Trans-aktion	***Berechtigungsvergabe***							***zeitliche Beschränk-ungen von Berechtig-ungen***	***Berechtigungsentzug bei***
	Events				***Delegation***	***Repository***			
	O	*A*	*Q*	*T*		*T-Sender*	*T-Emp-fänger*		
RQ			x		keine		x	x	Bestellung in Folge der Angebotsaufforderung
QU					keine				
PO	x	x		x	möglich		x		erfolgreiche Zustellung der bestellten Produkte
POC	x	x		x	möglich		x		Trennung der EPC's von Transaktion
ORP					möglich				
IT	x			x	möglich	x	x		erfolgreiche Auslieferung der Transportgüter
DAD					keine				
TS					keine				
IV					keine				
PAO					keine				
PA	x	x		x	möglich		x		erfolgreiche Zustellung der bezahlten Produkte
CR	x	x		x	möglich		x		erfolgreiche Zustellung der bezahlten Produkte

Tabelle 3.2.5 Beziehung zwischen Geschäftstransaktionen und Berechtigungen

Probleme	*Anforderungen*	
Wann ist die Berechtigungsvergabe, auf Basis von Geschäftstransaktionen, möglich?	Eine Berechtigungsvergabe kann für jede Geschäftstransaktion, die in einem TransactionEvent mit dem action-Wert ADD referenziert ist, erfolgen.	Q3
Wann ist der Berechtigungsenzug, auf Basis von Geschäftstransaktionen, möglich?	Ein Berechtigungsentzug kann für jede Geschäftstransaktion, die in einem TransactionEvent mit dem action-Wert DELETE referenziert ist, erfolgen.	Q4
Die Unterscheidung zwischen einem Berechtigungsentzug, der erst nach einer bestimmten Zeitspanne erfolgt, und einem Berechtigungsentzug, der infolge eines Abbruchs der Transaktion erfolgt, ist nicht aus dem TransactionEvent erkennbar.	Die Art des Berechtigungsentzug muss für die Automatisierung entscheidbar sein.	Q5
Die Formulierung der Berechtigungs-vergabe ist von den Sicherheits-bedürfnissen des Unternehmens abhängig.	Einfache Formulierung von komplexen und firmenspezifischen Regeln für die Berechtigungsgenerierung.	Q6

Tabelle 3.2.6 Probleme und Anforderungen von Transaktionen und Berechtigungen

Annahme	
TransactionEvents, die einen Entzug von Berechtigungen auslösen, werden vom System, welches die Geschäftstransaktionen empfängt, generiert und an das EPCIS Capture Interface übermittelt.	A3

Tabelle 3.2.7 Annahme zu Transaktionen und Berechtigungen

3.2.1.5. Auftreten von Geschäftstransaktionen und Berechtigungen

Eine Bestellung wird vom Kunden ausgelöst und dem Bestellsystem des Lieferanten, Händlers bzw. Herstellers, als Transaktion übermittelt. Die Kundenbestellung für Produkte kann vor, während oder nach deren Produktion beim Lieferanten eingehen. Für jede eingehende Kundenbestellung soll der Kunde die Berechtigung erhalten, alle Events im EPCIS-Repository des Lieferanten zu sehen, die im Rahmen seiner Bestellung im Unternehmen anfallen. Im Weiteren werden diese drei Fälle, bezogen auf die genannte Berechtigung, betrachte.

1.Fall: Bestellung von Produkten vor dessen Produktion

Bei der Neuwagenproduktion in der Automobilindustrie ist dieser Fall keine Seltenheit. So werden schon lange vor Produktionsbeginn eines PKW, die Bestellungen von Kunden für diesen Fahrzeugtyp entgegengenommen.

Im Rahmen der auftragsbezogenen Produktion ist folgendes Szenario, siehe Abbildung 3.2.5, denkbar. Eine eingehende Kundenbestellung bezieht sich auf ein noch nicht produziertes Produkt. Das Bestellsystem des Lieferanten erstellt ein entsprechendes TransactionEvent, welches die zukünftige EPC des bestellten Objekts enthält. Diese zukünftige Zuordnung liegt im Schema des EPCIS-Events begründet. Es sagt aus, dass ein neu auftretendes TransactionEvent ein nicht leeres *EpcList*-Element besitzt [EPC07]. Alle EPCIS-

Events, die in Beziehung mit dem bestellten Produkt stehen und nun in das EPCIS-Repository aufgenommen werden, beinhalten die Referenz *http://../po/12345,* die auf die zugehörige Bestellung verweist. Setzt sich ein bestelltes Produkt aus mehreren EPC-Objekten zusammen, können die aggregierten EPC-Objekte die Referenz auf die Bestellung[3] enthalten.

Für die Berechtigungsvergabe bedeutet das, dass Berechtigungen, bezogen auf die Referenz der Bestellung, vergeben werden können, wenn die aggregierten EPC-Objekte ebenfalls diese Referenz besitzen. Der Vorteil der Formulierung von Berechtigungen auf Basis des *bizTransaction*-Elements ist der, dass alle EPC-Objekte, die in Beziehung zur Bestellung stehen berücksichtigt werden ohne dass die konkreten EPCs aller Objekte angegeben werden müssen. Bei einer teilweisen Stornierung ist diese Berechtigungsvergabe, durch Generierung eines Verbotes, für die stornierten Produkte, erforderlich, um die bereits vergebenen Berechtigungen wieder einzuschränken.

Besitzen die aggregierten EPC-Objekte keine Referenz zur Bestellung, erfolgt die Berechtigtsvergabe für konkrete EPCs. Die aggregierten EPC-Objekte sind für diese Berechtigungsvergabe zu ermitteln.

```
<TransactionEvent>
  <eventTime>2006-08-18T08:00:00Z</eventTime>
  <eventTimeZoneOffset >+00:00</eventTimeZoneOffset >
  <bizTransactionList >
    <bizTransaction type="urn:epcglobal:fmcg:bttpo "> http ://.../ po/12345 </bizTransaction >
  </bizTransactionList >
  <epcList>
    <epc>100</epc>
  </epcList>
  <action >ADD</action >
</TransactionEvent>
```

Type	action	EPClist	parentID	bizTransactionList	bizStep	readPoint	eventTime
Transaction	ADD	100		http // .../po/12345			08:00
Object	ADD	1,2,3,4		http // .../po/12345	production	A	10:00
Object	OBSERVE	1		http // .../po/12345	production	B	10:00
Object	OBSERVE	2		http // .../po/12345	production	B	10:00
Object	OBSERVE	3		http // .../po/12345	production	B	11:00
Object	OBSERVE	4		http // .../po/12345	production	B	11:00
Objekt	ADD	100		http // .../po/12345	production	C	13:00
Aggregation	ADD	1,2,3,4	100	http // .../po/12345	production	D	13:25
Object	OBSERVE	100		http // .../po/12345	storing	E	13:30
Object	OBSERVE	100		http // .../po/12345	shipping	F	18:00

Abbildung 3.2.5 Bestellung von Produkten vor deren Produktion

2.Fall: Bestellung von Produkten während deren Produktion

Die Produktbestellung wird, mittels TransactionEvent, während des Produktionsprozesses in das EPCIS-Repository des Lieferanten eingefügt. Alle nun folgenden EPCIS-Events, die im Zusammenhang zur Kundenbestellung stehen, können eine Referenz zu dieser Geschäftstransaktion enthalten.

3) In der Abbildung 3.2.5 sind die optionalen Referenzen grau dargestellt.

EPCIS-Events des EPCIS-Repository können in diesem nicht geändert werden. Damit ist die nachträgliche Referenzierung von EPC-Objekten, die zu einem bestellten Produkt gehören und bereits im EPCIS-Repository integriert sind, nicht möglich. Das bedeutet für die Berechtigungsvergabe, dass die Berechtigungen, die bezogen auf die Referenz von Geschäftstransaktionen vergeben werden können, nicht alle zur Bestellung gehörenden Events umfasst. Um alle zugehörigen Events in Berechtigungen zu integrieren, müssen alle zur Bestellung gehörenden EPCs ermittelt werden.

Eine Bestellung geht, wie in Abbildung 3.2.6 gezeigt, als TransactionEvent ins EPCIS-Repository ein. Zu diesem Zeitpunkt ist nur der EPC des bestellten Objektes und die Referenz zur zugeordneten Bestellung bekannt. Auf diese bekannten Elemente können Berechtigungen vergeben werden. Eine vollständige Berechtigungsvergabe wird aber erst möglich, sobald das AggregationEvent in das EPCIS-Repository integriert wurde. Im Rahmen der Berechtigungsvergabe sind somit neben TransactionEvents ebenfalls AggregationEvents zu berücksichtigen. Die vollständige Berechtigung für das gegebene Beispiel basiert auf der Referenz der Geschäftstransaktion, sowie auf konkrete EPCs. Werden die EPCIS-Events im EPCIS-Repository, die zur Bestellung gehören, nicht nach deren Integration ins EPCIS-Repository mit dieser referenziert, basiert die vollständige Berechtigung ausschließlich auf konkreten EPCs.

```
<TransactionEvent>
  <eventTime>2006-08-18T10:00:00Z</eventTime>
  <eventTimeZoneOffset>+00:00</eventTimeZoneOffset>
  <bizTransactionList>
    <bizTransaction type="urn:epcglobal:fmcg:bttpo ">http://.../po/12345</bizTransaction>
  </bizTransactionList>
  <epcList>
    <epc>100</epc>
  </epcList>
  <action>ADD</action>
</TransactionEvent>
```

Type	action	EPClist	parentID	bizTransactionList	bizStep	readPoint	eventTime
Object	ADD	1,2,3,4			production	A	10:00
Object	OBSERVE	1			production	B	10:00
Object	OBSERVE	2			production	B	10:00
Transaction	ADD	100		http://.../po/12345			10:00
Object	OBSERVE	3		http://.../po/12345	production	B	11:00
Object	OBSERVE	4		http://.../po/12345	production	B	11:00
Objekt	ADD	100		http://.../po/12345	production	C	13:00
Aggregation	ADD	1,2,3,4	100	http://.../po/12345	production	D	13:25
Object	OBSERVE	100		http://.../po/12345	storing	E	13:30
Object	OBSERVE	100		http://.../po/12345	shipping	F	18:00

Abbildung 3.2.6 Bestellung von Produkten während deren Produktion

3.Fall: Bestellung von Produkten nach dessen Produktion

Für das bestellte Produkt sind schon alle Produktionsschritte abgeschlossen, wie in Abbildung 3.2.7 dargestellt. Alle zum Produkt gehörenden Events können rekursiv im EPCIS-Repository abgefragt werden. Die vollständige Berechtigungsvergabe für die Bestellung erfolgt auf Basis der konkreten EPCs.

Wird das bestellte Produkt zum Versand in einen Karton verpackt oder auf einer Palette mit anderen Produkten zusammen gefasst, kann die bereits vergebene Berechtigung, um diese Aggregation erweitert werden. Der Bestellende kann, infolge der Erweiterung, die Palette mit seiner bestellten Ware beobachten.

```
<TransactionEvent>
  <eventTime>2006-08-18T15:00:00Z</eventTime>
  <eventTimeZoneOffset>+00:00</eventTimeZoneOffset>
  <bizTransactionList>
    <bizTransaction type="urn:epcglobal:fmcg:btt:po">http://.../po/12345</bizTransaction>
  </bizTransactionList>
  <epcList>
    <epc>100</epc>
  </epcList>
  <action>ADD</action>
</TransactionEvent>
```

Type	action	EPClist	parentID	bizTransactionList	bizStep	readPoint	eventTime
Object	ADD	1,2,3,4			production	A	10:00
Object	OBSERVE	1			production	B	10:00
Object	OBSERVE	2			production	B	10:00
Object	OBSERVE	3			production	B	11:00
Object	OBSERVE	4			production	B	11:00
Objekt	ADD	100			production	C	13:00
Aggregation	ADD	1,2,3,4	100		production	D	13:25
Object	OBSERVE	100			storing	E	13:30
Transaction	ADD	100		http:// .../po/12345			15:00
Object	OBSERVE	100		http:// .../po/12345	shipping	F	18:00

Abbildung 3.2.7 Bestellung von Produkten nach deren Produktion

Zusammenfassung

Aus den drei zuvor betrachteten Anwendungsfällen ist die Schlussfolgerung zu ziehen, dass die vollständige Berechtigungsvergabe zeitlich vom Auftreten der Bestelltransaktion abhängig ist, wenn es sich bei dem bestellte EPC-Objekt, um ein aggregiertes EPC-Objekt handelt.

Die Formulierung der Berechtigungen kann auf Basis von referenzierten Geschäftstransaktionen bzw. konkreter EPCs beruhen. Die Formulierungsart ist von mehreren Faktoren abhängig, wie in den jeweiligen Anwendungsfällen beschrieben.

Für die vollständige Berechtigungsvergabe im 2. und 3. Fall sind die aggregierten EPC-Objekte zum bestellten Produkt zu ermitteln. Das bestellte Produkt kann sich beispielsweise, wie in Abbildung 3.2.8a gezeigt, aus mehreren Baugruppen zusammensetzen. Es umfasst mehrere Aggregationen von EPC-Objekten. Jedes AggregationEvent stellt eine Aggregationsebene, die eine bestimmte Tiefe besitzt, dar. Für die vollständige Formulierung der Berechtigungen sind alle ECPs, die in Abbildung 3.2.8a dargestellt sind, zu berücksichtigen. Die so formulierten Berechtigungen gewähren dem Handelspartner den unbeschränkten Einblick auf alle Events, die zu dem von ihm bestellten EPC-Objekt gehören. Zur Ermittlung aller EPCs, aus denen sich das bestellte Produkt zusammensetzt, ist eine rekursive Strategie erforderlich.

Die Komplexität der formulierten Berechtigungen, basierend auf konkreten EPCs, nimmt mit jeder zu berücksichtigenden Aggregationsebene zu. Möchte der Eigner des EPCIS-Repository dem Handelspartner nicht alle Details eines bestellten Produktes offenbaren, kann er ausgewählte AggregationEvents und deren untergeordnete Objekte vor dem Handelspartner verbergen (siehe Abbildung 3.2.8c) oder ihm nur bis zu einer bestimmten Aggregationstiefe Einblick in den Produktionsprozess gewähren (siehe Abbildung 3.2.8b).

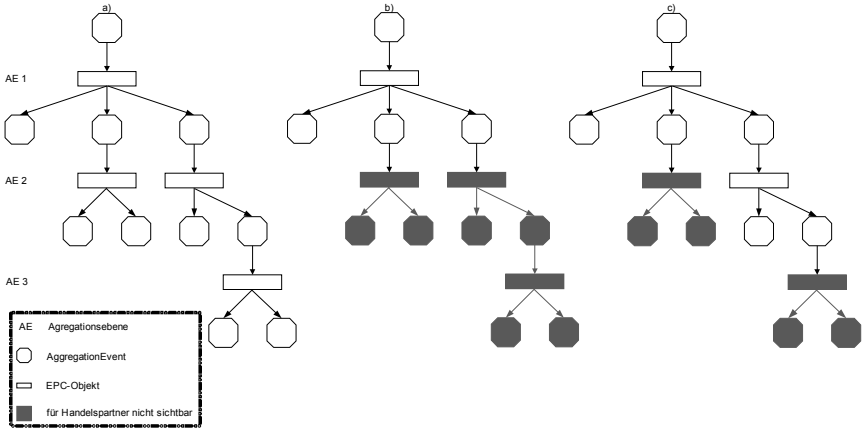

Abbildung 3.2.8 Aggregationsebenen eines EPC-Objekts

Die Transaktionen, Zahlungseingang und Gutschrift, sind ähnlich der Bestellung. Sie können neue Berechtigungen auf EPC-Objekte vergeben oder die bereits im Rahmen der Bestellung gewährten Berechtigung erweitern, in dem weitere Elemente der EPC-Objekte nun für den Kunden sichtbar sind. Bei diesen Transaktionen besteht ebenfalls die zeitliche Abhängigkeit zwischen eingehender Geschäftstransaktion und Berechtigungsformulierung. Anders ist es dagegen bei den Transaktionen RQ und IT, die sich nur auf das EPC-Objekt beziehen, welches im Rahmen dieser Transaktion angegeben wird. Es ist nicht sinnvoll, dem Kunden im Rahmen einer Angebotsaufforderung (*RQ*) Berechtigungen auf beispielsweise die Produktion von noch nicht bestellten Produkten zu vergeben oder dem Transportunternehmen Einzelheiten über die Produktion der zu transportierende Güter freizugeben.

Probleme	***Anforderungen***	
Eine vollständige Berechtigungsvergabe ist nicht immer mit dem TransactionEvent zu formulieren.	Für die Berechtigungsvergabe sind die TransactionEvents und die AggregationEvents zu berücksichtigen.	Q7
Es besteht eine Abhängigkeit, zwischen eingehenden Geschäftstransaktionen und formulierten Berechtigungen, bei aggregierten EPC-Objekten.	Eine Strategie und Methode zur Berücksichtigung aller für die Berechtigung benötigten Events.	Q8

Tabelle 3.2.8 Probleme und Anforderungen vom zeitlichen Auftreten der Transaktion

Annahme	
Die ID der Geschäftstransaktionen im bizTransaction-Element ist für ein EPCIS-Repository eindeutig.	A4

Tabelle 3.2.9 Annahme zum Auftreten von Geschäftstransaktion

3.2.2. Anwendungsfälle

Um weitere Erkenntnisse für die automatische Generierung von Zugriffsberechtigungen zu erhalten, werden konkrete Anwendungsfälle untersucht. Es wird angenommen, dass zwischen einer Transaktion und einem Handelspartner eine bijektive Beziehung besteht, und somit jeder Transaktion ein eindeutiger Useraccount im EPCIS-System zugeordnet werden kann. Eine weitere Annahme ist, das diese Zuordnung im *bizTransaction*-Element in der Form http://transaction.acme.com/Useraccount/Transaction/TransactionID vorliegt.

Annahme	
Jeder Transaktion ist ein eindeutiger Useraccount im EPCIS-System zugeordnet.	A5
Struktur des *bizTransaction*-Element: http://transaction.acme.com/Useraccount/Transaction/TransactionID	A6

Tabelle 3.2.10 Annahme zu Anwendungsfällen

3.2.2.1. Berechtigungsvergabe

In diesem Abschnitt wird der Zusammenhang zwischen TransactionEvent und XACML-PolicySet analysiert, um daraus für die Berechtigungsvergabe Rückschlüsse ziehen zu können. Für diese Analyse werden die Anwendungsfälle aus dem Kapitel 3.2.1.5. herangezogen.

Ausgehend vom TransactionEvent (Abbildung 3.2.9), welches die Bestellung eines Produktes beinhaltet und Auslöser der Berechtigung ist, wurde für jeden der drei Anwendungsfälle das vollständige und endgültige XACML-PolicySet formuliert. Jedes dieser PolicySets gewährt dem Handelspartner Spock den Lesezugriff auf alle Events, die zu seiner Bestellung gehörend. Das PolicySet für den zweiten Anwendungsfall ist identisch mit dem PolicySet des dritten Anwendungsfalls. Die formulierten PolicySets sind im Kapitel 8.7. zu sehen und werden im Weiteren den auslösenden EPCIS-Event gegenübergestellt.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

 <TransactionEvent>

  <eventTime>2006-08-18T11:53:01Z</eventTime>
  <eventTimeZoneOffset>+00:00</eventTimeZoneOffset>

  <bizTransactionList>
   <bizTransaction type="urn:epcglobal:fmcg:btt:po">
    http://transaction.acme.com/spock/po/1234
   </bizTransaction>
  </bizTransactionList>

  <epcList>
   <epc>100</epc>
  </epcList>

  <action>ADD</action>

 </TransactionEvent>
```

Abbildung 3.2.9 TransactionEvent für eine Bestellung

Die erste Gegenüberstellung in der Abbildung 3.2.10 bezieht sich auf den Anwendungsfall, in dem eine Bestellung vor der Produktion der bestellten Produkte erfolgt. Im konkreten Fall ist festgelegt, das die Referenz zu dieser Geschäftstransaktion in allen EPCIS-Events enthalten ist. Das TransactionEvent der Abbildung 3.2.9 und die zugehörige Policy (Kapitel 8.7.1.) werden in diesem Mapping in einer abstrakten Form dargestellt. Das PolicySet des Handelspartners Spock ist in der Abbildung 3.2.10 rechts zu sehen. Es besteht aus einer Policy, die eine Rule enthält. Das *Subjects*-Element der Policy entspricht den *Subjects*-Element des PolicySet. Ebenso ist das *Resources*-Element der Policy gleich dem der Rule. Für beide Elemente wird jeweils eines der beiden ausführlich im Mapping gezeigt. Das TransactionEvent, welches der Auslöser für die Erstellung des XACML-PolicySets ist, wird im Mapping links dargestellt. Das *bizTransaction*-Element ist das einzige Element des Events, welches für das Mapping verwendet wird. Eine direkte Beziehung besteht zwischen diesem Element und dem *predicate*-Element des *Resource*-Elements. Für die eindeutige Identifizierung des PolicySets, der Policy und der Rule wird das *bizTransaction*-Element, welches den Useraccount, die Geschäftstransaktion und deren ID umfasst, verwendet. Die Bezeichnung des Subjekts, für das die Berechtigung gelten soll, wird mit den Useraccount, der im *bizTransaction*-Element enthalten ist, bezeichnet.

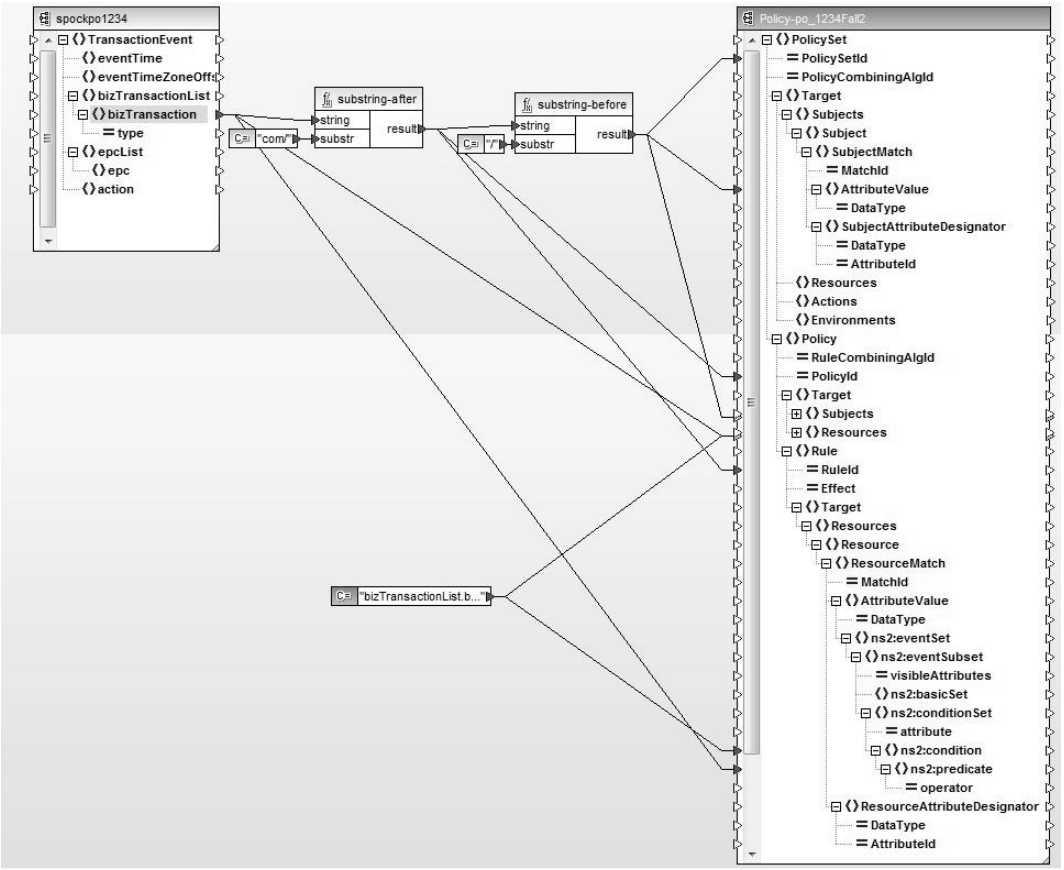

Abbildung 3.2.10 Mapping zwischen TransactionEvent und PolicySet 1

Um das XACML-PolicySet für dieses TransactionEvent zu generieren, ist ein Template zu erstellen, welches die in der Abbildung 3.2.10 gezeigten Abhängigkeiten und die Struktur des PolicySets enthält. In diesem Template sind alle

Elemente und Attribute des PolicySets, deren Werte nicht vom TransactionEvent abhängig sind, zu spezifizieren. Des Weiteren ist die Berücksichtigung von Ausnahmen und Sonderfällen wichtig, die für ausgewählte Handelspartner oder Produktklassen für die Berechtigungsvergabe existieren können. Mit einem so definierten Template kann für jedes TransactionEvent ein PolicySet erzeugt werden. Dieses PolicySet umfasst die Berechtigung des Handelspartners bezogen auf die EPCIS-Events, die im Zusammenhang mit der von ihm getätigten Geschäftstransaktion stehen.

Für einen Handelspartner, der den Transport der bestellten Ware übernimmt, sollen nicht die gleichen Berechtigungen, wie für einen Handelspartner, der eine Bestellung auslöst, gelten. Die Generierung der verschiedenen Berechtigungen kann mit einem Template-Dokument erfolgen. Es ist ebenfalls möglich für jede Transaktionsart ein separates Template-Dokument zu verwenden. Bei Änderungen in der Berechtigungsvergabe für eine Geschäftstransaktion ist das entsprechende Template-Dokument auszutauschen. Diese Variante ist gegenüber einem globalen Template-Dokument übersichtlicher und die Selektion der Transaktionsart findet im Vorfeld statt.

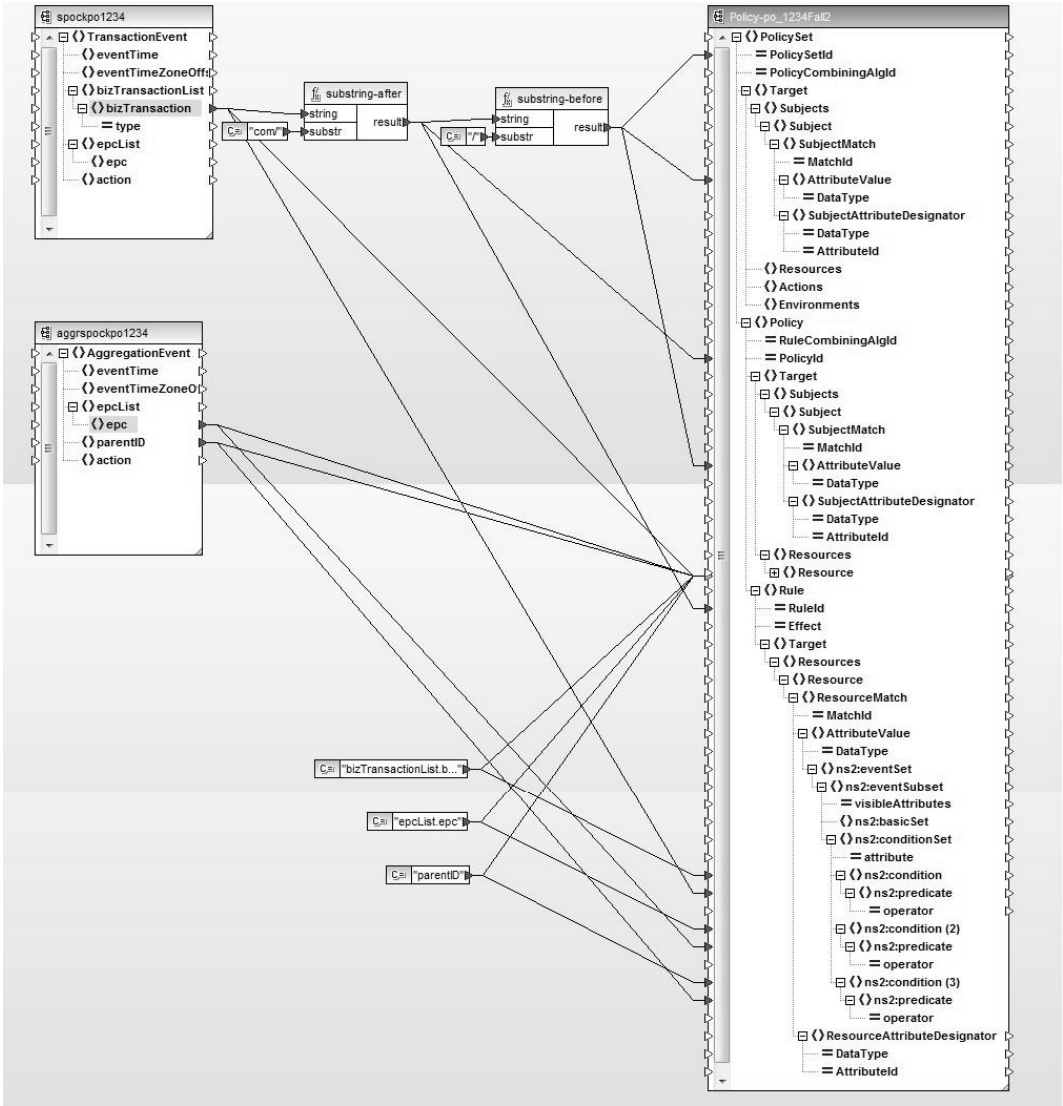

Abbildung 3.2.11 Mapping zwischen TransactionEvent und PolicySet 2

Die zweite Gegenüberstellung, die in Abbildung 3.2.11 gezeigt wird, beinhaltet das PolicySet, wie dieser für die Bestellung von Produkten während und nach deren Produktion zu erstellen ist. Das Mapping ist äquivalent dem in der Abbildung 3.2.10. Es wurde um das AggregationEvent erweitert. Das AggregationEvent ist notwendig, um alle Events bezogen auf die Geschäftstransaktion zu berücksichtigen. Die Formulierung des PolicySets, für die Bestellung von Produkten während deren Produktion, erfolgt in zwei Phasen, da das AggregationEvent zum Eingang des TransactionEvents noch nicht bekannt ist. Das PolicySet, das im Rahmen des TransactionEvents erstellt wurde, ist um das ConditionSet, welches sich auf die Elementwerte des AggregationEvents bezieht, zu erweitert.

Wie die Erweiterung schon bestehender Berechtigungen vorgenommen wird und ob für die Berechtigungserweiterungen das vollständige PolicySet generiert wird, ist von der verwendeten Integrationsstrategie abhängig.

Probleme	***Anforderungen***	
Wie erfolgt die Integration bzw. die Aktualisierung von Berechtigungen?	Eine effiziente und effektive Integrationsstrategie ist notwendig.	Q9
Die Berechtigungsformulierung, sowie die Berücksichtigung von Ausnahmen und Sonderfällen, soll optimal erfolgen.	Eine effektive und ausreichende Formulierung für Berechtigungen, Ausnahmen und Sonderfälle ist zu realisieren.	Q10
Die Struktur von PolicySets ist variabel.	Für die Automatisierung ist eine konkrete Struktur der zu generierenden PolicySets zu verwenden.	Q11

Tabelle 3.2.11 Probleme und Anforderungen zu Anwendungsfällen

3.2.2.2. Berechtigungsentzug

Ein Berechtigungsentzug erfolgt aus verschiedenen Gründen. Beispielsweise durch Beendigung von Transaktionen, Stornierung von bestellten Produkten oder Stornierung eines Transportauftrages. Die konkrete Festlegung ist vom besitzenden Unternehmen des EPCIS-Repositories zu treffen. In dieser Arbeit werden Berechtigungen, wie in Tabelle 3.2.5 beschrieben, entzogen. Der Entzug einer vergebenen Berechtigung ist mit Veränderungen am PolicySet verbunden, die die vollständige oder teilweise Löschung von Berechtigungen vorsieht. Diese Entscheidung ist vom TransactionEvent abhängig. Enthält das TransactionEvent alle EPCs des zugehörigen, inversen TransactionEvents oder ist das *EpcList*-Element des TransactionEvents leer, wird die Berechtigung vollständig gelöscht. Anderenfalls ist die vergebene Berechtigung so zu verändern, dass die vom TransactionEvent getrennten EPC-Objekte, nicht mehr in der Berechtigung enthalten sind.

Ein TransactionEvent, welches den Entzug von Berechtigungen auslöst, ist mit dem *action*-Wert DELETE (Abbildung 3.2.12) gekennzeichnet. Es besitzt ein leeres *EpcList*-Element, damit ist die Bestellung beendet und alle in Folge der Bestellung vergebenen Berechtigungen werden entzogen. Somit sind diese Berechtigungen aus dem PolicySet des Bestellenden zu löschen. Dabei ist es wichtig, den Bezug zwischen der Bestellung und den zugehörigen Policies und

Rules der Bestellung zu kennen, um diese gezielt aus dem PolicySet des Users zu entfernen.

Der teilweise Entzug von Berechtigungen ist bei Policies, die Berechtigungen bezogen auf das *bizTransaction*-Element beinhalten, durch das Entfernen von Policy, Rule oder conditonSet nicht zu realisieren. Beispielsweise eine Bestellung eines Handelspartners, die zwei noch nicht produzierte Produkte umfasst. Für den Handelspartner wird ein PolicySet, wie in 8.7.1., erstellt, welches ihm gewährt alle Events, die seine bestellten Produkte betreffen, zu sehen. Er entscheidet sich nach einer bestimmten Zeit eines der Produkt zu stornieren. Die Produkte sind produziert und befinden sich im Lager des produzierenden Unternehmens, als die Stornierung eingeht. In Folge der Stornierung ist die Berechtigung, die für die ursprüngliche Bestellung vergeben wurde, entweder komplett zu löschen und für die verbleibende Produktbestellung ist eine neue Berechtigung zu erstellen, die unabhängig vom *bizTransaction*-Element ist, oder die bereits vergebene Berechtigung ist zu beschränken, so dass dem Handelspartner der Zugriff auf das stornierte Produkt verwehrt wird.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

 <TransactionEvent>
  <eventTime>2006-08-18T11:53:01Z</eventTime>
  <eventTimeZoneOffset>+00:00</eventTimeZoneOffset>

  <bizTransactionList>
   <bizTransaction type="urn:epcglobal:fmcg:btt:po">
    http://transaction.acme.com/spock/po/1234
   </bizTransaction>
  </bizTransactionList>

  <epcList/>

  <action>DELETE</action>
 </TransactionEvent>
```

Abbildung 3.2.12 TransactionEvent für die Bestellungsbeendigung

Probleme	***Anforderungen***	
Der Zusammenhang zwischen Geschäfts-transaktionen und Berechtigungen ist wichtig.	Eine eindeutige Zuordnung von Geschäftstransaktion und Berechtigungen ist notwendig.	Q12
Wie erfolgt die Löschung von Berechtigungen?	Eine effiziente und effektive Deinte-grationsstrategie für Berechtigungen ist notwendig, die den teilweisen, sowie vollständigen Berechtigungs-entzug realisiert.	Q13

Tabelle 3.2.12 Probleme und Anforderungen zum Berechtigungsentzug

3.2.3. Bestellungsänderung

Eine Bestellungsänderung ist die Erweiterung, die teilweise Stornierung einer bereits getätigten Bestellung oder beides. Sie kann aber auch die vollständige Stornierung einer Bestellung implizieren.

Ein Handelspartner sendet beispielsweise, wie im Szenario 3.2.1.1 beschrieben, einem anderen Handelspartner eine Bestellungsänderung zu. Diese Nachricht

wird vom Bestellsystem des Empfängers entgegengenommen und vom System in zwei TransactionEvents überführt, die in das EPCIS-Repository über das EPCIS Capture Interface integriert werden.

Zum Einen wird ein TransactionEvent erzeugt, welches die Bestellungsänderung kennzeichnet. Es enhält die EPCs, um die die bisherige Bestellung erweitert wird, und den *action*-Wert ADD. Durch diesen *action*-Wert wird angezeigt, das neue Berechtigungen zu vergeben sind. Die neuformulierten Berechtigungen sind in die schon bestehenden Berechtigungen zu integrieren. Für die Integration ist eine Strategie erforderlich, die entscheidet, wie die neu erstellten Berechtigungen einzufügen sind. Von der Integrationsstrategie ist die Form, der zu formulierenden Berechtigungen, abhängig, um eine effiziente und ausreichende Berechtigungsformulierung vorzunehmen.

Zum Zweiten ist ein TransactionEvent zu generieren, welches die Stornierung von EPC-Teilmengen anzeigt, bei denen die Bestellungsänderung im *bizTransaction*-Element referenziert ist. Des Weiteren enthält das TransactionEvent die EPCs, die von der Bestellung getrennt werden, sowie das action-Element mit dem Wert DELETE. Mittels dem Wert DELETE wird signalisiert, das die bereits vergebenen Berechtigungen für die entsprechenden EPCs wieder zu entziehen sind. Um mit den entsprechenden Berechtigungen arbeiten zu können, ist es wichtig den Bezug, zwischen Geschäftstransaktionen und den Policies bzw. Rules dieser Geschäftstransaktionen zu kennen. Für die Veränderung der schon vorhandenen Berechtigungen ist eine effektive und effiziente Lösung zu finden, um die Kosten für die Bearbeitung des PolicySets möglichst gering zu halten.

3.2.4. Zusammenfassung

In der Tabelle 3.2.13 sind die Anforderungen, die im Rahmen der Analyse von Geschäftstransaktionen und den konkreten Anwendungsfällen ermittelt wurden, zusammengefasst. Die ermittelten Anforderungen wurden in funktionale und nicht funktionale Anforderungen unterteilt. Des Weiteren erfolgt eine konzeptionelle Einordnung in die Bereiche, Filterung von zu berücksichtigenden EPCIS-Events (EEF), Berechtigungstransformation (XE), Transformationsregeln (XS), Aktualisierung von Berechtigungen (PU), formulierte Berechtigungen (PS), EPC-Ermittlung (EF) und Anforderungen an das EPCIS-System (ES).

Diese Bereiche, ausgenommen das EPCIS-System (kurz ES), repräsentieren im Konzept der PIE Komponenten bzw. Elemente. Die Filterung von zu berücksichtigenden EPCIS-Events wird in der PIE-Komponente EPCIS-EventFiltering (kurz EEF) vorgenommen. Die Umwandlung der EPCIS-Events in deren entsprechende Berechtigungen erfolgt durch die XSLT-Engine (kurz XE). Die dafür benötigten Transformationsregeln sind in XSLT-Stylesheets (kurz XS) definiert. Der Policy-Updater (kurz PU) setzt die Berechtigungsänderungen mittels Aktualisierung der bestehenden Berechtigungen im PAP durch. Die Berechtigungen sind in Form von PolicySets (kurz PS) im PAP gespeichert. Um relevante EPCs eines EPC-Objektes bei der Berechtigungsformulierung zu berücksichtigen, erfolgt die EPC-Ermittlung während der Berechtigungstransformation. Die EPC-Ermittlung wird durch den EPC-Finder (kurz EF) realisiert.

Diese und weitere Komponenten und Elemente der PIE werden im nachfolgenden Kapitel vorgestellt.

	Anforderung		*Bereich*
	funktionale	***Nicht funktionale***	
Q1		Eine eindeutige Identifizierung von Geschäftstransaktionen im *bizTransaction*-Element.	ES
Q2	Jede referenzierte Geschäftstransaktion eines TransactionEvents ist bei der Formulierung von Berechtigungen zu prüfen.		EEF
Q3	Eine Berechtigungsvergabe kann für jede Geschäftstransaktion, die in einem TransactionEvent mit dem action-Wert ADD referenziert ist, erfolgen.		EEF
Q4	Ein Berechtigungsentzug kann für jede Geschäftstransaktion, die in einem TransactionEvent mit dem action-Wert DELETE referenziert ist, erfolgen.		EEF
Q5	Die Art des Berechtigungsentzug muss für die Automatisierung entscheidbar sein.		EEF, PU
Q6		Einfache Formulierung von komplexen und firmenspezifischen Regeln für die Berechtigungsgenerierung.	XS
Q7	Für die Berechtigungsvergabe sind die TransactionEvents und die AggregationEvents zu berücksichtigen.		EEF, XS
Q8	Eine Strategie und Methode zur Berücksichtigung aller für die Berechtigung benötigten EPCIS-Events.		EF
Q9	Eine effiziente und effektive Integrationsstrategie ist notwendig.		PU
Q10		Eine effektive und ausreichende Formulierung für Berechtigungen, Ausnahmen und Sonderfälle ist zu realisieren.	XS
Q11		Für die Automatisierung ist eine konkrete Struktur der zu generierenden PolicySets zu verwenden.	PS
Q12		Eine eindeutige Zuordnung von Geschäftstransaktionen und Berechtigungen ist notwendig.	XS, PS
Q13	Eine effiziente und effektive Deintegrationsstrategie für Berechtigungen ist notwendig, die denteilweisen, sowie den vollständigen Berechtigungsentzug realisiert.		PU

Tabelle 3.2.13 Anforderungseinordnung

4. Konzeption der Policy Instantiation Engine

Die Policy Instantiation Engine (kurz PIE) ist eine eigenständige Engine, die die automatische Generierung von Zugriffsberechtigungen in aidXACML, basierend auf Geschäftstransaktionen, die in das EPCIS-Repository integriert werden, vornimmt. Die Formulierung der Zugriffsberechtigungen erfolgt mittels XSLT. Erstellte Berechtigungen werden, je nach Updatestrategie, in den Policy Access Point integriert.

4.1. Policy Instantiation Engine (PIE)

Die PIE Architektur, in Abbildung 4.1.1 dargestellt, ist eine ereignisgesteuerte Systemarchitektur (siehe Kapitel 2.4.1.1.), die von den EPCIS-Events, welche über das EPCIS Capture Interface in das EPCIS-Repository gespeichert werden, gesteuert wird. Die Reaktion der PIE auf EPCIS-Events, die mittels EPCIS-Dokument (siehe Kapitel 4.2.) an die Engine übermittelt werden (a), erfolgt durch den EPCIS-EventFilter.

Der EPCIS-EventFilter (siehe Kapitel 4.6.) entscheidet für jedes eingehende EPCIS-Event, ob dieses im Rahmen der Zugriffskontrolle berücksichtigt wird. Für zu berücksichtigende EPCIS-Events löst der EPCIS-EventFilter entweder die Formulierung einer Zugriffsberechtigung (c) oder die Löschung einer Zugriffsberechtigung (d) aus. Das Event Handling des EPCIS-EventFilters beruht auf Daten, die das EPCIS-Event und die PIE-Datenbank beinhalten.

Die XSLT-Engine (siehe Kapitel 4.7.) nimmt die Formulierung der Zugriffsberechtigungen für EPCIS-Events mittels XSLT-Stylesheets (siehe Kapitel 4.4.) vor. Für jede Geschäftstransaktion ist ein separates XSLT-Stylesheet vorgesehen. Über das XSLT Control Interface können die XSLT-Stylesheets durch den XSLT-Manager in die PIE integriert, gelöscht bzw. aktualisiert (l) werden. Der XSLT-Manager synchronisiert den Bestand der XSLT-Stylesheets, für Geschäftstransaktionen, gegenüber der PIE-Datenbank (m). Die XSLT-Engine erhält das für die XSL-Transformation benötigte XSLT-Stylesheet (siehe Kapitel 4.4.) für EPCIS-Events vom XSLT-Manager (k). Während der Transformation vom EPCIS-Event zum PIE-PolicySet, welches ein optimiertes PolicySet repräsentiert (siehe Kapitel 4.3.3.), werden alle EPCs durch den EPC-Finder (e), die für die Formulierung von Berechtigungen relevant sind, ermittelt (f). Für die EPC-Ermittlung verwendet er die PIE-Datenbank.

Der EPC-Finder (siehe Kapitel 4.8.) enthält den Algorithmus zur EPC-Ermittlung und Methoden, die der XSLT-Parser bei der XSL-Transformation für die Integration relevanter EPCs ausführt. Die generierten PIE-PolicySets werden an den PIE-Handler weitergeleitet (g).

Der PIE-Handler (siehe Kapitel 4.9.) realisiert die Integration, sowie die Löschung von Berechtigungen über den Policy-Updater je nach Aktualisierungsstrategie. Bei der Umsetzung der permanenten Aktualisierung erfolgt keine Zwischenspeicherung der generierten PIE-PolicySets (n). Bei nicht permanenten Aktualisierungsstrategien erfolgt die Zwischenspeicherung der generierten PIE-PolicySets. Über das Query Rights Interface der PIE kann eine Aktualisierung der Zugriffsberechtigungen für einen Nutzer ausgelöst werden. Der PIE-Finder leitet die Aktualisierungsanfrage an den PIE-Handler weiter (o). Der PIE-Handler ermittelt (h) die betreffenden PIE-PolicySets, die zur Anfrage gehören, über die PIE-Datenbank, und löst die Integration für diese PolicySets über den Policy-

Updater aus (i).

Der Policy-Updater (siehe Kapitel 4.10.) setzt die Integration, die Aktualisierung und die Löschung von PolicySets, Policies und Rules am Poilcy Access Point (PAP), über den PAP-Client, durch (j).

Die PIE-Datenbank (siehe Kapitel 4.5.) enthält die für die PIE wichtigen Daten, um beispielsweise die Entscheidung zur Generierung von Berechtigungen zu treffen oder den vollständigen Berechtigungsentzug zu realisieren. Die Datenbank erhält diese relevanten Daten von den Komponenten: EPCIS-Eventfilter, XSLT-Manager und PIE-Handler.

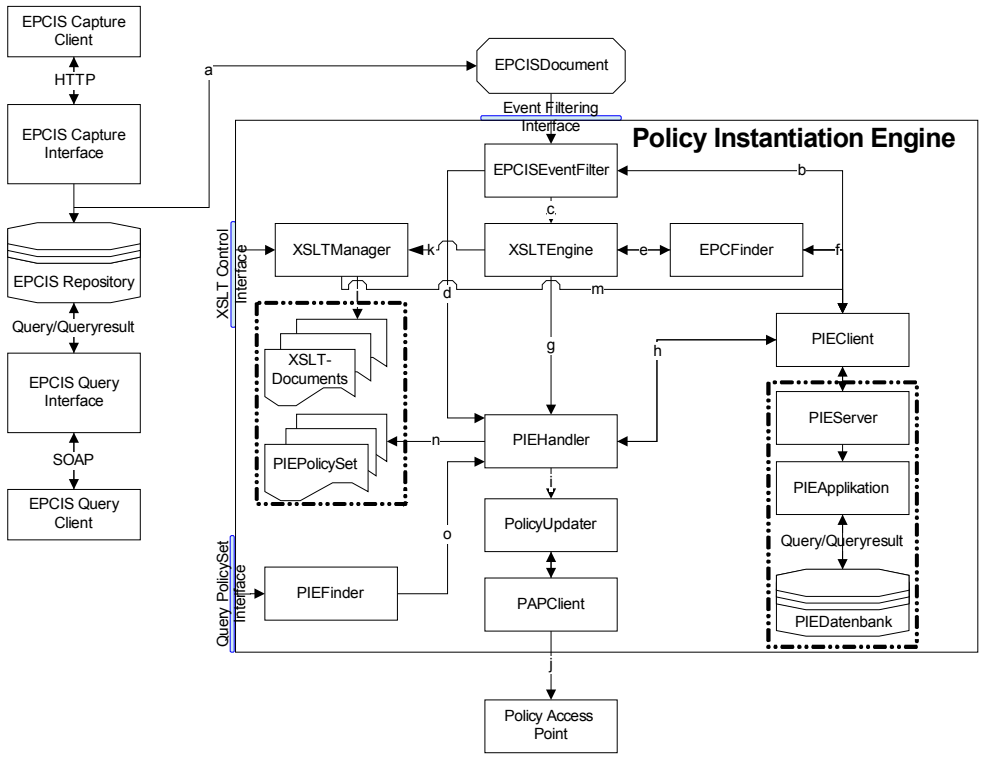

Abbildung 4.1.1 Policy Instantiation Engine (PIE)

4.2. EPCIS-Dokument

Das EPCIS-Dokument ist ein standardisierter XML-Nachrichtentyp, mit dem die EPCIS-Events über das Event Filtering Interface an die Policy Instantiation Engine übermittelt werden. Er besteht aus dem *EPCISBody* und kann optional einen *EPCISHeader* beinhalten. Der EPCISBody umfasst eine Eventliste, die die unterschiedlichen EPCIS-Eventtypen, entsprechend des Kapitel 2.1.2.2., strukturiert enthält. Das Schema des EPCIS-Dokumentes ist im Anhang 8.3. abgebildete.

Die Abbildung 4.2.1 zeigt ein einfaches EPCIS-Dokument, welches mittels EPCIS Capture Client von Fosstrak erstellt und an das EPCIS Capture Interface gesendet wurde. Das in der Eventliste enthaltene TransactionEvent wird geprüft und in das EPCIS-Repository gespeichert. Nach der Prüfung der EPCIS-Events wird das entsprechende EPCIS-Dokument an die PIE gesendet, um für diese EPCIS-Events die automatische Berechtigungsformulierung über die PIE zu realisieren.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
 <epcis:EPCISDocument
  xmlns:epcis="urn:epcglobal:epcis:xsd:1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  creationDate="2008-12-05T15:16:59.987+01:00" schemaVersion="1.0">

  <EPCISBody>
   <EventList>
    <TransactionEvent>
     <eventTime>2006-09-20T07:53:01Z</eventTime>
     <eventTimeZoneOffset>+00:00</eventTimeZoneOffset>

     <bizTransactionList>
      <bizTransaction
       type="urn:fosstrak:demo:biztrans:fmcg:QApassed">
       http://demo.fosstrak.com/QAtracker/po/q3432q4324</bizTransaction>
     </bizTransactionList>

     <epcList>
      <epc>urn:epc:id:sgtin:0057000.123780.7788</epc>
     </epcList>

     <action>ADD</action>
    </TransactionEvent>
   </EventList>
  </EPCISBody>
 </epcis:EPCISDocument>
```

Abbildung 4.2.1 Simple EPCIS Document vom EPCIS Capture Client generiert

4.3. PolicySet

Die aidXACML-Policysprache (siehe Kapitel 2.2.2.) wird zur Definition von Berechtigungen für das EPCIS-Repository-Zugriffskontrollsystem verwendet. Die hierarchische Struktur von XML-PolicySets ist in Abbildung 2.2.2 dargestellt. Sie zeigt, dass die strukturelle Definition eines PolicySets sehr variabel ist, und es somit verschiedene Möglichkeiten gibt, Berechtigungen zu strukturieren.

Für die automatische Berechtigungsformulierung ist ein konkret spezifizierter Aufbau von PolicySets notwendig, um effizient Berechtigungen für Nutzer integrieren, aktualisieren und löschen zu können. Ein Strukturdefinition wurde in diesem Zusammenhang für Systeme, die ein globales PolicySet oder userbezogenes PolicySet zur Speicherung von Berechtigungen verwenden, erstellt.

Des Weiteren hat sich während der Entwicklung des Konzeptes herausgestellt, das es aus Performenzgründen effizienter ist ein optimiertes PolicySet innerhalb der PIE als Informationsträger einzusetzen, genannt PIE-PolicySet. Dieses PIE-PolicySet wird zur Integration und zur Aktualisierung von Berechtigungen in den Policy Access Point verwendet.

4.3.1. Globale PolicySet

Ein globales PolicySet (siehe Abbildung 4.3.1) besteht aus zwei PolicySets. Zum Einen aus dem PolicySet für den „Superuser“ und zum Anderen aus dem PolicySet für „Alluser“.

Das PolicySet des Superuser ist für den Administrator, oder für Applikationen die den vollen Zugriff auf das EPCIS-Repository benötigen, vorgesehen. Die PIE könnte somit auch die EPC-Ermittlung über das EPCIS-Repository vornehmen.

Im Kapitel 4.5. wird auf die Vor- und Nachteile der Nutzung des EPCIS-Repository als Informationsquelle für die PIE eingegangen. Das PolicySet des Superusers besteht aus dem Target-Element, welches das Subjekt spezifiziert, und einer Policy. Diese Policy umfasst ein Target-Element, welches ebenfalls das Subjekt spezifiziert, und eine Rule, die mittels Target-Element die Berechtigung auf das EPCIS-Repository definiert.

Das PolicySet (Alluser) enthält ein leeres Target und die PolicySets aller User. Jedem User ist genau ein PolicySet zugeordnet. Das PolicySet für einen User wird mit den Useraccount, der eineindeutig jedem User zugeordnet ist, bezeichnet. Das PolicySet umfasst das Target-Element, welches das Subjekt[4] spezifiziert, sowie mindestens eine Policy. Eine Policy enthält die Berechtigungs-rules, die für eine vom User getätigte Transaktion (z.B. eine Bestellung) erstellt worden sind. Somit setzt sich die PolicyID aus dem Transactionstyp und der TransactionsID zusammen. Das Target-Element der Policy spezifiziert das Subjekt und die Ressourcen. Das Resources-Element der Policy ist die Vereinungungsmenge aller Resourse-Elemente, die in den Rules der Policy definiert sind. Eine Policy besitzt mindestens eine Rule, die mittels Target-Element die Berechtigungen auf das EPCIS-Repository definiert. Die RuleID ist die Erweiterung der PolicyID um den Timestamp, der die Eventtime des Events, für welches die Rule erstellt worden ist, angibt.

Für die automatische Generierung von Zugriffsrechten ist die Verwendung eines globalen PolicySet-Dokuments zur Speicherung von Zugriffsberechtigungen nicht geeignet. Da für Änderungen an einem PolicySet, wie die Integration einer neuen Policy oder Rule, das gesamte Dokument geladen, bearbeitet und anschließend zurück zu speichern ist.

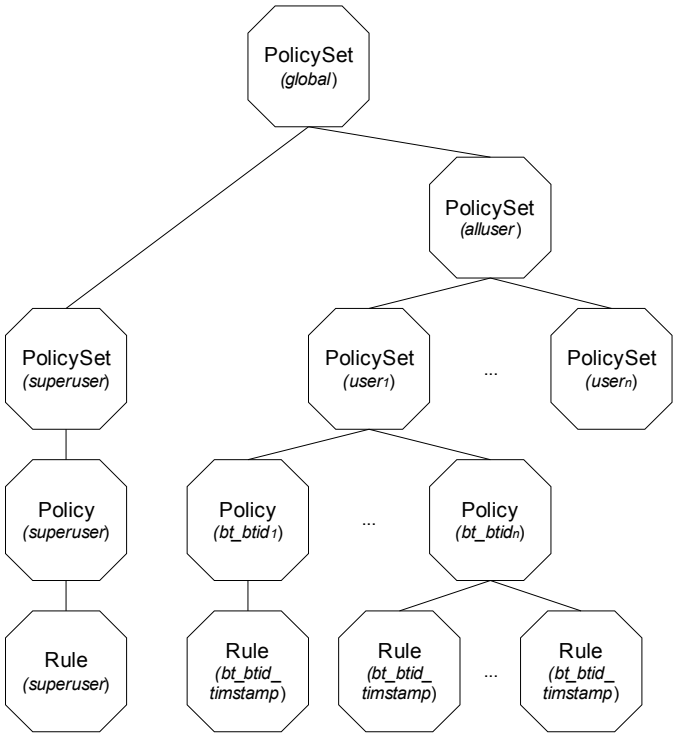

Abbildung 4.3.1 Global PolicySet Struktur

4) Das Subjekt entspricht den eindeutigen Useraccount.

4.3.2. PAP-PolicySet

Der Policy Access Point (PAP), der die Zugriffsberechtigungen für das EPCIS-Repository beinhaltet, realisiert die userbezogene Speicherung von PolicySets. Jeder User, der berechtigt ist auf das EPCIS-Repository zu zugreifen, besitzt ein separates PolicySet. Das PolicySet ist entweder so strukturiert, dass jede Policy die Zugriffsberechtigungen für eine Geschäftstransaktion spezifiziert (Abbildung 4.3.2a), oder das jede Rule die Zugriffsberechtigungen für eine Geschäfts-transaktion definiert (Abbildung 4.3.2b).

Die Struktur des PolicySets, in Abbildung 4.3.2a entspricht dem, im vorangegangenen Abschnitt beschriebenen, strukturellen Aufbau der PolicySets für User.

Das PolicySet, das in Abbildung Abbildung 4.3.2b dargestellt ist, umfasst genau eine Policy. Diese Policy wird durch den gleichen ID-Wert bezeichnet, wie das PolicySet der Policy. Das Target-Element der Policy ist leer. Alle Rules der Policy sind auszuwerten, um die Zugriffsentscheidung für Daten des EPCIS-Repositories zu treffen. Eine Policy enthält mindestens eine Berechtigungsrule. Jede Rule repräsentiert eine vom User getätigte Geschäftstransaktion für die Zugriffsberechtigungen vergeben worden sind. Somit setzt sich die RuleID aus dem Transactionstyp und der TransactionsID zusammen. Das Target-Element der Rule definiert die Zugriffsberechtigungen auf das EPCIS-Repository für die jeweilige Transaktion.

Im Rahmen der automatischen Formulierung von Zugriffsrechten, bringt die userbezogene Speicherung der PolicySets die Verkürzung der Bearbeitungszeit und die Kostensenkung für Aktualisierungen mit sich, da gezielt die zu verändernden PolicySets aus der PAP geladen, bearbeitet und zurück-gespeichert werden. Für die Bearbeitung der einzelnen PolicySets werden somit weniger Ressourcen benötigt, als bei der Bearbeitung eines globalen PolicySet-Dokumentes. Angenommen, ein globales PolicySet-Dokument enthält *n* userbezogene PolicySets, so werden für die Bearbeitung eines PolicySets *n* mal so viele Ressourcen wie für die direkte Bearbeitung eines userbezogenen PolicySets, gebraucht. Neu erstellte PolicySets werden bei der userbezogenen Speicherung von PolicySets direkt in den PAP integriert.

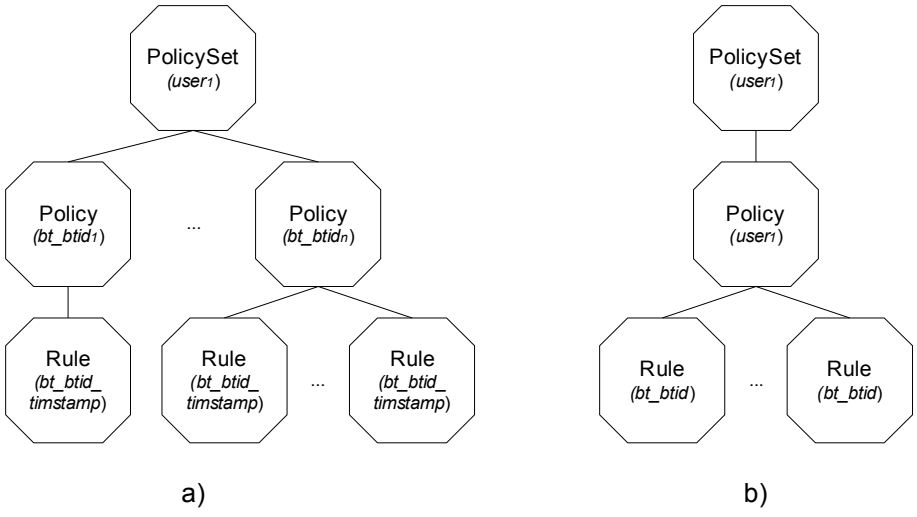

Abbildung 4.3.2 User PolicySet Strukturen

Die Tabelle 4.3.1 zeigt die beiden userbezogenen PolicySet-Definitionen im Vergleich zueinander. Die Entscheidung, welche der beider PolicySets im Rahmen der PIE umgesetzt werden, ist vom Gesamtsystem und den Unternehmenskriterien abhängig.

Im Rahmen der späteren Implementierung, wird das in der Abbildung 4.3.2b dargestellte PolicySet im PAP verwendet. Bei der Anwendung des anderen PolicySets sind entsprechende Änderungen am Policy-Updater und für die Strategie der teilweisen Beendigung von Transaktionen vorzunehmen.

Kriterien	***Abbildung 4.3.2 a***	***Abbildung 4.3.2 b***
Speicherplatz	benötigt mehr Speicherplatz als b, da das Policy-Target nicht leer und pro Transaktion eine Policy erstellt wird	ist gering, da nur eine Policy pro PolicySet erstellt und das Target-Element für Policies leer ist
Zugriffs-entscheidung	nur die zutreffenden Policies und deren Rules werden ausgewertet	alle Rules werden auszuwerten
Integration einer Verbotsrule	ist möglich, da mehrere Rules zu einer Transaktion zugeordnet werden können	ist nicht möglich, da sich nur eine Rule auf eine Transaktion bezieht Verbote, können durch Ruleerweiterungen (mittels eal:deniedEvents) oder durch die Neuformulierung der Rule, bezogen auf die EPCs, erfolgen
Berechtigungs-vergabe	Berechtigungen bezogen auf den biztransaction-Wert können generiert werden, da Integration von Berechtig-ungsverboten zu Transaktion möglich sind Erweiterung bereits vorhandener Berechtigungen durch Integation neuer Rules in die Policy	Berechtigungen sind, bezogen auf den biztransaction-Wert nicht erlaubt Erweiterung vorhandener Berechtigungen durch die Integration generierter EventSubsets in das EventSet der Rule
Berechtigungs-entzug	vollständige Löschung durch das Entfernen der entsprechenden Policy teilweise Löschung durch Generierung und Integration von Verbotsrules	vollständige Löschung durch das Entfernen der entsprechenden Rule teilweise Löschung durch die Neugenerierung der entsprechenden Rule

Tabelle 4.3.1 Gegenüberstellung der userbezogenen PolicySets

4.3.3. PIE-PolicySet

Das PIE-PolicySet (siehe Abbildung 4.3.3) ist ein optimiertes PolicySet. Es enthält alle Elemente und deren Werte, die zur Erstellung eines PAP-PolicySets, einer Policy bzw. einer Rule für ein PAP-PolicySet benötigt werden. Doppelungen der Target-Element- Inhalte sind nicht erlaubt. Die ID-Attribute des PIE-PolicySet sind an das zu verwendende PAP-PolicySet anzupassen.

Erfolgt die Speicherung des PolicySets im PAP, wie in Abbildung 4.3.2a, hat die Verwendung des PIE-PolicySets für generierte Berechtigungen den Vorteil, dass die Kosten für die Transformationszeit, gegenüber der vollständigen Generierung des PolicySets reduziert werden. Für die nicht permanente Aktualisierungs-strategie des Policy-Handlers ist die Reduzierung des Speicherbedarfs ein weiterer Vorteil für die generierten PolicySets. Die Entscheidung, ob ein PolicySet, eine Policy oder eine Rule für die Berechtigung einer Geschäftstransaktion des Users zu formulieren ist, muss nicht vor der XSL-Transformation, die durch die XSLT-Engine vorgenommen wird, getroffen werden. Sie wird vom Policy-Updater erst vor der Berechtigungsintegration in die PAP getroffen.

Werden PolicySets, wie in Abbildung 4.3.2b im PAP gespeichert, erfolgt die direkte Speicherung der generierten PolicySets der User, für die noch kein PolicySet im PAP gespeichert ist.

Für Unternehmen, die die permanente Aktualisierung des PAP umsetzen und einen stetigen Kundenwechsel bzw. Handelspartnerwechsel zu verzeichnen haben, ist die userbezogene Speicherung, in der Form von Abbildung 4.3.2b, zu bevorzugen.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<PolicySet
 xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
 xmlns:eal="http://inf.tu-dresden.de/eal/1.0/schema"
 PolicySetId="useraccount"
 PolicyCombiningAlgId="...">

 <Target>
  <Subjects>
   <Subject>
     ...
   </Subject>
  </Subjects>
 </Target>

 <Policy
  PolicyId="(wie PAPPolicySet)"
  RuleCombiningAlgId="...">
  <Target/>

  <Rule Effect="..." RuleId="(wie PAPPolicySet)">
   <Target>
    <Resources>
     <Resource>
      ...
     </Resource>
    </Resources>
   </Target>
  </Rule>
 </Policy>
</PolicySet>
```

Abbildung 4.3.3 PIE-PolicySet

4.4. XSLT-Stylesheets zur Formulierung von Berechtigungen

Das XSLT-Stylesheet enthält die Transformationsregeln für den XSLT-Parser, der anhand dieser Regeln die Umwandlung von EPCIS-Events in PIE-PolicySets realisiert.

Für jeden Typ von Transaktionen, für die eine Berechtigung zu formulieren ist, wird ein separates XSLT-Stylesheet benötigt, welches die entsprechenden Regeln für die Transformation beinhaltet. Diese eineindeutige Zuordnung ist darin begründet, dass die Berechtigungsformulierung von der Transaktionsart abhängig ist, wie in Kapitel 3.2.1.4. gezeigt. Für das Konzept der PIE bedeutet das, dass das XSLT-Stylesheet für eine Geschäftstransaktion unter der Transaktionsart, für die es bestimmt ist, in der PIE gespeichert wird.

Die Verwendung von XSLT-Stylesheets hat den Vorteil, das die automatische Formulierung von Zugriffsberechtigungen jederzeit an Veränderungen des Berechtigungskonzeptes im Unternehmen, ohne den Quellcode der PIE zu verändern, angepasst werden kann. Der Systemadministrator der PIE kann neue XSLT-Stylesheets für neue zu berücksichtigende Transaktionen in die PIE, über den XSLT-Manager, integrieren. Des Weiteren kann er XSLT-Stylesheets löschen, die für die Berechtigungsformulierung nicht mehr zu berücksichtigen sind. In diesem Fall ist durch ihn die Entscheidung zu fällen, ob alle für diese Transaktion bestehenden Berechtigungen in allen PolicySets der PAP ebenfalls gelöscht werden sollen. Eine entsprechende Routine ist im XSLT-Manager zu realisieren. Für Änderungen an der Berechtigungsformulierung für Transaktionen, infolge neuer Bestimmungen für die Berechtigungs-formulierungen des Unternehmens, können die bereits vorhandenen XSLT-Stylesheets durch den XSLT-Manager upgedatet werden. Mittels XSLT Control Interface ist der XSLT-Manager ansteuerbar.

Das XSLT-Stylesheet enthält Templates, die entsprechend dem Schema der Policyberechtigungssprache aidXACML strukturiert sind, zur Transformation von EPCIS-Events in PIE-PolicySets. Die Struktur der XSLT-Stylesheets ist vor-definiert und wird im Weiteren vorgestellt. Der Systemadministrator erweitert das allgemein vorgegebenen XSLT-Stylesheet für die Berechtigungsformulierung, im Rahmen der Ressourcenadressierungssprache EAL (siehe Kapitel 2.2.2.1.) und passt somit das formal vordefinierte XSLT-Stylesheet an die betreffende Transaktion und die Berechtigungsformulierung an die Unternehmensinteressen an. Der Aufbau der EAL-Templates wird ebenfalls in diesem Kapitel beschrieben

Die Validierungsprüfung des XSLT-Stylesheets, welches die Logik zur Formulierung von Berechtigungen beinhaltet, erfolgt vom Systemadministrator vor der Integration in die PIE. Zum Einen kann diese Prüfung durch externe XSLT-Verarbeitungsprogramme oder zum Anderen mit einer entsprechenden Prüfungsroutine, die der XSLT-Manager zur Verfügung stellt, erfolgen.

4.4.1. Struktur des XSLT-Stylesheets

Der strukturelle Aufbau eines XSLT-Stylesheets, welches die Formulierungs-regeln für Zugriffsberechtigungen bezogen auf eine Transaktionsart enthält, ist im Anhang 8.8 zu sehen. Als Erstes werden in ein XSLT-Stylesheet die benötigten Javaklassen eingebunden. Anschließend werden die globalen Parameter bzw. Variablen und die Templates zur Formulierung des PIE-

PolicySets definiert. Die globalen Variablen, die den zu verwendenden Combining-Algorithmus für die Policy und Rule angeben, können vom Systemadministrator verändert werden. Alle anderen Variablen dürfen nicht verändert werden. Insgesamt besteht das XSLT-Stylesheet aus 5 Templates. Im ersten Template ist die globale Struktur des PIE-PolicySets vorgegeben. Das Target-Element für das PolicySet, sowie der Rule ist jeweils durch ein Template spezifiziert. Im Template zur Definition des Target-Elementes der Rule ist die Integration der Resourcenadressierungsprache EAL, innerhalb des *AttributeValue*-Elementes bezogen auf das TransactionEvent bzw. das AggregationEvent, durch den Systemadministrator vorzunehmen. Für beide EPCIS-Events kann nicht das gleiche Muster für die EAL-Generierung verwendet werden, da sich die zu berücksichtigenden EPCs für beide Events unterscheiden. Eine Berechtigung auf ein AggregationEvent erfolgt beispielsweise, wenn die durch eine Transaktion bestellten EPC-Objekte in einen Karton verpackt oder auf einer Palette zusammengefasst werden. Sie ist die Erweiterung der Berechtigung, die auf die Bestelltransaktion vergeben wurde, bezogen auf den Karton bzw. die Palette. Für die spätere Integration dieser Berechtigungserweiterung ist es daher wichtig, dass die Information des EPCIS-Eventtyps, für welchen die Berechtigung erstellt worden ist, im *Description*-Element des PIEPolicySets mitgeführt wird.

4.4.2. EAL-Templates

Das EAL-Template für das TransactionEvent bzw. das AggregationEvent ist durch den Systemadministrator, im Rahmen der Geschäftstransaktion für die das XSLT-Stylesheet vorgesehen ist und im Rahmen des Berechtigungskonzept des Unternehmens, zu definieren. Der Aufbau des EAL-Templates erfolgt äquivalent zum EAL-Schema (siehe Kapitel 2.2.2.1. und [EAL08a]). Für Werte, die während der XSL-Transformation einzusetzen bzw. zu ermittelen sind, werden entsprechende XPath-Definitionen angegeben. Entsprechend dieser Definitionen erfolgt die Wertbelegung bei der XSL-Transformation. Die Abbildung 4.4.1 zeigt ein Beispiel für den Aufbau eines EAL-Templates für ein TransactionEvent.

```
<xsl:template name="EALTransactionEvent">

<eal:eventSet>
 <eal:eventSubset visibleAttributes="*" basicSet="ObjectEvent">

  <eal:conditionSets>
   <eal:conditionSet attribute="epcList.epc">

    <xsl:for-each select="//epc">
     <eal:condition>
      <eal:predicate operator="epc-match">
       <xsl:value-of select="string(.)"/>
      </eal:predicate>
     </eal:condition>
    </xsl:for-each>

   </eal:conditionSet>
  </eal:conditionSets>

 </eal:eventSubset>
</eal:eventSet>
</xsl:template>
```

Abbildung 4.4.1 EAL-Template für TransactionEvent

Das EAL-Template beschreibt eine EAL-Instanz, die aus einem eventSubset und einem conditionSet besteht. Zur Formulierung der EAL-Instanz werden die EPCs des zugrunde liegenden EPCIS-Events zur Formulierung von Berechtigungen verwendet. Die EAL-Instanz erlaubt, den Zugriff auf alle Attribute der im *predicate*-Element angegeben EPC-Objekte.

Im Anhang 8.8.2. sind einfache Beispiele zur XSLT-Spezifikation von *conditionSet*-Elementen zu sehen. Sie können entsprechend dem EAL-Schema in die Abbildung 4.4.1 eingefügt werden. Des Weiteren können Bedingungen für die Verwendung bestimmter conditionSet-Elemente spezifiziert werden. So bietet XSLT vielfältige Möglichkeiten Berechtigungen im Rahmen der attributgenauen und regelbasierten Eventadressierung zu erstellen. Die Definition von einfachen sowie komplexen Berechtigungen ist somit realisierbar.

4.5. PIE-Repository

Das PIE-Repository ist eine Datenbank, in der alle für die PIE relevanten Informationen für die Generierung von Zugriffsberechtigungen beinhaltet sind.

Es ist zu analysieren, welche Informationen durch die PIE-Datenbank für die automatische Formulierung von Zugriffsberechtigungen bereit zu stellen sind. Dafür werden die im Rahmen der Berechtigungsformulierung benötigten Informationen ermittelt. Im Anschluss an die Ermittlung erfolgt die Zuordnung des Informationsbedarfes zu den jeweiligen Komponenten. Es werden Informationsquellen zu Komponenten aufgezeigt und in einer Gegenüberstellung die Vor- und Nachteile zur Verwendung der PIE-Datenbank als Informationsquelle verglichen. Aus den Erkenntnissen dieser Analyse wurde das PIE-Datenbankschema (siehe Abbildung 4.5.1) entwickelt. Die Erläuterung erfolgt in Abschnitt 4.5.2.

Die PIE-Komponenten, EPCIS-EventFilter, XSLT-Manager und PIE-Handler, integrieren Daten, die für die PIE wichtig sind in die PIE-Datenbank, und rufen diese zum gegebenen Zeitpunkt für den Bearbeitungsprozess wieder ab. Der EPC-Finder ist nur berechtigt Daten aus der PIE-Datenbank abzurufen.

4.5.1. Informationsanalyse zur Berechtigungsformulierung

Für die Ermittlung des Informationsbedarfes zur Formulierung von Zugriffsberechtigungen sind die folgenden Fragen während des Bearbeitungsprozesses zu klären:

1. Ist für ein EPCIS-Event die Formulierung oder der Entzug von Zugriffsberechtigungen vorzunehmen?
2. Ist die Geschäftstransaktion im Rahmen der Zugriffsberechtigungen zu berücksichtigen?
3. Ist eine Geschäftstransaktion vollständig beendet?
4. Wie erfolgt die Ermittlung von aggregierten EPCs für ein EPC-Objekt, die bei der Zugriffsberechtigungsformulierung zu berücksichtigen sind?
5. Wie erfolgt die Ermittlung aller aggregierten EPCs für eine EPC?
6. Ist die Integration eines PolicySets, einer Policy oder einer Rule am PAP vorzunehmen?

7. Sind Zugriffsberechtigungen für einen User generiert worden, die noch nicht in den PAP integriert wurden?

In der Tabelle 4.5.1 erfolgt die Zuordnung der oben genannten Informationsfragen zu den jeweiligen PIE-Komponenten. Sie gibt gleichzeitig die Quelle mit der die Fragen beantwortet werden an. Anhand der ersten drei Fragen entscheidet der EPCIS-EventFinder, ob ein EPCIS-Event der Auslöser zur Berechtigungsformulierung oder zum Berechtigungsentzug ist. Die Antworten für die erste Frage erhält er aus der Abfrage der Daten des EPCIS-Events. Zur Beantwortung der zweiten Frage muss er prüfen, ob für eine Geschäftstransaktion ein XSLT-Stylesheet in der PIE zur Formulierung der Berechtigungen vorhanden ist. Bei der Initialisierung eines Berechtigungsentzuges ist es für den EPCIS-EventFinder wichtig, ob eine Geschäftstransaktion vollständig oder nur teilweise beendet wurde. Diese Information ist durch Abfrage des EPCIS-Repositories zu ermitteln, wenn die EPC-Liste des EPCIS-Events nicht leer ist. Für diese Ermittlung ist eine *poll*-Abfrage durch den EPCIS-EventFinder an das EPCIS Query Interface zu stellen. Das EPCIS Query Interface liefert die abgefragten EPCIS-Events in Form eines EPCIS-Dokumentes zurück. Aus dem erhaltenen EPCIS-Dokument sind alle EPCs der enthaltenen EPCIS-Events zu ermitteln und mit der EPC-Liste des EPCIS-Events, für das die Abfrage generiert wurde, zu vergleichen. Stimmen beide Listen überein, handelt es sich um eine vollständige Beendigung oder um eine teilweise Beendigung einer Geschäftstransaktion, anderenfalls handelt es sich um einen Abbruch der Geschäftstransaktion. Zur Ermittlung aller aggregierten EPCs für ein bestimmtes EPC-Objekt werden durch den EPC-Finder iterativ Abfragen für jede Aggregation auf das EPCIS-Repository ausgeführt.

Die Informationsermittlung aus dem EPCIS-Repository ist, durch die Formulierung der poll-Anfrage und die Durchsuchung des EPCIS-Dokumentes, Zeit- und Kostenintensiv.

In der Tabelle 4.5.2 werden die Informationsquellen aus der Tabelle 4.5.1 in Gegenüberstellung der Vor- und Nachteile, zur Verwendung der PIE-Datenbank als Informationsquelle, verglichen. Die Informationsquellen EPCIS-Event und EPC-Finder werden in dieser Tabelle nicht betrachtet, da die Verwendung der PIE-Datenbank als Alternative nicht sinnvoll ist.

PIE Komponente	***Informations-bedarf***	***Informationsquelle***
EPCIS-EventFilter	1	EPCIS-Event
	2	Speicher XSLT-Stylesheet
	3	EPCIS-Repository
XSLT-Engine	4	EPC-Finder
XSLT-Manager	-	-
EPC-Finder	5	EPCIS-Repository
PIE-Handler	7	Zwischenspeicher der PolicySet
PIE-Finder	-	-
Policy-Updater	6	PAP

Tabelle 4.5.1 Informationsbedarf der einzelnen PIE-Komponenten

Informations-quelle	*Vorteile zu PIE Datenbank*	*Nachteile zu PIE Datenbank*
Speicher XSLT-Stylesheet	Informationen in der XSLT-Stylesheet Bezeichnung	suche nach Stylesheetbezeichnung im Speicher der XSLT-Stylesheets
EPCIS-Repository	kein doppelte Speicherung von Informationen	Generierung poll-Abfrage Filterung Ergebnis aus EPCIS-Document zusätzliche Anfrage durch PIE auf EPCIS-Repository, dadurch höheres Anfragevolumen
Zwischenspeicher der PolicySet	Informationen in PolicySet	durchsuchen aller PolicySet, die gespeichert sind auf Informationen, hoher Aufwand
PAP	Konsistente Information kein doppelte Speicherung von Informationen	

Tabelle 4.5.2 Informationsquellen und deren Vor- und Nachteile zur PIE-Datenbank

Nach Tabelle 4.5.2 ist es für die Informationsquellen, den Speicher der XSLT-Stylesheets, des EPCIS-Repositories und den Zwischenspeicher der PolicySets, aus Performensgründen effektiv die entsprechenden Informationen in der PIE-Datenbank zu halten. Dadurch können die gewünschten Informationen schneller und effizienter für die PIE zur Verfügung gestellt werden.

4.5.2. PIE-Datenbankschema

Das PIE-Datenbankschema, welches aus der vorangegangenen Untersuchung entwickelt wurde, ist in Abbildung 4.5.1 dargestellt. Es besteht aus Tabellen, die die gewünschten Daten zu den Fragen 2,3,5 und 7 beinhalten.

Die Tabellen EPC und AggEPC spiegeln die EPC-Aggregation wieder. Dabei sind in der EPC-Tabelle alle *parentID* der AggregationEvents gespeichert, die an die PIE übermittelt wurden. In der AggEPC-Tabelle sind alle zugehörigen *childEPCs* gespeichert, mit der Referenz zur ID der *parendID* EPC.

Zur Speicherung der Transaktion und deren zugeordneten EPCs werden die BizTransaction- und die TransactionEPC-Tabelle verwendet. Die Daten für die vier Tabellen werden vom EPCIS-EventFilter in die PIE-Datenbank integriert.

Der XSLT-Manager speichert die Geschäftstransaktionstypen, beispielsweise *po* für die Bestellung, in die Tabelle GRforTransaction. Die Transaktionstypen, die in dieser Tabelle stehen, werden bei der Berechtigungsformulierung berücksichtigt. Für die Generierung dieser Berechtigungen existiert im PIE-Speicher ein entsprechendes XSLT-Stylesheet. Der EPCIS-EventFilter prüft mittels der GRforTransaction-Tabelle, ob eine im EPCIS-Event referenzierte Transaktion für die PIE ein auslösendes EPCIS-Event repräsentiert und somit eine Reaktion, wie die Formulierung von Berechtigungen, erfolgt.

Die Tabelle PIEPolicySet enthält Informationen, wie den vollständigen BizTransaction-Wert (*biztransaction*) oder den Transaktionstyp (*transaction*) zu den im Zwischenspeicher befindlichen PIE-PolicySets, für die noch keine Aktualisierung im PAP erfolgt ist. Der PIE-Handler speichert die Informationen zu

den generierten PolicySets in die PIEPolicySet-Tabelle. Er ruft diese Informationen bei Bedarf wieder ab bzw. löscht diese. Die PIE-PolicySets, die in der PIE zwischengespeichert werden, sind im Speicher der PIE unter der zugeordneten ID zu speichern. Diese ID entspricht der eindeutigen ID der zugehörig Information zum PIE-PolicySet in der PIEPolicySet-Tabelle. Die Bezeichnung des PIE-PolicySet ist damit unabhängig von der PolicySet-Struktur des PAP. Die mehrfache Speicherung von generierten PolicySets in der PIE ist somit nicht möglich.

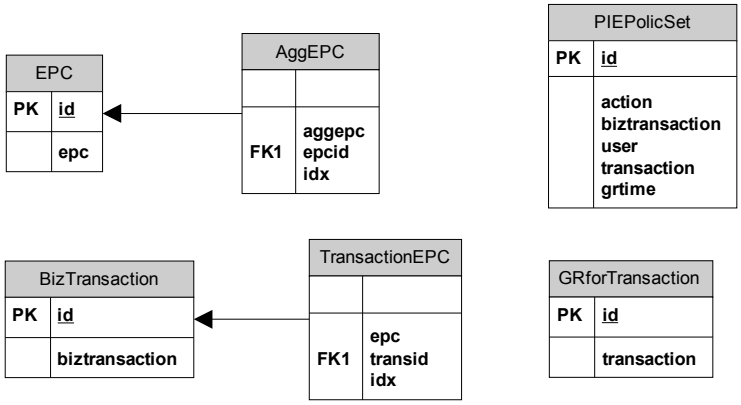

Abbildung 4.5.1 PIE-Datenbankschema

4.6. EPCIS-EventFilter

Der EPCIS-EventFilter ist ein Trigger, der die Berechtigungsformulierung bzw. den Berechtigungsentzug für EPCIS-Events auslöst. Die Logik des Triggers ist in Abbildung 4.6.1 dargestellt. Die Funktionsweise wird nachfolgend erläutert.

Der EPCIS-EventFilter arbeitet auf Basis von EPCIS-Events, die ihm in EPCIS-Dokument von dem EPCIS-System übermittelten werden. Für jedes erhaltene EPCIS-Event entscheidet der EPCIS-EventFilter anhand seiner beinhalteten Logik, wie er auf dieses EPCIS-Event reagieren muss.

4.6.1. Funktionsweise des EPCIS-EventFilters

Die erste Entscheidung für die Reaktion des EPCIS-EventFilters ist der Eventtyp eines EPCIS-Events. So löst der EPCIS-EventFilter für ObjectEvents und QuantityEvents keine Berechtigungsformulierungen bzw. keinen Berechtigungs-entzug aus. Für AggregationEvents und TransactionEvents ist eine weitere Entscheidung, die Prüfung des Wertes des Action-Elements, vorzunehmen. Der Aktionwert gibt an, ob das vorliegende EPCIS-Event im Rahmen der Berechtigungsformulierung oder des Berechtigungsentzuges zu betrachten ist. Die vier möglichen Reaktionswege, die sich daraus ergeben, werden im Folgenden erläutert und sind in der Abbildung 4.6.1 blau nummeriert.

1. Reaktionsweg für TransactionEvent mit dem Aktionwert ADD

Für jede im TransactionEvent referenzierte Geschäftstransaktion ist zu ermitteln, ob für diese Transaktion eine Berechtigung zu formulieren ist. Die Ermittlung erfolgt durch Abfrage der GRforTransaction-Tabelle der PIE-Datenbank. Existiert kein Eintrag für die Transaktion in dieser Tabelle, so wird die Anfrage mit *false*

beantwortet und für diese Transaktion ist keine Zugriffsberechtigung zu formulieren. Anderenfalls wird der Wert *true* zurück geliefert. Die Tabellen, BizTransaction und TransactionEPC, der PIE Datenbank werden, bezogen auf diese Transaktion, aktualisiert. Sie enthalten die Beziehungen zwischen Geschäftstransaktionen und deren zugeordneter EPC-Objekte für die PIE. Anschließend erfolgt die Generierung der Zugriffsberechtigungen für die Transaktion. Zur Generierung der Berechtigungen ist ein TransactionEvent mit nur einer referenzierten Transaktion zu verwenden. Für TransactionEvents, die mehrere Transaktionsreferenzen umfassen, ist für jede zu berücksichtigende Geschäftstransaktion ein eigenes TransactionEvent zu erzeugen und mit diesem die Formulierung vorzunehmen. Diese Formulierung wird durch den EPCIS-EventFilter über die XSLT-Engine ausgelöst.

2. Reaktionsweg für TransactionEvent mit dem Aktionwert DELETE

Ein TransactionEvent, welches die Bedingung für diesen Reaktionsweg erfüllt, kann eine Neuformulierung oder die vollständige Löschung von bereits vergebenen Zugriffsberechtigungen nach sich ziehen. Um dies zu ermitteln, wird die EPCList des TransactionEvents auf EPC-Objekte getestet. Ist die EPC-Liste des EPCIS-Events leer, wird die Löschung für jede zu berücksichtigende Geschäftstransaktion, laut GRforTransaction-Tabelle der PIE-Datenbank, über den PIE-Handler ausgelöst. Enthält das TransactionEvent eine nicht zu berücksichtigende Geschäftstransaktion, erfolgt keine Reaktion für diese Transaktion bezogen auf dieses EPCIS-Event. Bei einer nicht leeren EPC-Liste des TransactionEvents, ist für jede zu berücksichtigende Transaktion die EPC-Liste, aus der PIE-Datenbank abzufragen. Die EPC-Listen, des EPCIS-Events und der abgefragten Transaktion, sind miteinander zu vergleichen. Sind die Listen identisch, dann wird die Löschung der Berechtigungen, für diese Transaktion über den PIE-Handler, veranlasst. Stimmen beide Listen nicht überein, werden die EPC der EPCList des TransactionEvents in der PIE-Datenbank für die betreffende Transaktion gelöscht und ein neues TransactionEvent wird erstellt. Dieses TransactionEvent enthält die Referenz der Transaktion und die verbleibenden EPCs dieser Transaktion. Der Aktionwert dieses Events ist ADD. Alle anderen Elemente werden vom ursprünglichen TransactionEvent übernommen. Für das neuerstellte TransactionEvent wird die Formulierung der Zugriffsberechtigungen, durch den EPCIS-EventFilter über die XSLT-Engine, ausgelöst.

Für die Systeme, die die userbezogene Speicherung der PolicySet im PAP, wie in Abbildung 4.3.2a verwenden, hat der EPCIS-EventFilter ebenfalls die Möglichkeit die Neuformulierung einer bereits bestehenden Berechtigung vorzunehmen. Anderenfalls kann er für das ursprüngliche TransactionEvent, bezogen auf eine Geschäftstransaktion, die Formulierung einer Rule, die ein Verbot für die betreffenden EPCs des TransactionEvents beinhaltet, auslösen.

3. Reaktionsweg für AggregationEvent mit dem Aktionwert ADD

Ein AggregationEvent ist bei der Berechtigungsformulierung dann zu beachten, wenn es den Aktionwert ADD besitzt und für die EPC der ParentID oder für mindestens eine EPC aus der Liste der ChildEPCs bereits eine Berechtigung formuliert worden ist. Der erste Fall tritt ein, wenn eine Bestellung, wie im Kapitel 3.2.1.5. beschrieben, während der Produktion eingeht und die aggregierten EPCs zum bestellten EPC-Objekt noch nicht bekannt sind. Der zweite Fall tritt

ein, wenn die bestellten EPC-Objekte zu Verpackungseinheiten, in Kartons bzw. auf Paletten, zusammengefasst werden.

Zu Beginn des Reaktionsweges werden die Tabellen, EPC und AggEPC, der PIE-Datenbank für jedes eingehende AggregationEvent aktualisiert. Sie enthalten alle Aggregationsbeziehungen, die im EPCIS-System auftreten. Für die EPC der ParentID eines AggregationEvents werden die zugehörigen Geschäftstransaktionen über die PIE-Datenbank abgefragt, für die diese EPCs existieren. Diese Abfrage ist notwendig, da Transaktionen nur optional im AggregationEvent referenziert sind. Liefert die Datenbankabfrage keine Tupel zurück, also eine leere Liste, erfolgen erneute Abfragen auf die PIE-Datenbank.

Für jedes ChildEPC-Element des AggregationEvents werden nun die zugehörigen Geschäftstransaktionen ermittelt. Ist ein ChildEPC-Element mit keiner Geschäftstransaktion verbunden, wird eine leere Liste für die Datenbankanfrage zurückgegeben. Es wird keine Berechtigungsformulierung für die EPC der ParentID des AggregationEvents, bezogen auf die jeweilige Transaktion, ausgelöst. Wird eine nicht leere Liste infolge der Datenbankabfrage zurückgegeben, ist für jede Transaktion, bezogen auf die EPC der ParentID des AggregationEvents, eine Berechtigung durch die XSLT-Engine, ohne die Verwendung des EPC-Finders, zu formulieren. Für die Generierung dieser Berechtigung wird ein neues AggregationEvent erzeugt, welches die relevanten Elemente, die ParentID, sowie die Referenz für die betreffende Transaktion enthält.

Liefert die erste Datenbankabfrage, mit der die Geschäftstransaktionen der EPC der ParentID eines AggregationEvents ermittelt werden, keine leere Liste als Ergebnis zurück, ist für jede zurückgelieferte Transaktion ein neues TransactionEvent zu erstellen. Dieses TransactionEvent enthält alle zur Transaktion gehörenden EPCs und die Transaktion selbst, die als einzige Referenz im BizTransactionList-Element enthalten ist. Der Aktionwert des TransactionEvents ist ADD. Weiterhin sind die für ein TransactionEvent relevanten Elemente und deren Werte, siehe Tabelle 2.1.2, vom AggregationEvent zu übernehmen. Für das neu erstellte TransactionEvent wird die Zugriffsberechtigungsformulierung durch den EPCIS-EventFilter über die XSLT-Engine ausgelöst.

4. Reaktionsweg für AggregationEvent mit dem Aktionwert DELETE

Für ein AggregationEvent, welches den Aktionwert DELETE besitzt, können mehrere Berechtigungsentzüge auslöst werden. Diese Entscheidung ist für die PIE nicht eindeutig entscheidbar, da es sich bei einem solchen AggregationEvent entweder um eine Auflösung einer Aggregation oder um eine Löschung der Aggregation im EPCIS-System handelt. Im letzteren Fall, soll die gelöschte Aggregation nicht mehr für den User als EPCIS-Event sichtbar sein. Diese Entscheidung kann nicht, mit den der PIE zur Verfügung stehenden Informationsquellen, dem AggregationEvent oder der PIE-Datenbank, entschieden werden. Daher ist ein Berechtigungsentzug durch die PIE, im Rahmen dieses Reaktionsweges, nicht möglich und die Entscheidung ist vom Administrator zu treffen. Es ist sinnvoll von Seitens der PIE eine Vorfilterung der AggregationEvents vorzunehmen, und nur die AggregationEvents zur manuellen Prüfung weiterzuleiten, für deren enthaltenen EPC-Objekte Berechtigungen formuliert worden sind.

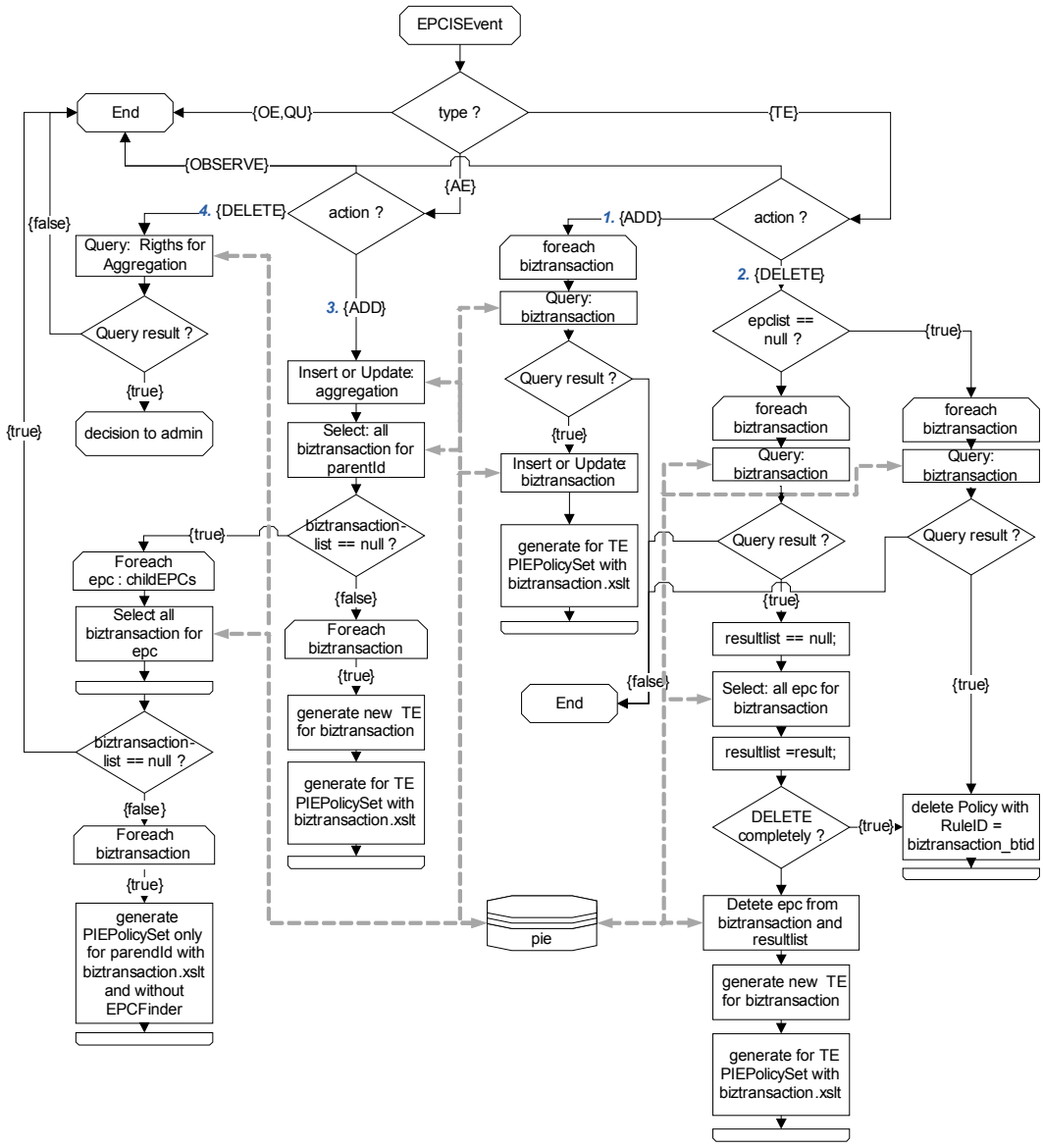

Abbildung 4.6.1 Triggerlogik des EPCIS-EventFilters

4.7. XSLT-Engine

Die XSLT-Engine realisiert die Formulierung von Berechtigungen über die XSL-Transformation mittels XSLT-Stylesheet. Ausgelöst wird diese Formulierung durch den EPCIS-EventFilter. Die erstellten Berechtigungen werden an den PIE-Handler weitergeleitet.

4.7.1. Varianten der Instanziierung

Im Rahmen des Konzeptes wurden zwei Varianten der Instanziierung von Berechtigungen betrachte. Die erste Generierungsvariante ist die Formulierung eines PIE-PolicySets. Durch die PIE-PolicySet-Generierung wird die Entscheidung, ob ein PolicySet, eine Policy oder eine Rule neu zu erstellen oder zu aktualisieren ist, erst nach der XSL-Transformation getroffen.

Bei der zweiten Instanziierungsvariante wird diese Entscheidung bereits vor der Transformation getroffen. In diesem Fall wird ein EPCIS-Event in ein PolicySet, eine Policy oder eine Rule umgewandelt und anschließend wird das transformierte Element in den PAP integriert.

Die Tabelle 4.7.1 vergleicht beide Varianten anhand verschiedener Kriterien, die für die PIE von Bedeutung sind. Entscheidende Nachteile der individuellen Generierungsvariante sind die eingeschränkte Flexibilität gegenüber Änderungen im PAP und die komplexe Gesamtstruktur der XSLT-Stylesheets. Aufgrund dieser komplexen Struktur ist der Austausch von XSLT-Stylesheets für Transaktionen nicht während der Laufzeit möglich. Somit wird die XSLT-Generierung für das PIE-PolicySet in der PIE umgesetzt.

Kriterien	***PIEPolicySet Generierung***	***Individuelle Generierung***
Update-entscheidung	nach der Formulierung getroffen	vor der Formulierung getroffen
Generierungs-aufwand	größer als bei Individueller Generierung	optimal
Zwischen-speicherbedarf	konstant	von Generierung abhängig
Flexibilität auf Änderung im PAP nach der Generierung	hohe Flexibilität, aus PIEPolicySet kann das PolicySet, die Policy oder die Rule erstellt werden	eingeschränkt, es kann aus einer Rule kein PolicySet (bzw. für Policy der Struktur Abbildung 4.3.2b), ohne weitere Informationen, erzeugt werden
Bezug Berechtigung zu Transaktionen	vorhanden	vorhanden
Bezug Berechtigung zu User	vorhanden	bei der Generierung von Berechtigungsrules fehlt dieser Bezug
XSLT-Stylesheet-struktur	ein Stylesheet pro Transaktion	ein Stylesheet für das PolicySet, Policy, Rule und für jede Transaktion das Stylesheet des PolicySet enthält die Referenz auf das Stylesheet der Policy das Stylesheet der Policy enthält die Referenz auf das Stylesheet der Rule das Stylesheet der Rule enthält die Referenz auf die Stylesheet der Transaktionen
XSLT-Stylesheets einfügen bzw. löschen	während der Laufzeit der PIE Aktualisierung der Tabelle GrforTransaction der PIE	nicht während der Laufzeit, da Änderungen am Rule Stylesheet vorzunehmen ist dynamisches einfügen in XSLT-Stylesheets nicht möglich

Tabelle 4.7.1 Vergleich der Instanziierungsarten

4.7.2. XSLT-Generierung von PIE-PolicySets

Die Funktionsweise der XSLT-Engine, die in der PIE integriert ist, wird in der Abbildung 4.7.1 gezeigt. Für EPCIS-Events, die genau eine zu berücksichtigende Geschäftstransaktion referenzieren und vom EPCIS-Eventtype, AggregationEvent oder TransactionEvent sind, existieren XSLT-Stylesheets, bezogen auf den Transaktionstyp. Bei der Umwandlung vom EPCIS-Events in das PIE-PolicySets wird das entsprechende XSLT-Stylesheet vom XSLT-Parser eingelesen und das PIE-PolicySet erstellt. Das PIE-PolicySet enthält die Zugriffsberechtigungen für dieses EPCIS-Event.

Während der Transformation des EPCIS-Events zum PIE-PolicySet, können die EPCs, die für die Formulierung von Berechtigung relevant sind, durch den EPC-Finder über die PIE-Datenbank ermittelt werden. Die Methoden des EPC-Finders werden, wie in Kapitel 2.4.3 beschrieben, in das XSLT-Stylesheet eingebunden und sind somit durch den XSLT-Parser ausführbar.

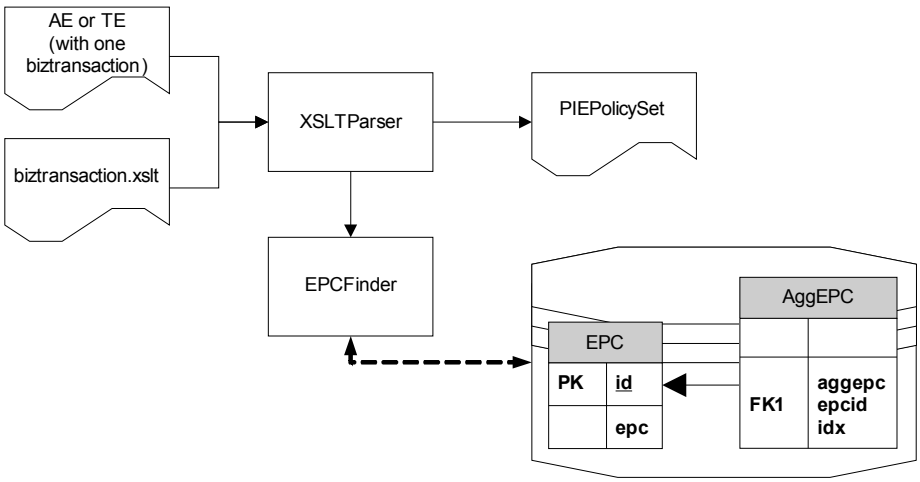

Abbildung 4.7.1 Umwandlung von EPCIS-Events in PIE-PolicySets

4.8. EPC-Finder

Der EPC-Finder dient zur Ermittlung aller EPCs, die zu einem EPC-Objekt gehören. Die Ermittlung kann in der Aggregationstiefe beschränkt werden. Des Weiteren können EPC-Objektklassen von der Ermittlung ausgeschlossen werden. Somit ist die Generierung von Berechtigungen individuell, wie in Abbildung 3.2.8b und c, realisierbar.

Der EPC-Finder stellt Methoden zur Integration relevanter EPCs, die im Rahmen der XSL-Transformation genutzt werden können, und den Algorithmus zur EPC-Ermittlung zur Verfügung. Der Algorithmus ist in Abbildung 4.8.1 dargestellt und wird im Weiteren erläutert.

4.8.1. EPC-Ermittlungsalgorithmus

Die EPC-Ermittlung erfolgt für jeweils eine EPC-Objekt. Für dieses EPC-Objekt können die Beschränkungen, die Aggregationstiefe (*permitdepth*) bzw. die nicht zu betrachtende EPC-Objektklassen *(hiddenEPCclass),* angegeben werden.

Der Grundgedanke der EPC-Ermittlung ist, anhand des Aggregationsbaumes, für das gegebene EPC-Objekt eine Liste mit Knoten zurückzugeben, die die zu berücksichtigende EPC-Menge für die jeweiligen Berechtigungen beinhaltet.

Bei jeder Iteration werden alle Kinderknoten von den Elternknoten einer Ebene des Aggregationsbaumes ermittelt. Die nicht erlaubten Knoten werden aus der erhaltenen Menge entfernt. Die verbleibende Knotenmenge wird zur Rückgabemenge hinzugefügt und für die erneute Ermittlung der Kinderknotenmenge verwendet. Die Abbruchbedingung für die iterative EPC-Ermittlung ist die Überschreitung der erlaubten Aggregationstiefe, eine leere Knotenmenge für die Ermittlung oder eine leere Knotenmenge als Ermittlungsergebnis.

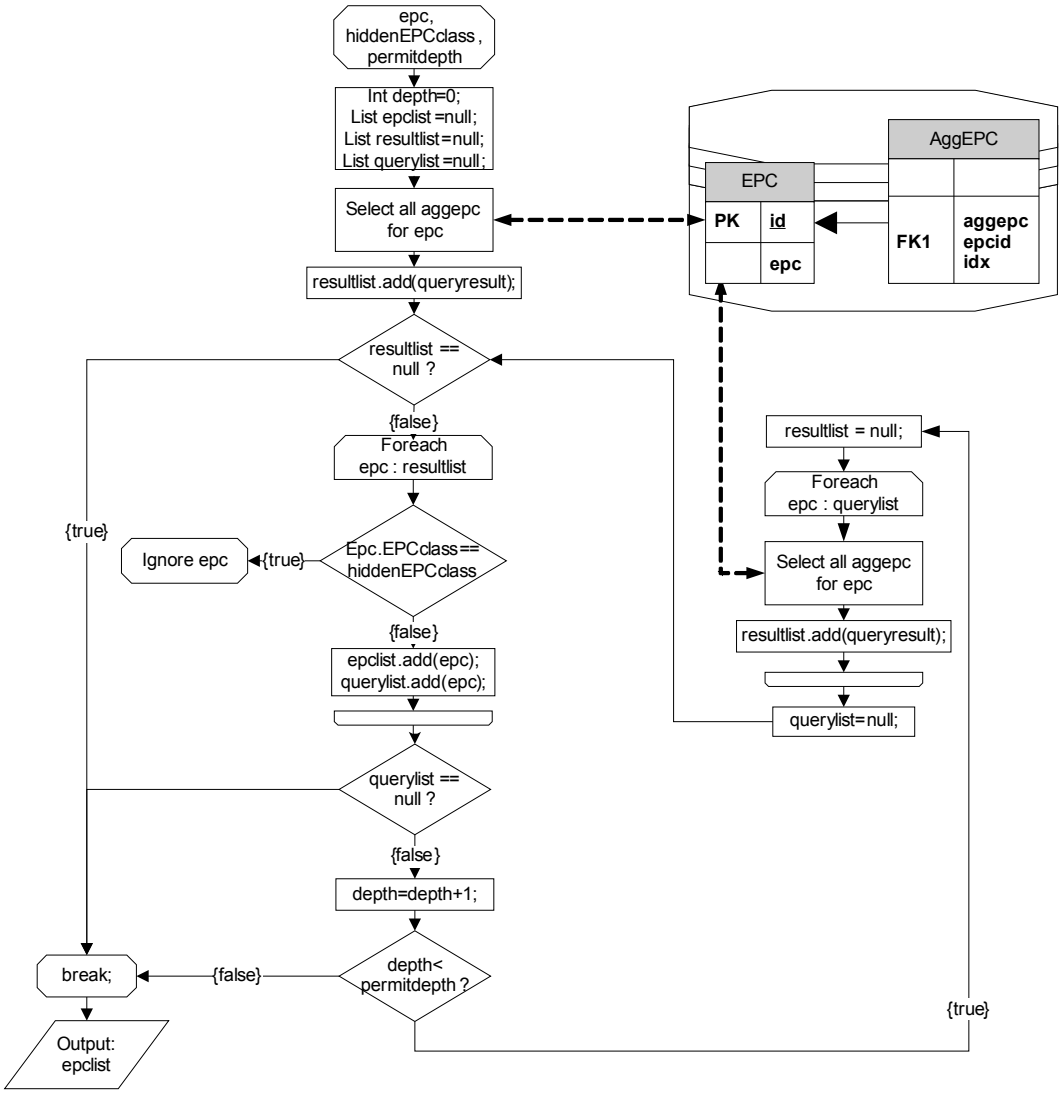

Abbildung 4.8.1 Ermittlung der zu berücksichtigenden EPCs eines EPC-Objekts

Die Funktionsweise des Algorithmus zur EPC-Ermittlung wird anhand des Beispiels, welches in Abbildung 4.8.2 dargestellt ist, beschrieben. Das Beispiel zeigt den Aggregationsbaum für ein Objekt mit der Bezeichnung A:113. Diese Bezeichnung wird stark vereinfacht repräsentiert. Die Objektbezeichnung setzt sich aus der Objektklasse und der eindeutigen ObjektID zusammen. Für das Objekt A:113 sollen nun die relevante EPC-Menge ermittelt werden. Dabei sind

die Objektklassen X ,sowie N nicht und nur Objekte bis zu eine Aggregationstiefe von 3 zu berücksichtigen. Die Abbildung 4.8.3 zeigt den Ablauf des Algorithmus, der die Ergebnismenge {B:139,C:17,D:159,M:719,E:11,P:99,F:1} zurückgibt.

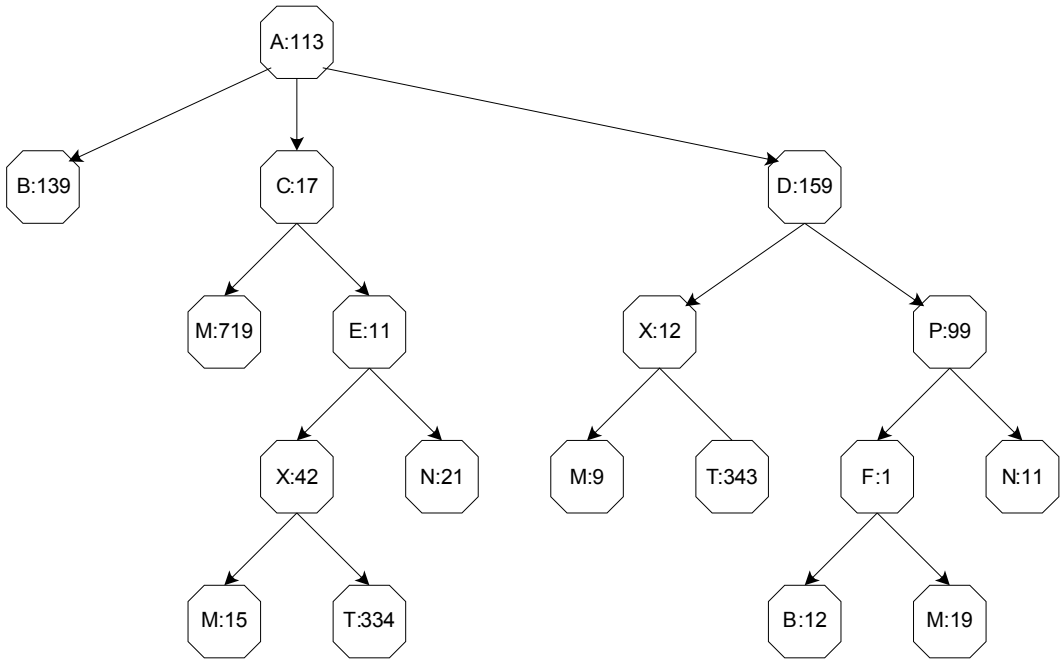

Abbildung 4.8.2 Aggregationsbaum für das Objekt A:133

```
geg: epc= A:113, permitdepth=3, hiddenEPCclass=[X,N]

Beginn:
       depth=0;
       epclist=null;
       resultlist=null;
       querylist=null;

       resultlist=[B:139,C:17,D:159]

1.Iteration:
       epclist=[B:139, C:17, D:159];
       querylist=[B:139, C:17, D:159];
       depth=1;
       resultlist=[M:719,E:11,X:12,P:99]

2.Iteration:
       epclist=[B:139,C:17,D:159,M:719,E:11,P:99];
       querylist=[M:719,E:11,P:99]
       depth=2;
       resultlist=[X:42,N:21,F:1,N:11]

2.Iteration:
       epclist=[B:139,C:17,D:159,M:719,E:11,P:99,F:1];
       querylist=[F:1]
       depth=3;
       break;

Output: epclist=[B:139,C:17,D:159,M:719,E:11,P:99,F:1]
```

Abbildung 4.8.3 Ablauf der EPC-Ermittlung für ein Objekt

4.9. PIE-Handler

Der PIE-Handler setzt die Aktualisierungstrategie, die für die PIE gewählt wird, um. Im Rahmen dieser Arbeit wird die permanente, die periodische und die userbezogene Aktualisierungsstrategie vorgestellt und auf Anpassungen des PIE-Handler für die jeweilige Strategie eingegangen. Anschließend werden die Strategien gegenübergestellt und verglichen. Zur Realisierung der Integration, sowie der Löschung von Berechtigungen verwendet der PIE-Handler den Policy-Updater.

4.9.1. Permanente Aktualisierungsstrategie

Diese Strategie setzt die Veränderungen für PolicySets des PAP sofort um. Sie kann für die Integration und die Löschung von Berechtigungen verwendet werden. Es erfolgt keine Zwischenspeicherung von PolicySets innerhalb der PIE, somit wird die Tabelle PIEPolicySet der PIE-Datenbank nicht benötigt. Des Weiteren ist das Query Rights Interface, sowie der PIE-Finder bei dieser Strategie überflüssig. Die permanente Aktualisierung hat den Vorteil, das im PAP die aktuellen Berechtigungen gespeichert sind und die für Zugriffsent-scheidungen auf Daten des EPCIS-Repositories zur Verfügung stehen. Diese Methode bringt einen hohen Performanceaufwand für die permanente Aktualität mit sich, da auch nicht benötigte Berechtigungen an den PAP übermittelt werden.

4.9.2. Periodische Aktualisierungsstrategie

Diese Aktualisierungsstrategie nimmt periodisch Veränderungen an PolicySets der PAP vor. Sie ist für die Integration und die Löschung von Berechtigungen vorgesehen. Bei dieser Strategie ist die Zwischenspeicherung von PolicySets innerhalb der PIE erforderlich. Informationen, zu den im Speicher der PIE befindlichen PIE-PolicySets, werden in der Tabelle PIEPolicySet der PIE-Datenbank durch den PIE-Handler hinterlegt. Anhand dieser gespeicherten Informationen ist es dem Administratior der PIE möglich, die in der PIE gespeicherten PIE-PolicySets, beispielsweise nach einem Systemausfall, wieder herzustellen. Die periodische Aktualisierung hat den Vorteil, dass der Aktualisierungszeitpunkt gesteuert werden kann und es keinen permanenten Zugriff seitens der PIE auf die PAP gibt. Das kann die Performance des gesamten Systems verbessern, was zum Einen vom System selbst und zum Anderen von der gewählten Periode zur Aktualisierung abhängig ist. Bei dieser Methode sind Berechtigungen, die in der PAP gespeichert sind nicht immer aktuell. Das bedeutet, dass für Zugriffsentscheidungen auf das EPCIS-Repository die benötigten Berechtigungen teilweise noch nicht oder dass zu entziehende Berechtigungen immer noch zur Verfügung stehen. Ein weiterer Nachteil ist, wie bei der permanenten Aktualisierung, dass nicht benötigte Berechtigungen an den PAP übermittelt werden.

4.9.3. Userbezogene Aktualisierungsstrategie

Die userbezogene Strategie der Aktualisierung von Berechtigungen des PAP bringt den Vorteil, das nur Berechtigungen in den PAP integriert werden, für die Anfragen an das EPCIS-Repository gestellt wurden sind. Berechtigungen, die nicht für Abfragen benötigt werden, werden nach einer bestimmten Zeit aus dem Speicher der PIE gelöscht, ohne dem PAP übermittelt worden zu sein. Der PAP

enthält somit nur Berechtigungen, die auch wirklich angewendet werden. Für den Berechtigungsentzug ist diese Aktualisierungsstrategie nicht geeignet. Berechtigungen, die für eine Bestellung vergeben und in den PAP integriert wurden, sind infolge einer Stornierung der Bestellung auch wieder zu löschen.

Über das Query Rights Interface werden die Useranfragen an den PIE-Finder übermittelt, der die Anfragen an den PIE-Handler weiterleitet und die Aktualisierung auslöst. Dieses Vorgehen hat Auswirkungen auf die Antwortzeit für die Useranfrage des EPCIS-Repository, da sich diese Antwortzeit um die Zeit der Integration der Berechtigung verlängert.

4.9.4. Zusammenfassung

Die Tabelle 4.9.1 fasst die wichtigsten Erkenntnisse aus diesem Kapitel für die jeweiligen Aktualisierungsstrategien zusammen und stellt sie gegenüber. Es ist je nach Unternehmensinteressen, Systemvoraussetzungen und der Frequentierung des EPCIS-Repositories zu entscheiden, welche Strategie verwenden wird.

Für ein EPCIS-System, bei dem für jede Transaktion eine Zugriffsentscheidung, infolge der Anfrage an das System, die vollständig Übermittlung der generierten Berechtigungen an den PAP zur Folge hat, ist die Umsetzung der permanenten Aktualisierungsstrategie am effektivsten. Da alle Berechtigungen an den PAP übermittelt werden, ohne das die Zwischenspeicherung der Berechtigungen erfolgt und es keine zeitliche Verzögerungen für die Ermittlung von Zugriffsentscheidungen gibt.

Kriterien	***permanent***	***periodisch***	***userbezogene***
Aktualisierung	permanent	periodischen Zeitanstände	per Useranfrage
Verwendung	Integration und Löschung von Berechtigungen	Integration und Löschungen von Berechtigungen	Integration von Berechtigungen
Speicherung von PolicySets in PIE	nein	ja	ja
Berechtigungen in PAP	aktuell	teilweise aktuell	aktuell, für User nach deren Userabfrage
Übermittlung generierter Berechtigungen	vollständig	Vollständig, verzögert	individuell nach Bedarf
Einfluss auf Zugriffs-entscheidungen	nein	Aktualität	Entscheidungszeit
mögliche Kombinationen	-	userbezogene Strategie	permanente Strategie für Berechtigungsentzug

Tabelle 4.9.1 Vergleich der Strategien zur Aktualisierung

4.10. Policy-Updater

Der Policy-Updater wird zur Integration, zur Aktualisierung und zur Löschung von Berechtigungen in PolicySets des PAP verwendet. Die Struktur des Policy-Updaters ist abhängig von der userbezogenen Speicherung der PolicySets im PAP (siehe Kapitel 4.3.2.).

Die Funktionsweise des Policy-Updaters wird im Rahmen dieses Kapitels für den PAP, dessen PolicySets wie in Abbildung 4.3.2b aufgebaut sind, beschrieben. Der Policy-Updater arbeitet nur auf Policies, deren IDs mit dem Useraccount[5] bezeichnet werden. Diese Policies ist ausschließlich für die automatische Formulierung von Zugriffsberechtigungen vorgesehen. Manuell erstellte Berechtigungen sind in das PolicySet des Users nur in Policies zu integriert, die nicht mit der für die PIE reservierten ID bezeichnet worden sind.

Mittels PAP-Client können Usernamen zu PolicySets, sowie PolicySets selbst abgefragt werden. Des Weiteren können PolicySets gesetzt und gelöscht werden. Die Prüfung der zu integrierenden PolicySets auf Wohlgeformtheit und Gültigkeit wird von Seitens des PAP durchgeführt.

4.10.1. Integration von Berechtigungen

Der Prozessablauf, der vom Policy-Updater zur Integration bzw. zur Aktualisierung von Berechtigungen des PIE-PolicySets im PAP vorgenommen wird, ist in Abbildung 4.10.1 dargestellt.

Im Rahmen der Berechtigungintegration stehen dem Policy-Updater vier Reaktionsvarianten zur Verfügung.

Die erste Variante ist die direkte Integration des PIE-PolicySets in den PAP. Diese direkte Integration wird vom Policy-Updater vorgenommen, wenn es im PAP für den User, für welchen die Berechtigungen im PIE-PolicySet formuliert sind, kein PolicySet oder ein PolicySet ohne Policies existiert.

Besitzt der User ein PolicySet mit mindestens einer Policy und existiert unter diesen Policies keine Policy, deren ID dem Useraccount entspricht, so ist die Policy des PIE-PolicySets in das PolicySet des Users von PAP zu integrieren. Das aktualisierte PolicySet ist in den PAP zurückzuschreiben.

Weitere Varianten sind die Integration der Rule des PIE-PolicySets in das PolicySet des Users von PAP oder das Überschreiben der Rule für die Geschäftstransaktion im PAP-PolicySet des Users mit der Rule des PIE-PolicySets. Diese Integrationvarianten werden für PolicySets vom PAP angewendet, die eine Policy besitzen, deren ID mit dem Useraccount bezeichnet ist. Existiert in dieser Policy eine Rule, die die gleiche Bezeichnung trägt, wie die Rule des PIE-PolicySets, so wird diese Rule überschrieben. Andernfalls wird die Rule des PIE-PolicySets in die Policy des PAP integriert. Das aktualisierte PolicySet wird bei beiden Varianten wieder in den PAP zurückzuschreiben.

Die vier gezeigten Reaktionsmöglichkeit beziehen sich auf PIE-PolicySets, die auf Basis von TransactionEvents formuliert worden sind. Der Bezug, auf welchem EPCIS-Event die generierten PIE-PolicySets basieren, ist im *Description*-Event des PIE-PolicySets angegeben.

Die PIE-PolicySets, deren Description-Wert den String AggregationEvent

5) Der Useraccount ist identisch den Usernamen.

beinhaltet, sind infolge der Umwandlung eines AggregationEvents entstanden. Diese PIE-PolicySets stellen Erweiterungen von bereits bestehender PolicySets im PAP dar. Die EAL-Instanz dieser PolicySets ist, um das eventSubset der EAL-Instanz des PIE-PolicySets, zu erweitern. Das erweiterte PolicySet wird anschließen in den PAP zurückgeschrieben. Der Policy-Updater gibt eine Fehlermeldung aus, wenn die Integration der Instanzerweiterung nicht möglich ist.

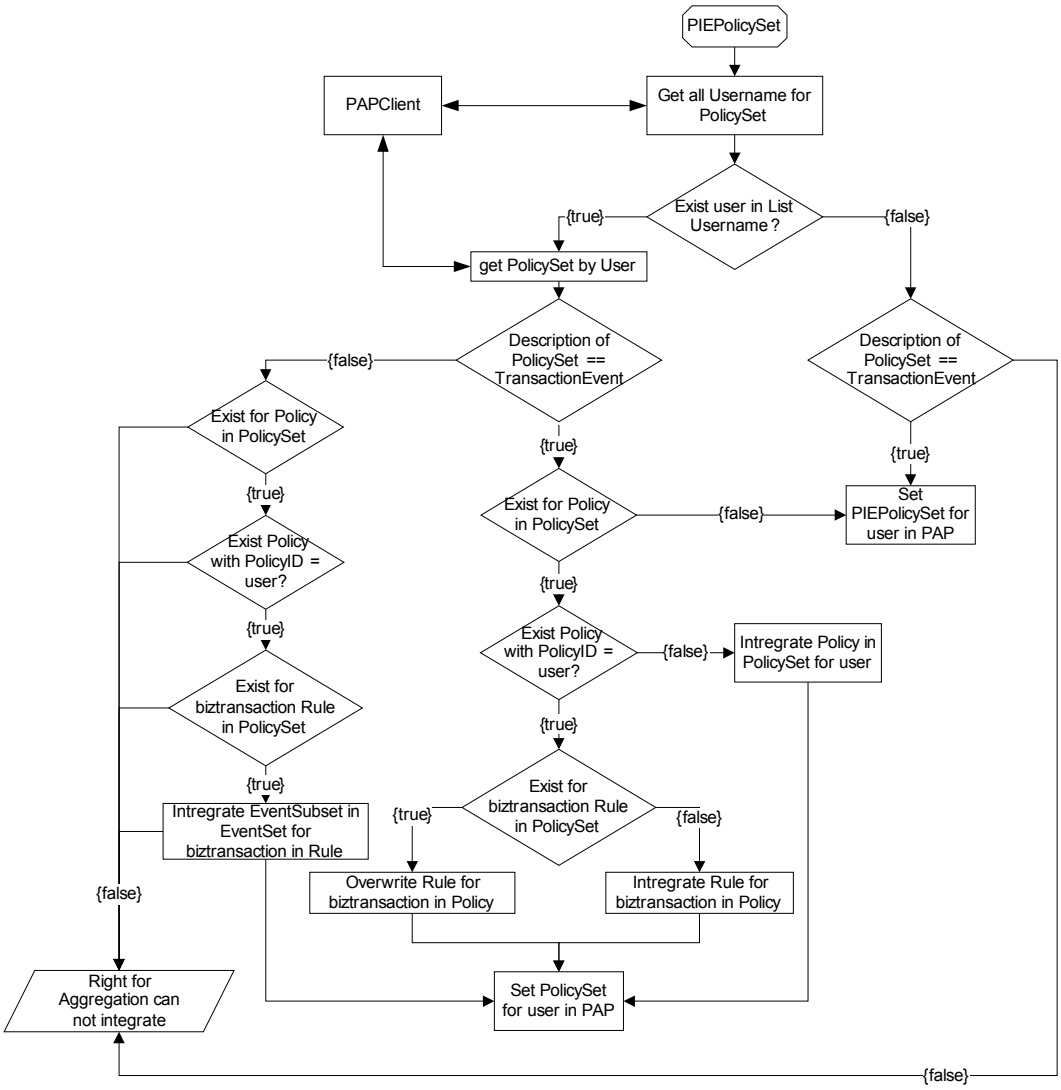

Abbildung 4.10.1 Integration von Berechtigungen

4.10.2. Deintegration von Berechtigungen

Die Löschung von Berechtigungen wird durch den EPCIS-EventFinder für EPCIS-Events angewendet, die den vollständigen Berechtigungsentzug auslösen. Er übermittelt dem Policy-Updater den biztransaction-Wert dieser EPCIS-Events, deren zugehörige Berechtigungen zu löschen sind.

Der Prozess zur Löschung von Berechtigungen, der dann im Policy-Updater abläuft, ist in Abbildung 4.10.2 dargestellt. Der Policy-Updater hat vier Möglichkeiten auf den Löschungsimpuls des EPCIS-EventFinders zu reagieren.

Zum Ersten kann der Policy-Updater die Löschung des gesamten PolicySets des Users veranlassen, wenn für den User nur die zu löschenden Berechtigungen in seinem PolicySet existieren.

Eine weitere Reaktionsmöglichkeit ist es, die Policy, die für die PIE zur Bearbeitung reserviert ist, zu löschen. Das erfolgt dann, wenn im PolicySet des Users mehrere Policies enthalten sind und die Policy für die PIE nur die zu löschenden Berechtigungen beinhaltet. Umfasst die Policy nicht nur die zu löschenden Berechtigungen, so wird in dieser Policy die Rule für die Geschäftstransaktion, die im übermittelten biztransaction-Wert enthalten ist, gelöscht. Das aktualisierte PolicySet wird in beiden Fällen wieder in den PAP zurückgeschrieben.

Die vierte und letzte Reaktionsmöglichkeit des Policy-Updaters im Rahmen der Berechtigungslöschung ist eine Fehlermeldung, die durch das Fehlen des PolicySets für den User im PAP, durch das Fehlen der Policy für die Bearbeitung der PIE oder durch das Fehlen der Rule für die zu löschenden Berechtigungen generiert wird.

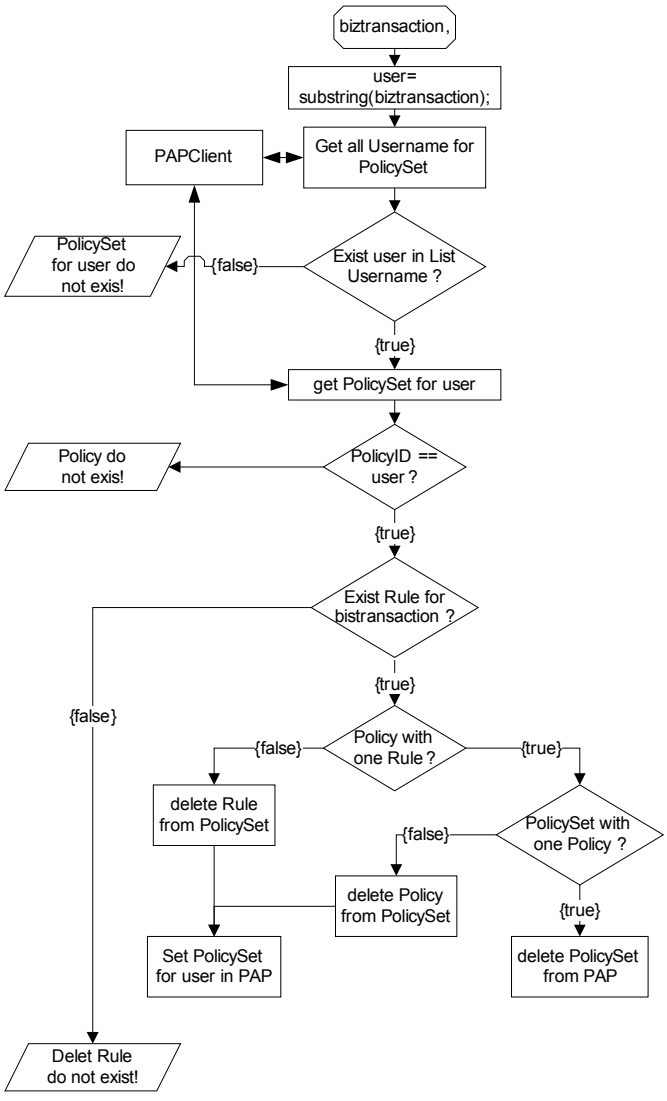

Abbildung 4.10.2 Löschen von Berechtigungen

4.11. Zusammenfassung

Das vorgestellte Konzept der PIE realisiert die geschäftstransaktionsbasierte Formulierung von Zugriffsberechtigungen, unter der Berücksichtigung der zuvor gestellten Anforderungen (siehe Tabelle 3.2.13). Das PIE-Konzept umfasst eine Entscheidungslogik für die Berechtigungsvergabe und den Berechtigungsentzug, bezogen auf relevante EPCIS-Events und deren zu berücksichtigende Geschäftstransaktionen. Die Definition der Regeln für die Generierung von Berechtigungen erfolgen durch XSLT-Stylesheets, damit ist die konstante Formulierung von komplexen unternehmens- und userspezifischen, sowie produktbasierten Zugriffsberechtigungen realisierbar. Ausnahmen und Sonderfälle können ebenfalls bei der Formulierung berücksichtigt werden. Während der Berechtigungsgenerierung können die noch fehlenden EPCs im Rahmen der XSL-Transformation von EPCIS-Events, die für die vollständige Erstellung der Zugriffsberechtigungen notwendig sind, durch den EPC-Finder integriert werden. Die Integration neuer Berechtigungen, die Aktualisierung bzw. Erweiterung, sowie die Löschung vorhandenen Zugriffsberechtigungen wird, in Abhängigkeit von der gewählten Aktualisierungsstrategie, effizient und effektiv am PAP durchgesetzt. Für diese Durchsetzung wurden zwei konkrete Strukturen des PolicySets entwickelt, die im PAP für die Automatisierung verwendet werden können.

In der Konzeption der PIE sind die wesentlichsten Programmabläufe grafisch dargestellt. Sie bilden die Grundlage für die spätere Implementierung und vollständige Umsetzung der PIE. In der prototypischen Implementierung der PIE für diese Arbeit werden ausgewählte Teile des PIE-Konzeptes, die die Funktionalität des Konzeptes der PIE zeigen und unterstreichen, umgesetzt. Die prototypischen Implementierung wird im nachfolgenden Kapitel vorgestellt. Die Umsetzung der PIE erfolgt für PAP-PolicySets, die die Struktur wie in Abbildung 4.3.2b besitzen.

5. Implementierung der Policy Instantiation Engine

Die prototypische Implementierung des PIE-Konzeptes verwendet das open-source EPCIS-Projekt von Fosstrak, ehemals Accada. Mit dem Fosstrak-Modul EPCIS Capture Application werden EPCIS-Events generiert. Die generierten EPCIS-Events werden im EPCIS-Dokument an die PIE übermittelt, nach deren erfolgreicher Integration in das EPCIS-Reporitory. Die implementierte PIE realisiert vollständig die Berechtigungsvergabe, sowie den Berechtigungsentzug nach der permanenten Aktualisierungsstrategie für die zu berücksichtigenden Geschäftstransaktionen und setzt die Vergabe bzw. den Entzug am PAP durch. Der Stand der Umsetzung der PIE, die verwendete Technologie für die PIE und die Struktur der PIE-Implementierung wird im Kapitel 5.2. beschrieben. Des Weiteren wird auf die Systemintegration der PIE und die Definition, der zu berücksichtigenden Geschäftstransaktionen, für die PIE eingegangen. Im Anschluss daran wird die PIE-Datenbank, die EPC-Ermittlung durch den EPC-Finder und dessen Integration in XSLT-Stylesheets vorgestellt. Im 6. Kapitel erfolgt die Bewertung und Validierung der PIE-Konzeption, sowie der PIE-Implementierung.

5.1. EPC Information Services (Fosstrak)

Das open-source EPCIS-Projekt von Fosstrak wird im Rahmen der prototypischen Umsetzung der PIE für die Generierung von EPCIS-Events verwendet.

Das Fosstrak EPCIS-Projekt umfasst drei separate Module: eine EPCIS-Repository Implementierung, eine interaktive EPCIS Capture Application und eine interaktive EPCIS Query Application [FOS08]. Die Abbildung 5.1.1 zeigt die Client-Server-Architektur des EPCIS-Projekts. Dieses Projekt basiert auf der Programmiersprache Java und entspricht weitgehend den Spezifikationen von EPCglobal.

Der Server ist das EPCIS-Repository, welches Schnittstellen für die Client-Anbindung zur Verfügung stellt. Das EPCIS-Repository läuft als Webservice auf einem Tomcat-Server und speichert die EPCIS-Events in einer MySQL-Datenbank, unter Verwendung des Spring- und Hibernate-Frameworks. Es analysiert die Clientanfragen und verarbeitet diese entsprechend der, in der Spezifikation definierten, Regeln. Die verwendeten Übermittlungsprotokolle der Client-Application ist XML over HTTP bzw. SOAP over HTTP [FOS08].

Die Clients sind die EPCIS Capture Application und die EPCIS Query Application. Die EPCIS Capture Application ermöglicht die Generierung von EPCIS-Events über eine grafische Benutzeroberfläche. Ein so definiertes EPCIS-Event wird im EPCIS-Dokument zur Integration an das EPCIS-Repository, durch die EPCIS Capture Application, übermittelt. Im Rahmen der prototypischen Umsetzung wurde die EPCIS Capture Application als Eventgenerator für die PIE verwendet. Die Übermittlung der EPCIS-Events erfolgt an die PIE mittels standardisierter EPCIS-Dokumente, nach dem Erhalt des positiven Request für die Speicherung der EPCIS-Events in das EPCIS-Repository. Die spätere, also nicht prototypische PIE-Integration erfolgt auf der Seite des EPCIS-Repository-Servers.

Die Abfrage der EPCIS-Events erfolgt mittels EPCIS Query Application, die ebenfalls über eine graphische Benutzeroberfläche verfügt. Die EPCIS Query Application wird im Rahmen der prototypischen Umsetzung nicht benötigt.

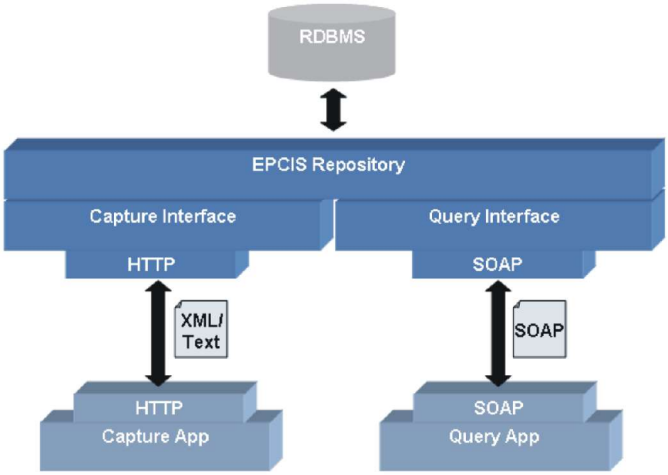

Abbildung 5.1.1 EPCIS-Projekt von Fosstrak [FOS08]

5.2. PIE-Prototyp

Der PIE-Prototyp realisiert die geschäftstransaktionsbasierte Formulierung für Zugriffsberechtigungen für EPCIS-Repositories, auf Basis der erweiterten Berechtigungssprache aidXACML.

Die Abbildung 5.2.1 zeigt hellgrün, welche Teile des Konzeptes vollständig im PIE-Prototypen implementiert sind, für EPCIS-Events, die eine Geschäftstransaktion referenzieren. Die grau unterlegten Komponenten der PIE-Konzeption wurden nicht in der prototypischen Implementierung umgesetzt. Im Rahmen der Entwicklung der PIE erfolgt die Einbindung des Prototypen in den EPCIS Capture Client des Fosstrak EPCIS-Projekt. Die implementierten Komponenten setzen die konzeptionell beschriebenen Programmabläufe um.

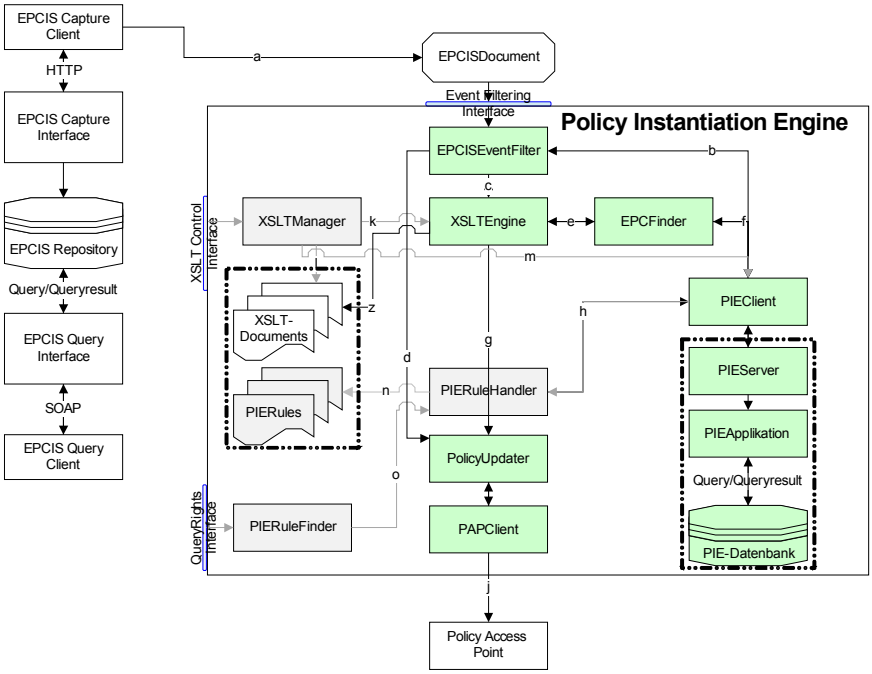

Abbildung 5.2.1 Prototyp der PIE

5.2.1. Technologie

Für Entwicklung des Prototypen wurde die objektorientierte Programmiersprache Java 1.6 verwendet.

Java ist weitestgehend plattformunabhängig. Entwickelte und kompilierte Java-Programme sind somit unverändert auf unterschiedlichen Hardware-Plattformen lauffähig [Jes00].

Die Einbindung der PIE in schon existierende Java-Projekte, wie das EPCIS-Projekt von Fosstrak, ist direkt mittels Instanziierung der EPCISEventFilter-Klasse oder über Webservice möglich. Für die Umsetzungen von Webservices bietet die Java-Plattform eine einfache Implementierung durch Annotationen an. Des Weiteren kann die Verarbeitung, Manipulation und Validierung von XML-Daten über Java-Objekte durch das JAXB-Framework einfach realisiert werden. Das open-source Framework Hibernate stellt die Lösung für das Objekt/ Relationale-Mapping (O/R-Mapping) dar. Das zentrale Element des Konzeptes ist die Abbildung von einfachen Klassen (Pojo) auf Tabellen und Spalten einer relationalen Datenbanken. Hibernate ermöglich die einfache Integration, Bearbeitung und Löschung von Daten in dieser Datenbanken. Die Erweiterung von XSLT durch Java ist notwendig, für die vollständige Formulierung der Zugriffsberechtigungen. Bei dieser Formulierung wird für jedes EPC-Objekt, unter Verwendung der EPCFinder-Klasse des Paketes pie.epcfinder, die zugehörigen EPCs ermittelt. Zur XSL-Transformation wird das JDOM-Framework und das XALAN-Framework genutzt. Das XALAN-Framework ist für die Java-Erweiterung von XSLT, siehe Kapitel 2.4.3., wichtig.

Die PIE-Datenbank ist eine MySQL-Datenbank und es wird der Datenbankserver MySQL 5.0 verwendet.

Die Entscheidung für diese Datenbanktechnologie wurde getroffen, weil das Fosstrak EPCIS-Projekt Grundlage für die prototypische Implementierung der PIE ist und bereits diese Technologie für das EPCIS-Repository einsetzt. Somit liegt es nahe die bereits vorhandene Datenbanktechnologie zu verwenden und die PIE-Datenbank ebenfalls als MySQL-Datenbank aufzusetzen. In der Praxis ist die Anpassung der PIE-Datenbank an das zu integrierende System vorzunehmen, da es aus Kostengründen, wie Anschaffungs-, Bereitstellungs-, Lizenz- sowie Wartungskosten, in einem Unternehmen günstiger ist die bereits vorhandenen Technologien zu nutzen und eine einheitliche Datenbank-technologie zu verwenden.

5.2.2. Programmpaket des PIE-Prototypen

Die Programmpakete des Prototypen sind in Abbildung 5.2.1 dargestellt. Die Gliederung wurde entsprechend der Komponenten des PIE-Konzeptes vorgenommen. Für die Komponenten, EPCIS-EventFilter, XSLT-Engine, EPC-Finder und Policy-Updater, existiert jeweils ein Paket. Die enthaltenen Java-Klassen eines Paketes einer Komponente realisieren deren Funktionalität, wie sie im Konzept der PIE beschrieben ist. Im webclient-Paket sind die Client-Klassen für den PAP und die PIE-Datenbank zu finden. Die XSLT-Stylesheets für die zu berücksichtigenden Geschäftstransaktionen sind im xslt-Paket beinhaltet. Die PIE-Datenbank ist eine Web Application, die auf einem Glasfish Server läuft. Über den PIE-Webservice steht die benötigte Funktionalität, zur Interaktion mit der PIE Datenbank, zur Verfügung.

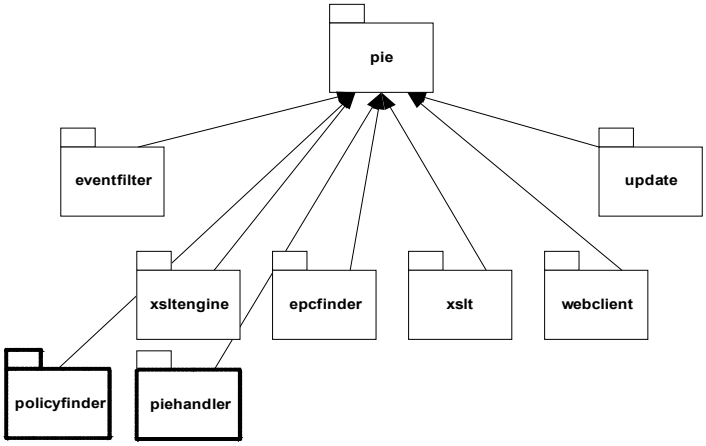

Abbildung 5.2.2 PIE-Paketstruktur

5.2.2.1. Systemintegration

Die PIE kann in EPCIS-Systeme, durch die Verwendung des EPCIS Event Filtering Interface, integriert werden (siehe Abbildung 5.2.3). Eine weiter Möglichkeit ist, die PIE als Java Application auf einem Server als Webservice bereitzustellen und durch einen entsprechenden PIE-Client die Integration in EPCIS-Systeme vorzunehmen.

In beiden Integrationsvarianten erfolgt die Einbettung der PIE, nach der erfolgreichen Speicherung der EPCIS-Events im EPCIS-Repository, um den Event-Integrations-Prozess nicht zu verzögern. Die EPCIS-Events werden in Form des EPCIS-Dokumentes an die PIE übermittelt. Die Prüfung der EPCIS-Events auf semantische und syntaktische Fehler wird nicht durch die PIE vorgenommen, da diese Prüfung bereit vor der Speicherung des EPCIS-Events in das EPCIS-Repository erfolgte.

```
import de.tu.dresden.inf.senera.pie.eventfiltering.EPCISEventFiltering;

...{   ...
       EPCISEventFiltering pie = new EPCISEventFiltering();
       pie.posttoPIE(data);
       ...}
```

Abbildung 5.2.3 PIE-Integration

5.2.2.2. Geschäftstransaktionen der PIE

Die Geschäftstransaktionen, die im Rahmen der PIE für die Formulierung der Zugriffsberechtigungen und den Berechtigungsentzug zu berücksichtigen sind, besitzen ein XSLT-Stylesheet und sind in der GRforTransaction-Tabelle der PIE-Datenbank mit deren Kurzbezeichnung enthalten. Die Kurzbezeichnung ist die Bezeichnung, mit der die Geschäftstransaktion im BizTransaction-Wert referenziert wird. Das XSLT-Stylesheet für eine Geschäftstransaktion ist im xslt-Paket unter deren Kurzbezeichnung gespeichert. Zum Beispiel die Geschäfts-transaktion „Bestellung“ wird im BizTransation-Wert mit po, wie in Tabelle 3.2.3 gezeigt, referenziert. In der GRforTransaction-Tabelle der PIE-Datenbank existiert ein Eintrag in der Spalte transaction mit dem Wert po und ein XSLT-Stylesheet im xslt-Paket mit po.xslt.

Im Prototypen können Veränderungen am Inhalt der GRforTransaction-Tabelle der PIE-Datenbank, mit den Methoden setgrft und deletegrft, über den Webservice-Tester vorgenommen werden.

Abbildung 5.2.4 Webservice-Tester der PIE-Datenbank

5.2.2.3. Sequenzdiagramme der PIE

Die folgenden Sequenzdiagramme verdeutlichen die Prozessabläufe der PIE (siehe Kapitel 4.) und veranschaulichen die Verwendung der Klassen und Pakete innerhalb der PIE. Die Klasse PAPOperation stellt Methoden zur Entscheidung, zur Integration bzw. zur Löschung von Berechtigungen zur Verfügung und wird von den Java-Klassen, Update und Delete, verwendet. Die PAPOperation-Klasse wird nicht explizit in den Diagrammen dargestellt. Die OperationList-Klasse wird zur Konvertierung und zum Vergleich von Listen benutzt und ist daher für die Darstellung von Prozessabläufen der PIE nicht relevant. Diese Diagramme sind informativ für die spätere Weiterentwicklung der PIE gedacht und werden im Folgenden nicht näher betrachtet. Weitere Informationen zu Klassen und Methoden ist in der Java-Dokumentation des Prototypen der PIE zu finden.

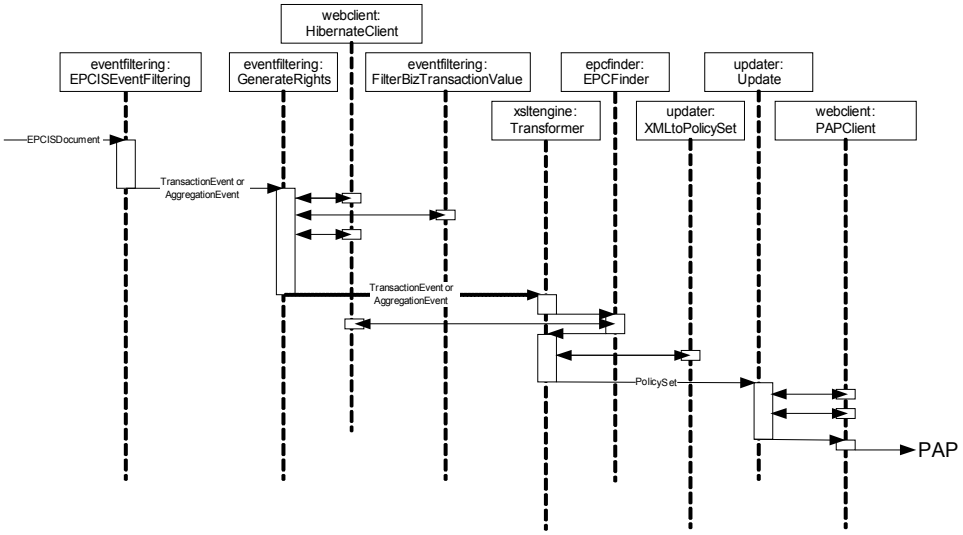

Abbildung 5.2.5 Berechtigungsvergabe

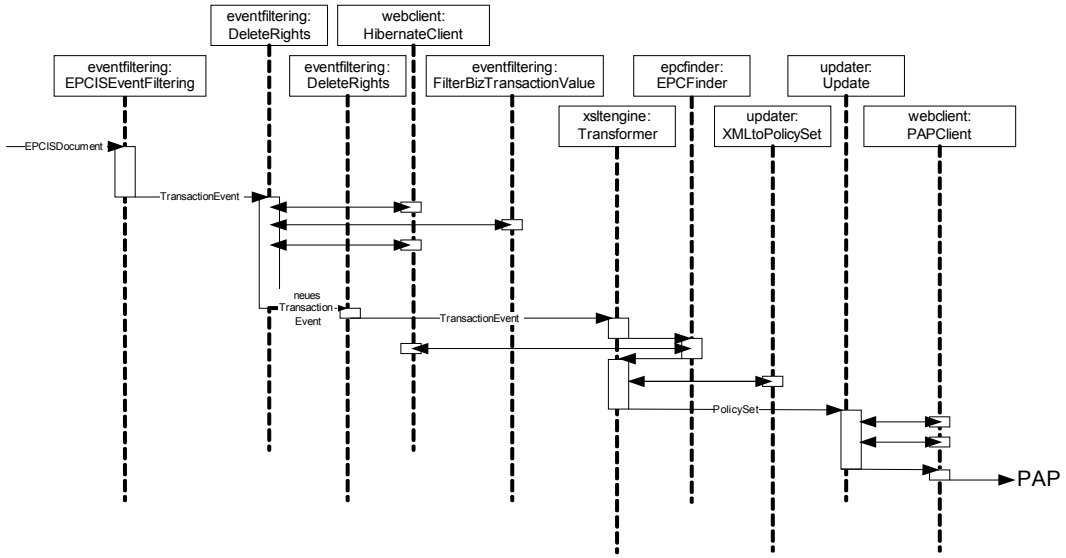

Abbildung 5.2.6 teilweiser Berechtigungsentzug

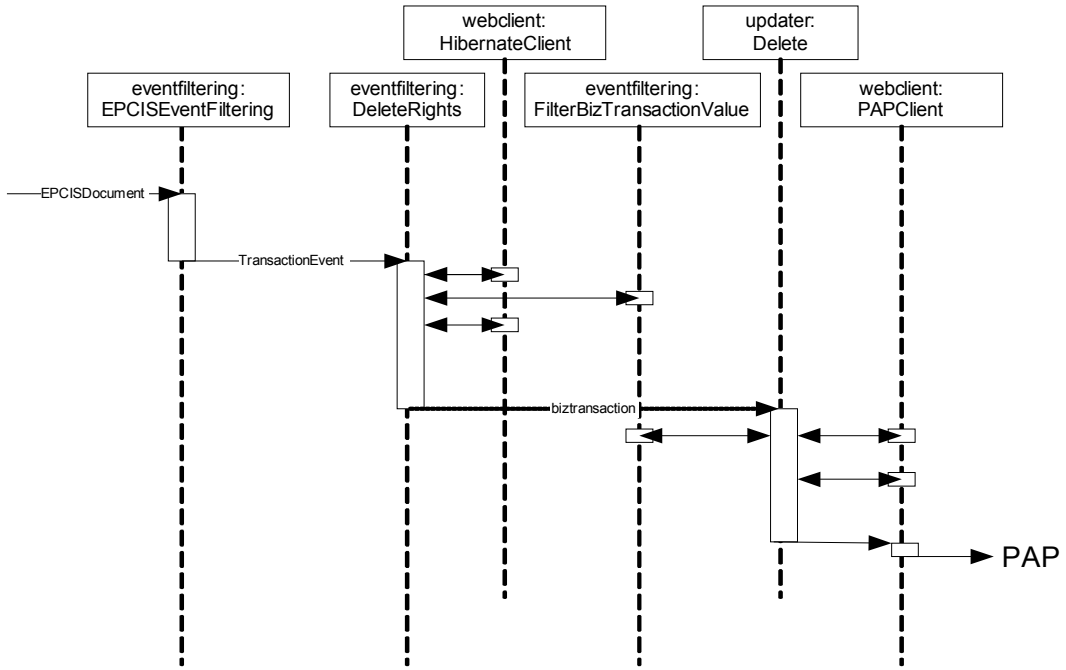

Abbildung 5.2.7 vollständiger Berechtigungsentzug

5.3. PIE-Datenbank

Die PIE-Datenbank ist eine MySQL-Datenbank. Das Schema der PIE-Datenbank ist im Anhang 8.9. abgebildet. Sie umfasst Daten, die für die Formulierung und den Entzug von Zugriffsberechtigungen benötigt werden. Um. die PIE-Datenbank verwenden zu können wurde ein separates Projekt HibernateDB erstellt, welches die Hibernate Web Application für die PIE-Datenbank enthält. Die Übermittlung der Daten zwischen der PIE und der PIE-Datenbank Application erfolgt mittels Webservice.

Hibernate wird in immer mehr Software-Architekturen als Persistenz-Framework eingesetzt. Es bietet neben den Persistenzfunktionen ebenfalls Unterstützungen für das Caching, das Clustering, das Transaktionsmanagement, sowie Recovery Strategien an [OLW08], [Hen08].

Im Weiteren wird das Hibernate-Konzepte im Rahmen der Implementierung der Hibernate Web Application präsentiert und anschließend auf die Struktur des HibernateDB-Projektes eingegangen.

5.3.1. Hibernate-Konzept und -Implementierung

Die Abbildung 5.3.1 zeigt die verschiedenen Konzepte, die von Hibernate verwendet werden [OLW08]. Diese vier Konzepte werden im Folgenden, bezogen auf die Implementierung des Projekt HibernateDB, vorgestellt

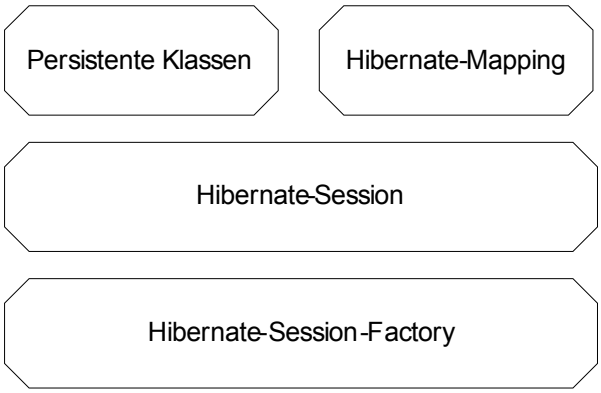

Abbildung 5.3.1 Überblick über die Hibernate-API [OLW08]

5.3.1.1. Persistente Klassen

Die persistenten Java-Klassen spielen eine zentrale Rolle beim O/R-Mapper. Ihr Aufbau ist sehr einfach. Sie umfassen eine Sammlung von Eigenschaften und enthalten entsprechende Getter- und Setter-Methoden für diese Eigenschaften. Des Weiteren besitzt jede persistente Klasse mindestes einen Konstruktor, den Default-Konstruktor, damit kann Hibernate die Klasse später instanziieren. Die persistenten Klassen enthalten die Daten, die in der Datenbank gespeichert sind bzw. gespeichert werden sollen [OLW08].

In Abbildung 5.3.2 wird die BizTransaction-Klasse als Beispiel einer persistenten Java-Klasse gezeigt. Sie besitzt eine eindeutige ID, den Namen einer Geschäfts-transaktion und die, für diese Geschäftstransaktion, im TransactionEvent angegebenen EPC-Objekte. Es handelt sich um ein einfaches Java-Objekt, welches von keiner Hibernate-Klasse abgeleitet wird.

```
package de.tu.dresden.inf.senera.pie.hibernate;

import java.util.List;

public class BizTransaction {

    public long id;
    public String biztransaction;
    public List<String> epcs;

public BizTransaction(){}

public BizTransaction(String biztransaction){
    this.biztransaction=biztransaction;}

public BizTransaction(String biztransaction, List<String> epcs){
    this.biztransaction=biztransaction;
    this.epcs=epcs;}

    public String getBiztransaction() {
return biztransaction;}

    public void setBiztransaction(String biztransaction) {
        this.biztransaction = biztransaction;}

    public List<String> getEpcs() {return epcs;}

    public void setEpcs(List<String> epcs) {this.epcs = epcs;}

    public void addEpc(String epc) {getEpcs().add(epc);}

    public long getId() {return id;}

    public void setId(long id) {this.id = id;}
}
```

Abbildung 5.3.2 persistente BizTransaction-Klasse

5.3.1.2. Hibernate-Session

Die primäre Schnittstelle zu Hibernate für Entwickler, ist die Hibernate-Session[6]. Unter der Verwendung der Session können Objekte in die Datenbank gespeichert, aus der Datenbank gelesen, sowie gelöscht und die Änderungen an Objekten persistiert werden. Die wichtigsten Methoden der Hibernate-Session sind: *Session.load(Class,Object), Session.createQuery(String), Session.save (Object), Session.delete(Object) und Session.flush().* Eine Hibernate-Session wird mittels Hibernate-Session-Factory erzeugt [OLW08].

5.3.1.3. Hibernate-Session-Factory

Die erstellte Hibernate Web Application für die PIE-Datenbank enthält genau eine SessionFactory[7], die als *tread-safe* und als unveränderlich angesehen werden kann. Sie wird beim Start der Web Application einmalig instanziiert und konfiguriert. Die Konfiguration der SessionFactory erfolgt mit der XML-Datei, die im Klassenpfad, unter dem Namen hiebernate.cfg, zu finden ist. In dieser Datei sind die Einstellungen zur Datenbankverbindung und zusätzliche Einstellungen zu Hibernate beinhaltet, sowie die gemappten Klassen werden aufgelistet [OLW08]. Abbildung 5.3.3 zeigt die Konfigurationsdatei für das erstellte HibernateDB-Projekt.

6) org.hibernate.Session
7) org.hibernate.SessionFactory

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate
Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-
configuration-3.0.dtd">

<hibernate-configuration>
 <session-factory>

<!-- Datenbank Connection Einstellungen -->
  <property name="hibernate.connection.driver_class">
   com.mysql.jdbc.Driver</property>
  <property name="hibernate.connection.url">
   jdbc:mysql://localhost:3306/pie</property>
  <property name="hibernate.connection.username">pie</property>
  <property name="hibernate.connection.password">pie</property>
  <property name="hibernate.dialect">
   org.hibernate.dialect.MySQLDialect</property>

<!-- Zusätzliche Hibernat-Properties-->
  <property name="hibernate.current_session_context_class">
   thread</property>
  <property name="hibernate.show_sql">true</property>

<!-- Datenbank Connection Einstellungen -->
  <mapping resource=
   "de/tu/dresden/inf/senera/pie/hibernate/GRforTransaction.hbm.xml"/>
  <mapping resource=
   "de/tu/dresden/inf/senera/pie/hibernate/PIERule.hbm.xml"/>
  <mapping resource=
   "de/tu/dresden/inf/senera/pie/hibernate/BizTransaction.hbm.xml"/>
  <mapping resource=
   "de/tu/dresden/inf/senera/pie/hibernate/EPC.hbm.xml"/>

 </session-factory>
</hibernate-configuration>
```

Abbildung 5.3.3 XML-Konfigurationsdatei der SessionFactory

5.3.1.4. Hibernate-Mapping

Die Grundidee vom O/R-Mapping ist die Abbildung von objektorientierten Klassen und ihren Eigenschaften auf Tabellen und Spalten einer relationalen Datenbank. Das bedeutet vereinfacht, dass ein O/R-Mapper einen Datensatz aus der Datenbank durch ein entsprechendes Java-Objekt darstellt. Änderungen am Java-Objekt wirken sich somit auf den entsprechenden Inhalt einer Datenzeile aus [OLW08].

O/R-Mapper erzwingen keine starre 1:1 Beziehung von Klassen zu Tabellen und von Eigenschaften zu Spalten dieser Tabelle. Die Definition der Abbildung zwischen der Java- und der Datenbankwelt erfolgt in XML-Dateien, die dieses Mapping spezifizieren. Diese werden für die Hibernate-Konfiguration benötigt [OLW08].

Die Abbildung 5.3.4 zeigt das Mapping zwischen der BizTransaction-Klasse, die vereinfacht dargestellt ist, und den Tabellen, Transaction und TransactionEPC, der PIE-Datendank. Die Eigenschaften *id* und *biztransaction* dieser Klasse werden auf die gleichnamigen Spalten der Transaction-Tabelle abgebildet. Die Liste *epcs* repräsentiert eine Sammlung von Eigenschaften. Sie kann mehrere EPCs als String enthalten. Jede Eigenschaft *epc* der Eigenschaftsmenge *epcs* wird auf die gleichnamige Spalte der Transaction-Tabelle abgebildet.

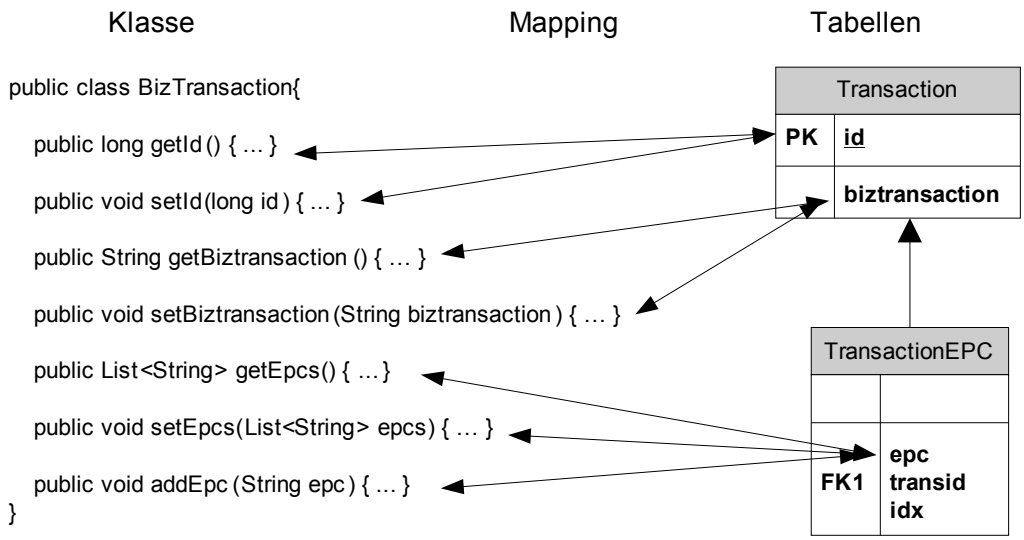

Abbildung 5.3.4 Abbildung zwischen der Klasse und deren Tabellen durch Mapping

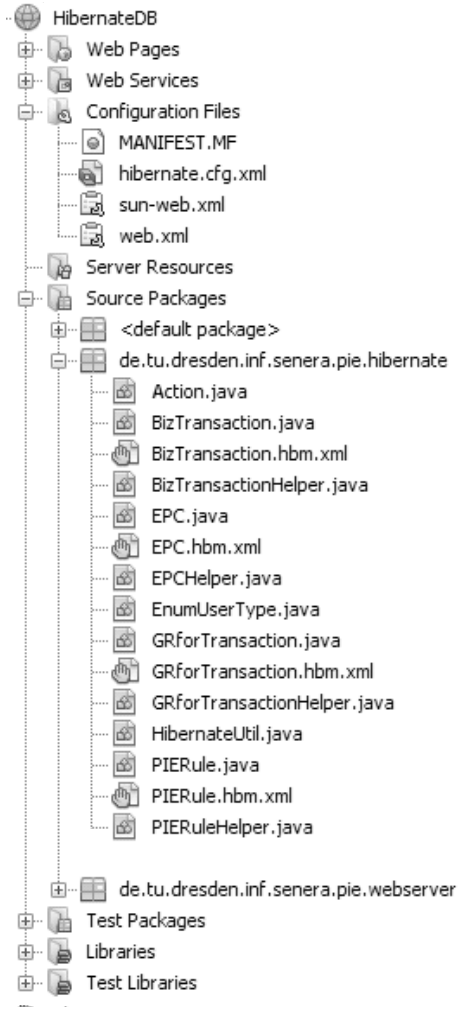

Abbildung 5.3.5 Projektstruktur

5.3.2. Projektstruktur von HibernateDB

Die Struktur des HibernateDB-Projektes ist in Abbildung 5.3.5 dargestellt. Das hibernate-Paket umfasst die persistenen Klassen und deren XML-Mapping-Dateien. Zur jeder persistenen Klasse gehört eine entsprechende Helper-Klasse, die die Methoden zur Datenmanipulation der gemappten PIE-Datenbank-Tabellen zur Verfügung stellt. Die Helper-Klassen und die HibernateUtil-Klasse, die zur Instanziierung der SessionFactory verwendet wird, sind ebenfalls im hibernate-Paket beinhaltet. Die Methoden der Helper-Klassen, die die PIE für die Realisierung der geschäftstransaktionsbasierten Formulierung der Zugriffs-berechtigungen benötigt, werden der PIE mittels Webservice zur Verfügung gestellt. Die Operationen für den Webservice sind in der Hibernate-Klasse des webserver-Pakete definiert .

5.4. Integration der EPC-Ermittung in XSLT-Stylesheets

In diesem Kapitel wird die Integration der EPC-Ermittlung in XSLT-Stylesheets präsentiert. Sie wird benötigt, um relevante EPCs bei der Berechtigungs-formulierung von Zugriffsberechtigungen berücksichtigen zu können. So das für jedes EPC-Objekt, welches in einem TransactionEvent enthalten ist, bestimmt werden kann, auf welche aggregierten EPC-Objekte eine weitere Zugriffs-berechtigung vergeben werden soll.

Die Entscheidung, die EPC-Ermittlung im Rahmen des Transformations-prozesses von Zugriffsberechtigungen vorzunehmen, liegt darin begründet, dass XSLT dem Administrator die Möglichkeit bietet den EPC-Ermittlungsprozess individuell zu steuern. Er kann Ausnahmen und Sonderfälle mittels XSLT definieren und hat somit eine große Flexibilität in der Formulierung von Zugriffsberechtigungen in Form von XSLT-Stylesheets.

Die EPCFinder-Klasse des EPC-Finders befindet sich im epcfinder-Paket. Sie stellt dem Administrator vier Methoden für die EPC-Ermittlung, die wie in Kapitel 4.8.1. beschrieben beschränkt werden kann, zur Verfügung.

Diese Methoden sind:

- *findallEpc(String epc)*
- *findallEpc(String epc, int permitdepth)*
- *findallEpc(String epc, String epcclasslist)* und
- *findallEpc(String epc, int permitdepth, String epcclasslist).*

Sie unterscheiden sich in der Angabe der Beschränkungen. Die Aggregations-tiefe wird für ein EPC-Objekt mittels *permitdepth* beschränkt. Sie gibt an bis zur welcher Aggregationsebene die EPC-Ermittlung erlaubt ist. Die EPC-Objekt-klassen, die bei der Berechtigungsvergabe nicht zu berücksichtigen sind, werden durch *epcclasslist* angegeben. Um die oben genannten Methoden für die EPC-Ermittlung in XSLT nutzen zu können, erfolgt die Einbettung des EPC-Finders wie in Abbildung 5.4.1 dargestellt. Die Erläuterungen zur Java-Erweiterung von XSLT ist im Kapitel 2.4.3. zu finden.

Im XSLT-Stylesheet werden die Beschränkungsparameter mit XSL-Variablen, wie im Beispiel der Abbildung 5.4.2 gezeigt, definiert. Die *permitdepth*-Variable repräsentiert ein Integer. Die *epcclasslist*-Variable ist ein String, der eine Aufzählung von EPC-Objektklassen enthält. Diese EPC-Objektklassen werden

dem EPC-Finder in der Form: *hiddenepcclass$_1$,hiddenepcclass$_2$,...,hiddenepcclass$_n$* übermittelt. Jedes hiddenepcclass-Element bezeichnet eine nicht zu berücksichtigende EPC-Objektklasse.

Die gewünschte Methode des EPC-Finders für die EPC-Ermittlung wird im select-Attribut des value-of-Elementes aufgerufen und gibt einen String zurück. Der zurückgegebene String beinhaltet die Menge der vollständig in XML formulierten EAL-Condition-Elemente der zu berücksichtigenden EPCs eines EPC-Objektes, welches in einem TransactionEvent enthalten ist. Die in XML-Notation erstellten EAL-Condition-Elemente werden in das EAL-ConditionSet mittels value-of-Element eingebunden. Dieses Element besitzt das disable-output-escaping-Attribute mit dem Wert yes. Das Attribute ist erforderlich, damit bei der Integration der EAL-Condition-Elemente die Tag-Anfangszeichen bzw. -Endzeichen nicht durch < bzw. > von XSLT-Parser ersetzt werden.

```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xmlns:eal="http://inf.tu-dresden.de/eal/1.0/schema"
xmlns:xalan="http://xml.apache.org/xalan"
xmlns:java="http://xml.apache.org/xslt/java" exclude-result-
prefixes="xalan java"
xmlns:my-epcfinder="de.tu.dresden.inf.senera.pie.epcfinder.EPCFinder"
extension-element-prefixes="my-epcfinder">

<xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>

<xalan:component
 prefix="my-epcfinder" elements="init" functions="findallEpc">

  <xalan:script lang="javaclass"
   src="xalan://de.tu.dresden.inf.senera.pie.epcfinder.EPCFinder">
  </xalan:script>

</xalan:component>

<my-epcfinder:init/>
```

Abbildung 5.4.1 Integration EPC-Finder

```
  <xsl:for-each select="//epc">

    <xsl:variable name="epc" select="string(.)"/>
    <xsl:variable name="permitdepth">2</xsl:variable>
    <xsl:variable name="hiddenepcclass">
     urn:epc:id:sgtin:0057000.111111, urn:epc:id:sgtin:0057000.222222
    </xsl:variable>

    <eal:condition>
      <eal:predicate operator="epc-match">
        <xsl:value-of select="$epc"/>
      </eal:predicate>
    </eal:condition>

    <xsl:value-of select=
     "my-epcfinder:findallEpc($epc,$permitdepth,$hiddenepcclass)"/>

  </xsl:for-each>
```

Abbildung 5.4.2 Verwendung des EPC-Finders

6. Bewertung und Validierung

Im Rahmen der Bewertung und Validierung wird die entwickelte Konzeption der PIE im Zusammenhang mit der Umsetzung des Prototypen zusammenfassend präsentiert. Anschließend erfolgt die kritische Auseinandersetzung mit der erstellten Konzeption und die Bewertung des Konzeptes der PIE. Im Weiteren wird der Prototyp bewertet und dessen Grundfunktionalität anhand von konkreten Beispielen veranschaulicht.

6.1. Konzeption

Das entwickelte Konzept der PIE realisiert die geschäftstransaktionsbasierte Formulierung und den geschäftstransaktionsbasierten Entzug von Zugriffsberechtigungen. Geschäftstransaktionen, die im EPCIS-Repository gespeichert werden, werden im standardisierten EPCIS-Dokument an die PIE übermittelt. Jedes im EPCIS-Dokument enthaltene EPCIS-Event prüft die PIE, ob dieses EPCIS-Event die Vergabe oder den Enzug von Berechtigungen auslöst. Diese Überprüfung bzw. Filterung von EPCIS-Events, siehe Kapitel 4.6., erfolgt durch die Auswertung der EPCIS-Eventdaten und durch die in der PIE-Datenbank gesammelten Informationen. Die Berechtigungsvergabe, die Berechtigungserweiterung und der teilweise Berechtigungsentzug wird mittels XSL-Transformation des eingehenden EPCIS-Events bzw. des durch die PIE neu generierten TransactionEvents in das PIE-PolicySet (siehe Kapitel 4.3.3.) durchgeführt. Das PIE-PolicySet enthält die neu formulierten Zugriffsberechtigungen, die anschließend mittels PIE-Integrationsstrategie (siehe Kapitel 4.10.1.) im PAP durchgesetzt werden. Der vollständige Berechtigungs-entzug erfolgt ohne XSL-Transfomation. Anhand des BizTransactions-Wertes des TransactionEvents wird der Entzug mittels PIE-Deintegrationsstrategie durch die PIE am PAP vorgenommen. Die prototypische Umsetzung der PIE ist für die PolicySet-Struktur des PAP der Abbildung 4.3.2b implementiert. Durch die Verwendung dieser PolicySet-Struktur wird eine klare Trennung der manuellen und automatischen Berechtigungsbearbeitung erzielt. Diese strikte Trennung ist für die effiziente und korrekte Funktionsweise der PIE wichtig, um Differenzen die zwischen der manuellen und automatischen Berechtigungsformulierung auftreten können, als Fehlerquelle auszuschließen. Durch die Verwendung von XSLT-Stylesheets steht dem Administrator eine formale Beschreibungssprache, basieren auf Templates, für die Formulierung von Berechtigungen zur Verfügung. XSLT bietet ein enormes Potenzial, um einfache und komplexe Berechtigungen, sowie Ausnahmen und Sonderfälle formal zu definieren. Diese Flexibilität von XSLT ermöglicht ebenfalls aggregierte EPCs zu EPC-Objekten, im Rahmen der Berechtigungsgenerierung, mit einzubeziehen. Das erstellte PIE-Konzept legt die grundlegende Struktur der XSLT-Stylesheets fest. Die individuelle Gestaltung der Berechtigungen in dem EAL-eventSet erfolgt durch den Administrator.

Die PIE-Konzeption beinhaltet das vollständig umsetzbare Lösungskonzept für die automatische Berechtigungsgenerierung, unter Verwendung des Kontextes der Geschäftstransaktionen. Dieses Konzept ist nutzerorientiert, indem es individuell auf die Bedürfnisse und Unternehmensinteressen angepasst werden kann. Änderungen der Geschäftstransaktionen, für die Zugriffsberechtigungen erstellt bzw. entzogen werden sollen, und die formale Berechtigungsdefinition die der Berechtigungsgenerierung zu Grunde liegt, ist einfach an die neuen

Nutzerbedürfnisse an zu passen. Diese Anpassung kann von Administrator während der Laufzeit der PIE vorgenommen werden. Entsprechende Routinen wurden dafür im Konzept vorgeschlagen. Die PIE beinhaltet eine Triggerlogik zum Auslösen der Berechtigungsvergabe bzw. des Berechtigungsentzuges, die durch Vorfilterung der EPCIS-Events im EPCIS-System optimiert werden kann. Die PIE benötigt für die Durchführung der Vergabe und des Entzuges von Zugriffsberechtigungen nicht nur Informationen des auslösenden Events, sondern weiterführende Informationen. Informatioen wie: Wurde für die vorliegende Transaktion bereits eine Berechtigung erstellt oder handelt es sich bei dem Berechtigungsentzug, um einen vollständigen oder teilweisen Entzug? Um der PIE schnell und effizent die benötigten Informationen zur Verfügung zu stellen, sowie die Performenz des EPCIS-Repository und des PAP durch Abfragen der PIE nicht unnötig zu beeinträchtigen, ist die PIE-Datenbank entwickelt worden. Die Kosten für die PIE-Datenbank sind im Verhältnis zu der dadurch erreichten Effizenzsteigerung der PIE und Perfomenzersparnis vertretbar. Die Vergabe bzw. der Entzug von Berechtigungen ist erfolgreich durchgesetzt, wenn die PIE diese Veränderungen an die PIE-Datenbank und an den PAP übermittelt. Kann eine ausgelöste Veränderung an Berechtigungen nicht durchgeführt werden erfolgt ein entsprechender Eintrag des Fehlers in die PIE-Log-Datei und der Administrator ist über dieses Fehlverhalten der PIE informiert. Die Prüfung der Berechtigungsstylesheets kann extern bzw. intern mittels XSLT-Manager erfolgen. Die XSLT-Stylesheets enthalten die Berechtigungslogik und sind für die korrekte Berechtigungsformulierung verantwortlich. Die Validierung dieser formalen Berechtigungsmuster ist die Validierung für die PIE, die die XSLT-Stylesheets entsprechend der enthaltenen Entscheidungslogik nutzt und die formulierten Berechtigungen am PAP durchsetzt. Der Berechtigungsentzug wird im Konzept der PIE permanent oder periodisch zusammen mit der Berechtigungsvergabe am PAP durchgeführt. Bei einem Berechtigungsentzug, der periodisch im Rahmen einer permanenten Aktualisierungsstrategie für die Vergabe von Berechtigungen am PAP erfolgen soll, muss die Art des Berechtigungsentzuges der PIE zugänglich gemacht werden und das Konzept dementsprechend erweitert werden. Zum Zeitpunkt der Entwicklung des PIE-Konzeptes ist die Art des Berechtigungsentzuges, für eine periodische Strategie der Aktualisierung des Berechtigungsenzuges losgelöst von der Strategie der Berechtigungsvergabe, nicht entscheidbar. Das bedeutet, das die PIE mit den ihr zur Verfügung stehenden Informationen nicht entscheiden kann, ob es sich um einen erfolgreichen Abschluss oder um den Abbruch einer Geschäftstransaktion handelt. Des Weiteren ist die Aggregations-auflösung und die Aggregationslöschung nicht mit den gegeben Informationen unterscheidbar. Diese Informationen sind durch externe Abfragen oder die Einbeziehung des Administrators zu ermitteln. Die Entscheidung zur Art des Berechtigungsentzuges bzw. zur Art der Aggregationauflösung ist vom externen System abhängig, das Konzept ist dementsprechend zu erweitern. Diese Konzepterweiterung stellt keinen Nachteil dar.

Die PIE-Konzeption erfüllt die an die geschäftstransaktionsbasierte Formulierung gestellten Anforderungen. Es beinhaltet zusätzlich den Berechtigungsentzug, basierend auf dem Kontext der Geschäftstransaktionen. Dem Administrator steht mittels XSLT-Stylesheets eine abstrakte Form der Formulierung von Zugriffsberechtigungen zur Verfügung. Die PIE realisiert die automatische Umsetzung der abstrakt formulierten Berechtigungsregeln.

6.2. PIE-Prototype

Der PIE-Prototyp setzt die wesentlichen Teile des entwickelten PIE-Konzeptes um. Er realisiert vollständig die eventbasierte Berechtigungsdefinition unter Verwendung des semantisch reicheren Konzeptes der Geschäftstransaktionen, für EPCIS-Events, die eine Geschäftstransaktion referenzieren. Die implementierte PIE enthält die entwickelten Programmabläufe des PIE-Konzeptes und zeigt, das dass PIE-Konzept umsetzbar ist.

Mittels XSLT-Stylesheets steht dem Administrator des EPCIS-Repositories eine komfortablere und abstraktere Form der geschäftstransaktionsbasierten Formulierung von Berechtigungen zur Verfügung. Der administrative Aufwand für die Berechtigungsformulierung wurde durch die PIE gemindert und vereinfacht, wie im nachfolgende Abschnitt erläutert.

Für jede zu berücksichtigende Geschäftstransaktion ist ein XSLT-Stylesheet definiert, welches entsprechende Berechtigungsregeln für die Berechtigungsformulierung der PIE beinhaltet. Die Definition dieser Regeln wird vom Administrator, durch die entsprechende Erweiterung des XSLT-Stylesheets (siehe Anhang 8.8.), vorgenommen. Der Aufwand den der Administrator für die Erstellung und die Prüfung des XSLT-Stylesheets, sowie der Integration des XSLT-Stylesheets in die PIE und gegebenenfalls für die Aktualisierung des XSLT-Stylesheets infolge von Veränderungen in der Berechtigungspolitik des Unternehmens hat, repräsentiert in Summe den administrativen Aufwand für die PIE bzw. für die Formulierung von Berechtigungen. Bei der manuellen Formulierung entsteht für jedes EPCIS-Events, das zu berücksichtigende Geschäftstransaktionen für die Berechtigungsformulierung enthält, ein administrativer Aufwand. Die Differenz, bezogen auf eine Geschäftstransaktion, zwischen dem manuellen Formulierungsaufwand von Berechtigungen und dem administrative Aufwand der PIE gibt die Einsparung im Aufwand an.

Bei einem durchschnittlichen Aufwand für die manuelle Berechtigungsformulierung, bezogen auf eine Geschäftstransaktion, von ca. 12 min, im Gegensatz zu dem einmaligen Definitionsaufwand für das XSLT-Stylesheet dieser Geschäftstransaktion von ca. 140 min und einer Generierungszeit für die Berechtigungsformulierung von durchschnittlich ca. 2 min für die PIE, berechnet sich die Reduzierung des Aufwandes für den Administrator mit der Funktion: n*12-140 (n = Anzahl der EPCIS-Events[8]). Im konkreten Beispiel bedeutet das, dass sich ab dem 12. EPCIS-Event[8] der Einsatz der PIE rentiert. Ab dem 14. EPCIS-Event[8] zeichnet sich eine Ersparnis, bezogen auf die benötigte Zeit für die vollständige Berechtigungsformulierung, gegenüber der manuellen Formulierung ab. Dieses Beispiel zeigt die Effektivität des Einsatzes der PIE zur Formulierung von Berechtigungen gegenüber der manuellen Berechtigungsformulierung.

Die Funktionalität der PIE wurde anhand von ausgewählten Testfällen überprüft. Diese Überprüfungsmethode steht dem Administrator, in der praxisbezogen Integration der PIE, ebenfalls als zusätzliche Validierungsmaßnahme zur Verfügung. Die folgenden drei wesentlichen Anwendungsfälle, die Berechtigungsvergabe, der teilweise und der vollständige Berechtigungsentzug, zeigen die grundlegende Funktionalität des erstellten PIE-Prototyps.

8) EPCIS-Events, die eine Geschäftstransaktion referenzieren und die Berechtigungsformulierung auslösen.

6.2.1. Die Berechtigungsvergabe

Mittels EPCIS-Capture-Application wird das in der Abbildung 6.2.1 gezeigte TransactionEvent erzeugt. Dieses EPCIS-Dokument wird an die PIE übermittelt. Die PIE überprüft das eingehende TransactionEvent im Rahmen der Berechtigungsvergabe. Nach dieser Prüfung, wie in Abbildung 4.6.1 beschrieben, aktualisiert die PIE die PIE-Datenbank für das TransactionEvent, siehe Abbildung 6.2.2. Anschließend erstellt die PIE die entsprechenden Zugriffsberechtigungen für den User QAtracker infolge seiner Bestellung. Da im PAP für den User QAtracker noch kein PolicySet vorhanden ist, wird das von der PIE generierte PolicySet, siehe Abbildung 6.2.3, in den PAP integriert.

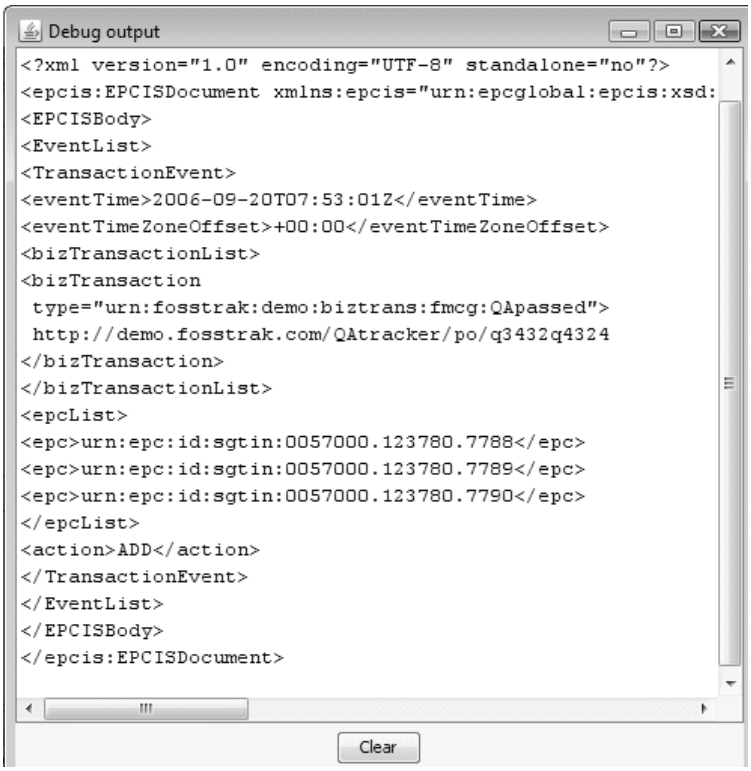

Abbildung 6.2.1 EPCIS-Dokument für die PIE

BizTransaction

id	biztransaction
1	http://demo.fosstrak.com/QAtracker/po/q3432q4324

TransactionEPC

transid	epc	idx
1	urn:epc:id:sgtin:0057000.123780.7788	0
1	urn:epc:id:sgtin:0057000.123780.7789	1
1	urn:epc:id:sgtin:0057000.123780.7790	2

Abbildung 6.2.2 Tabellen der PIE Datenbank

```
<?xml version="1.0" encoding="UTF-8"?>

<PolicySet xmlns:eal="http://inf.tu-dresden.de/eal/1.0/schema"
xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
PolicySetId="QAtracker" PolicyCombiningAlgId="http://inf.tu-
dresden.de/aidxacml/1.0/combiners/policy/eal-core-intersect">

 <Description>TransactionEvent</Description>
 <Target>
  <Subjects>
   <Subject>
    <SubjectMatch
     MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue
      DataType="http://www.w3.org/2001/XMLSchema#string">QAtracker
     </AttributeValue>
     <SubjectAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:role"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </SubjectMatch>
   </Subject>
  </Subjects>
 </Target>

<Policy PolicyId="QAtracker" RuleCombiningAlgId="http://inf.tu-
dresden.de/aidxacml/1.0/combiners/rule/eal-core-intersect">
<Target/>

<Rule Effect="PartialPermit" RuleId="po_q3432q4324">
 <Target>
  <Resources>
   <Resource>
    <ResourceMatch MatchId="http://www.inf.tu-
     dresden.de/eal/0.2/function/joins">
     <AttributeValue DataType="http://www.inf.tu-
      dresden.de/eal/0.2/schema">
      <eal:eventSet>
       <eal:eventSubset visibleAttributes="*" basicSet="ObjectEvent">
        <eal:conditionSets>
         <eal:conditionSet attribute="epcList.epc">
          <eal:condition>
           <eal:predicate operator="epc-match">
            urn:epc:id:sgtin:0057000.123780.7788
           </eal:predicate>
          </eal:condition>
          <eal:condition>
           <eal:predicate operator="epc-match">
            urn:epc:id:sgtin:0057000.123780.7789
           </eal:predicate>
          </eal:condition>
          <eal:condition>
           <eal:predicate operator="epc-match">
            urn:epc:id:sgtin:0057000.123780.7790
           </eal:predicate>
          </eal:condition>
         </eal:conditionSet>
        </eal:conditionSets>
       </eal:eventSubset>
      </eal:eventSet>
     </AttributeValue>
     <ResourceAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
      DataType="http://www.inf.tu-dresden.de/eal/0.2/schema"/>
     </ResourceMatch>
     </Resource>
    </Resources>
  </Target>
</Rule>
</Policy>
</PolicySet>
```

Abbildung 6.2.3 PolicySet des Users QAtracker im PAP

6.2.2. Der teilweise Berechtigtungsentzug

Der User QAtracker storniert zwei Produkte von seiner zuvor getätigten Bestellung. Das EPCIS-Dokument, der Abbildung 6.2.4, wird an die PIE übermittelt infolge der Bestellungsstornierung. Die PIE löst nach der enthaltenen Triggerlogik, Abbildung 4.6.1, den teilweisen Berechtigungsentzug der bereits vergeben Berechtigungen für diese Bestellung aus. Der Inhalt der TransactionEPC-Tabelle der PIE-Datenbank wird durch die PIE, wie in Abbildung 6.2.5 dargestellt, verändert. Ein neues TransactionEvent für die Aktualisierung der Berechtigungen wird generiert, siehe Abbildung 6.2.6. Die bestehenden Berechtigungen für die Bestellung sind in der Rule des User-PolicySets, die die ID q3432q4324 besitzt definiert (siehe Abbildung 6.2.3). Diese Rule wird durch die PIE im PAP-PolicySet überschrieben. Das aktualisierte PolicySet für den User QAtracker, das in den PAP zurückgespeichert wird, ist in Abbildung 6.2.7 zu sehen.

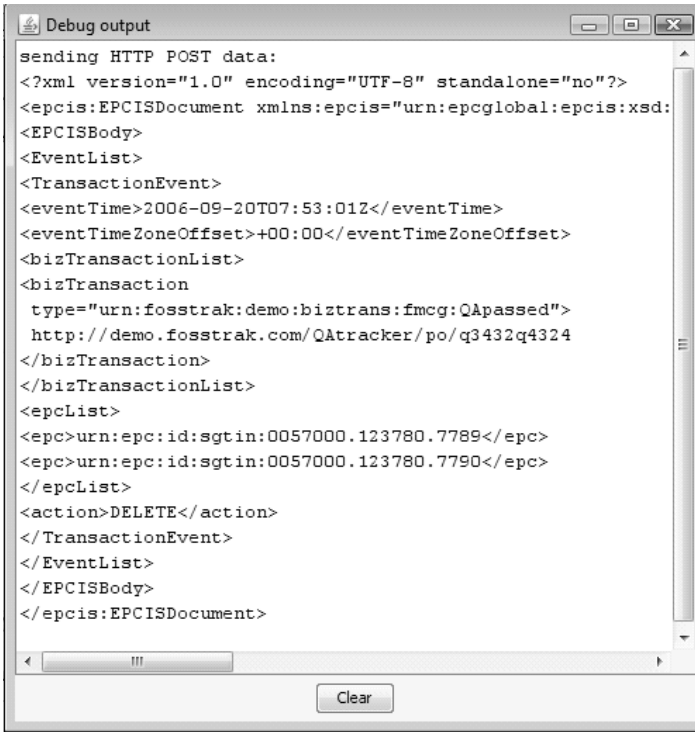

Abbildung 6.2.4 EPCIS-Dokument für die PIE

BizTransaction

id	biztransaction
1	http://demo.fosstrak.com/QAtracker/po/q3432q4324

TransactionEPC

transid	epc	idx
1	urn:epc:id:sgtin:0057000.123780.7788	0

Abbildung 6.2.5 Tabellen der PIE-Datenbank

```
<TransactionEvent
xmlns:ns2="http://www.unece.org/cefact/namespaces/StandardBusinessDocumentHe
ader" xmlns:ns4="urn:epcglobal:epcis:xsd:1" xmlns:ns3="urn:epcglobal:epcis-
query:xsd:1" xmlns:ns5="urn:epcglobal:epcis-masterdata:xsd:1">

    <eventTime>2006-09-20T07:53:01Z</eventTime>
    <eventTimeZoneOffset>+00:00</eventTimeZoneOffset>
    <bizTransactionList>
        <bizTransaction type="urn:fosstrak:demo:biztrans:fmcg:QApassed">
          http://demo.fosstrak.com/QAtracker/po/q3432q4324
        </bizTransaction>
    </bizTransactionList>
    <epcList>
        <epc>urn:epc:id:sgtin:0057000.123780.7788</epc>
    </epcList>
    <action>ADD</action>
</TransactionEvent>
<?xml version="1.0" encoding="UTF-8"?>
<PolicySet xmlns:eal="http://inf.tu-dresden.de/eal/1.0/schema"
xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" PolicySetId="QAtracker"
PolicyCombiningAlgId="http://inf.tu-dresden.de/aidxacml/1.0/combiners/policy/eal-
core-intersect">
 <Target>
  <Subjects>
   <Subject>
    <SubjectMatch
     MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">QAtracker
     </AttributeValue>
     <SubjectAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:role"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </SubjectMatch>
   </Subject>
  </Subjects>
 </Target>

<Policy PolicyId="QAtracker" RuleCombiningAlgId="http://inf.tu-
dresden.de/aidxacml/1.0/combiners/rule/eal-core-intersect">
<Target/>
<Rule Effect="PartialPermit" RuleId="po_ q3432q4324">
 <Target>
  <Resources>
   <Resource>
    <ResourceMatch
     MatchId="http://inf.tu-dresden.de/aidxacml/1.0/matchers/eal-core-match">
     <AttributeValue DataType="http://www.inf.tu-dresden.de/eal/0.2/schema">
      <eal:eventSet>
       <eal:eventSubset visibleAttributes="*" basicSet="ObjectEvent">
        <eal:conditionSets>
         <eal:conditionSet attribute="epcList.epc">
          <eal:condition>
           <eal:predicate operator="epc-match">
            urn:epc:id:sgtin:0057000.123780.7788
           </eal:predicate>
          </eal:condition>
         </eal:conditionSet>
        </eal:conditionSets>
       </eal:eventSubset>
      </eal:eventSet>
     </AttributeValue>
     <ResourceAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
      DataType="http://www.inf.tu-dresden.de/eal/0.2/schema"/>
     </ResourceMatch>
     </Resource>
    </Resources>
  </Target>
</Rule>
</Policy>
</PolicySet>
```

Abbildung 6.2.7 PolicySet des Users QAtracker im PAP

6.2.3. Die vollständige Beendigung einer Transaktion

Die Bestellung des User QAtracker wurde erfolgreich beendet und an die PIE wird das EPCIS-Dokument, der Abbildung 6.2.8, übermittelt. Die PIE löst nach der Analyse des TransactionEvents und der entsprechenden PIE-Datenbank-Tabellen, der Abbildung 6.2.5, den vollständigen Berechtigungsentzug für diese Bestellung aus. Die Inhalte der Tabellen, BizTransaction und TransactionEPC, werden bezogen auf diese Bestelltransaktion gelöscht. Für den User QAtracker sind nur die Berechtigungen, die für diese Bestellung vergeben wurden, im PolicySet des PAP enthalten. Somit setzt die PIE für den User QAtracker die Löschung seines PolicySet im PAP durch.

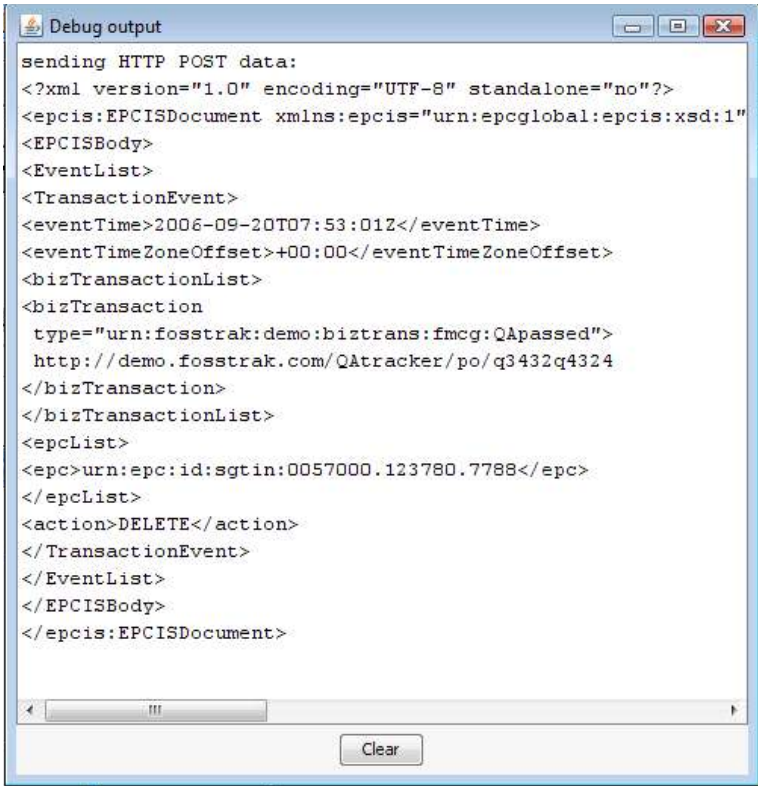

Abbildung 6.2.8 EPCIS-Dokument für die PIE

7. Zusammenfassung/Ausblick

Der Tend der automatischen Erfassung von Handelsgütern und Ladungsträgern, die mit Transpondern versehen sind, mit der RIFD-Technologie in globalen Wertschöpfungsketten, zeichnet sich ab. Die erfassten Daten werden in Form von EPCIS-Events im lokalen EPCIS-Repository der Firma, die diese Datenerfassung vornimmt, gespeichert. Der EPCIS-Standard spezifiziert die EPCIS-Eventtypen und die Austausch-Schnittstellen. Er ermöglicht die Interoperabilität zwischen den Handelspartnern einer Wertschöpfungskette und bringt somit mehr Transparenz in der Bewegung, in der Lokalisierung und in der Verteilung von mit EPC-Transponder gekennzeichneten Objekten innerhalb einer Wertschöpfungskette. Die erreichbare Transparenz ist abhängig von den Sicherheitsbedürfnissen der Handelspartner dieser Wertschöpfungskette. Besonders sensibel ist der Umgang mit Firmendaten bei im Wettbewerb stehenden Handelspartnern. Die Firmendaten des EPCIS-Repositories sind somit entsprechend des Sicherheitsbedürfnisses der Firma zu schützen. Der EPCIS-Standard lässt die Realisierung des Zugriffskontrollsystems für EPCIS-Repositories jedoch offen.

Die Entwicklung ein entsprechendes Zugriffskontrollsystems für EPCIS-Repositories wird im Rahmen eines Forschungsprojektes bei SAP Research entwickelt. Die Zugriffsberechtigungen des entwickelten EPCIS-Zugriffskontroll-systems werden in der XML-basierend Berechtigungssprache aidXACML formuliert und die effiziente Durchsetzung der Berechtigungen erfolgt mittels Query Rewriting. Die Zugriffsberechtigungen, die für ein EPCIS-Repository existieren, werden im lokalen Policy Access Points gespeichert. Sie realisieren die attributgenaue und regelbasierte Eventfreigabe für EPCIS-Repositories. Da die manuelle Formulierung dieser Berechtigungen sehr komplex ist, entstand die Idee der Automatisierung der Berechtigungsformulierung. Der Ansatz zur Vereinfachung der Berechtigungsspezifikationen beruht auf dem semantisch reicheren Konzept der Geschäftstransaktionen, die in globalen Wertschöpfungs-ketten zwischen den Handelspartnern ausgetauscht werden.

In dieser Diplomarbeit wurden relevante Techniken und Systeme für die Realisierung dieser Automatisierungsidee betrachtet. Eine Systematisierung und Analyse von Geschäftstransaktionen in Wertschöpfungsketten, sowie die Untersuchung bestehender Abhängigkeiten zwischen Geschäftstransaktionen und deren entsprechender Zugriffsberechtigungen wurde vorgenommen. Anhand der daraus erhaltenen Erkenntnisse wurden Anforderungen, die für das zu entwickelnde PIE-Konzept von Bedeutung sind, erörtert.

Das Ergebnis dieser Arbeit ist ein vollständiges und funktionales Konzept, welches eine komfortablere und abstraktere Form der Formulierung von Zugriffsberechtigungen, unter Verwendung der Semantik von Geschäfts-transaktionen für die Automatisierung, bereitstellt. Die prototypische Implementierung setzt die wesentlichen Teile des entwickelten PIE-Konzeptes um. Der PIE-Prototyp realisiert die eventbasierte Berechtigungsdefinition auf Basis der Geschäftstransaktionen für EPCIS-Events, die eine Geschäfts-transaktion referenzieren. Er setzt die entwickelten Programmabläufe des PIE-Konzeptes um und zeigt, das dass PIE-Konzept realisierbar ist.

Das erstellt PIE-Konzept stellt den konkreten Lösungsansatz dar, der die beschriebenen Limitationen (siehe Aufgabenstellung) behebt, sowie dem Administrator des EPCIS-Repositories eine komfortablere und abstraktere Form

der geschäftstransaktionsbasierten Formulierung von Berechtigungen mittels XSLT bietet. Die Flexibilität, die die Berechtigungsformulierung durch die Verwendung von aidXACML besitzt, bringt einen hohen administrativen Aufwand im Rahmen der manuellen Generierung mit sich. Dieser Aufwand wurde durch das entwickelte PIE-Konzept gemindert und vereinfacht (wie im Kapitel 6.2. beschrieben), durch die Verwendung von Berechtigungsregeln, die je nach vorliegendem Muster angewendet werden.

Die Erstellung und der damit verbundene Aufwand, der abstrakten und formalen Berechtigungsstylesheets für Geschäftstransaktionen, die im Rahmen der Berechtigungsformulierung der PIE berücksichtigt werden sollen, kann durch entsprechende Tools vereinfacht werden. Zukünftig und für den praxisbezogenen Einsatz der PIE ist die Erweiterung der PIE durch ein grafisches Tool wünschenswert, welches dem Administrator die Erstellung der XSLT-Stylesheets für Geschäftstransaktionen mittels Track & Drop ermöglicht, die Prüfung für die generierten XSLT-Stylesheets vornimmt und die Ansteuerung des XSLT-Managers über das XSLT Control Interface realisiert. Des Weiteren ist die Integration eines komplexen Monitoring-Systems denkbar, mit dem der Administrator spezielle Auswertungen für die Optimierung der PIE vornehmen kann, beispielsweise die Veränderung des Aktualisierungsintervalls für die periodische Aktualisierung von Berechtigungen. Das Monitoring-System realisiert die Überwachung der Funktionalität der PIE und bietet damit weitere Mechanismen, beispielsweise Statistiken und grafische Auswertungen, an, um ein Fehlverhalten der PIE zur erkennen.

Diese Arbeit hat gezeigt, das die Automatisierung der Berechtigungsformulierung auf Basis von Geschäftstransaktionen sehr komplex, aber realisierbar ist und die Beziehungen zwischen Geschäftstransaktionen und formulierten Berechtigungen in Abhängigkeit zu den Sicherheitsbedürfnissen der Unternehmen und deren Sicherheitspolitik stehen. Das Konzept der PIE bietet Unternehmen eine Lösung für die individuelle und nutzerorientierte automatische Realisierung der geschäftstransaktionsbasierten Berechtigungsformulierung mittels XSLT an.

8. Anhang

8.1. Policy-Modell

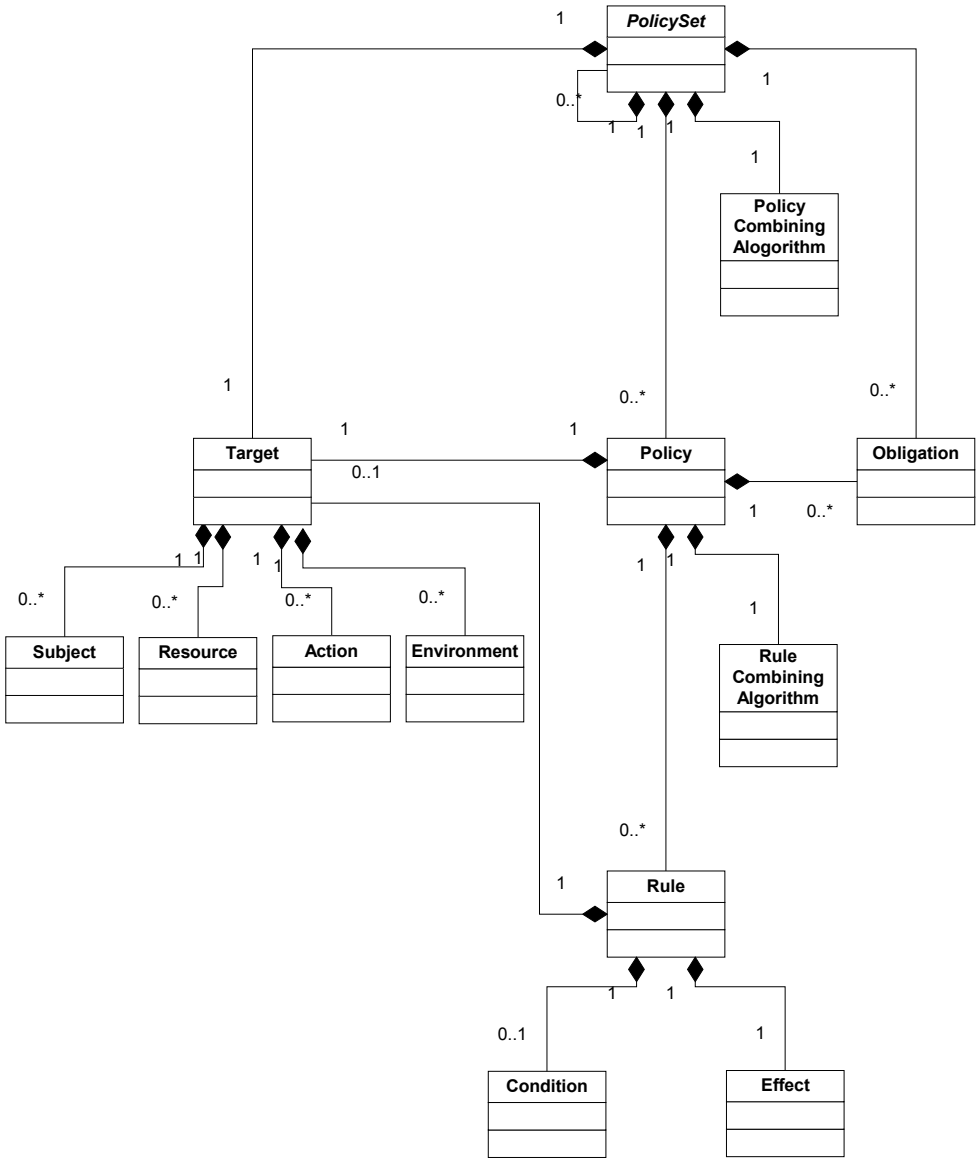

Abbildung 8.1.1 Policy-Modell [Mos05]

8.2. VocabularyList von EPCIS-Elementen

Die Tabelle 8.2.1 umfasst die EPCIS-Elemente, die vom Type anyURI sind. Sie ordnet diesen Elementen entsprechende Masterdatentypen, sowie formale URI-Namen in URN-Notation zu. Im Weiteren werden diese URN-Notationen verwendet, um spezielle Belegungen für diese EPCIS-Element zu ermitteln. Mittels *SimpleMasterDataQuery* Anfrage werden die Vocabularies zu bestimmten EPCIS-Elementen über http://rolls.mit.edu/EPCIS_REST/ abgefragt. Ausgewählte Abfrageergebnisse sind in den jeweilig nachfolgenden Abschnitten beinhaltet, und zeigen spezielle Belegungen für EPCIS-Elemente. Ergänzt werden die VocabularyElemente durch [ACC07] und [EPC07b].

Field Event Name	***Associated Master Data Type***	***Formal URI Name***
bizLocation	Business Location	`urn:epcglobal:epcis:vtype:BusinessLocation`
bizStep	Business Step	`urn:epcglobal:epcis:vtype:BusinessStep`
bizTransactionList	bizTransaction: Business Transaction	`urn:epcglobal:epcis:vtype:BusinessTransaction`
bizTransactionList	type: Business Transaction Type	`urn:epcglobal:epcis:vtype:BusinessTransactionType`
disposition	Disposition	`urn:epcglobal:epcis:vtype:Disposition`
epcClass	EPC Class	`urn:epcglobal:epcis:vtype:EPCClass`
readPoint	Read Point	`urn:epcglobal:epcis:vtype:ReadPoint`

Tabelle 8.2.1 Field Events and Associated Master Data Types [BEA06]

8.2.1. BusinessStep

```
<VocabularyList>
 <Vocabulary type="urn:epcglobal:epcis:vtype:BusinessStep">
  <VocabularyElementList>

     <!--http://rolls.mit.edu/EPCIS_REST/-->
   <VocabularyElement id="urn:epcglobal:hls:bizstep:commissioning"/>
   <VocabularyElement
    id="urn:epcglobal:hls:bizstep:casetopalletaggregation"/>
   <VocabularyElement id="urn:epcglobal:fmcg:bizstep:receiving"/>
   <VocabularyElement id="urn:epcglobal:fmcg:bizstep:picking"/>
   <VocabularyElement id="urn:epcglobal:fmcg:bizstep:shipping"/>

     <!--[ACC07]-->
...<VocabularyElement id="urn:epcglobal:fmcg:bizstep:physinv"/>
...<VocabularyElement id="urn:epcglobal:fmcg:bizstep:pickandpack"/>
   <VocabularyElement id="urn:epcglobal:fmcg:bizstep:loading"/>
   <VocabularyElement id="urn:epcglobal:fmcg:bizstep:production"/>

     <!--[EPC07b] -->
   <VocabularyElement id="urn:epcglobal:fmcg:bizstep:accepting"/>
   <VocabularyElement id="urn:epcglobal:fmcg:bizstep:holding"/>
   <VocabularyElement id="urn:epcglobal:fmcg:bizstep:storing"/>
```

```
    <VocabularyElement id="urn:epcglobal:fmcg:bizstep:stocking"/>
    <VocabularyElement id="urn:epcglobal:fmcg:bizstep:repacking"/>
    <VocabularyElement id="urn:epcglobal:fmcg:bizstep:packing"/>
    <VocabularyElement id="urn:epcglobal:fmcg:bizstep:staging_outbound"/>
    <VocabularyElement id="urn:epcglobal:fmcg:bizstep:reserving"/>
    <VocabularyElement id="urn:epcglobal:fmcg:bizstep:encoding"/>
    <VocabularyElement id="urn:epcglobal:fmcg:bizstep:commissioning"/>
    <VocabularyElement id="urn:epcglobal:fmcg:bizstep:decommissioning"/>
    <VocabularyElement id="urn:epcglobal:fmcg:bizstep:destroying"/>
    <VocabularyElement id="urn:epcglobal:fmcg:bizstep:dispensing"/>
    <VocabularyElement id="urn:epcglobal:fmcg:bizstep:other"/>
   </VocabularyElementList>
  </Vocabulary>
</VocabularyList>
```

8.2.2. BusinessTransactionType

```
<VocabularyList>
 <Vocabulary type="urn:epcglobal:epcis:vtype:BusinessTransactionType">
  <VocabularyElementList>

     <!--http://rolls.mit.edu/EPCIS_REST/-->
   <VocabularyElement id="urn:epcglobal:fmcg:btt:po"/>
   <VocabularyElement id="urn:epcglobal:fmcg:btt:bol"/>

     <!--[ACC07]-->
   <VocabularyElement id="urn:epcglobal:fmcg:btt:asn"/>
  </VocabularyElementList>
 </Vocabulary>
</VocabularyList>
```

8.2.3. Disposition

```
<VocabularyList>
 <Vocabulary type="urn:epcglobal:epcis:vtype:Disposition">
  <VocabularyElementList>

     <!--http://rolls.mit.edu/EPCIS_REST/-->
   <VocabularyElement id="urn:epcglobal:hls:disp:active"/>
   <VocabularyElement id="urn:epcglobal:fmcg:disp:sellable_available"/>

     <!--[ACC07]-->
   <VocabularyElement id="urn:epcglobal:fmcg:disp:readyforuse"/>
   <VocabularyElement id="urn:epcglobal:fmcg:disp:readyforpickup"/>
   <VocabularyElement id="urn:epcglobal:fmcg:disp:inrepair"/>
   <VocabularyElement id="urn:epcglobal:fmcg:disp:transit"/>

     <!--[EPC07b] -->
   <VocabularyElement id="urn:epcglobal:hls:disp:inactiv"/>
   <VocabularyElement id="urn:epcglobal:fmcg:disp:reserved"/>
   <VocabularyElement id="urn:epcglobal:fmcg:disp:encoded"/>
   <VocabularyElement id="urn:epcglobal:fmcg:disp:in_transit"/>
   <VocabularyElement
   id="urn:epcglobal:fmcg:disp:sellable_not_accessible"/>
   <VocabularyElement id="urn:epcglobal:fmcg:disp:sellable_accessible"/>
   <VocabularyElement id="urn:epcglobal:fmcg:disp:non_sellable"/>
   <VocabularyElement id="urn:epcglobal:fmcg:disp:non_sellable_expired"/>
   <VocabularyElement
   id="urn:epcglobal:fmcg:disp:non_sellable_recalled"/>
   <VocabularyElement id="urn:epcglobal:fmcg:disp:non_sellable_damaged"/>
   <VocabularyElement
   id="urn:epcglobal:fmcg:disp:not_sellable_no_pedigree_match"/
   <VocabularyElement id="urn:epcglobal:fmcg:disp:returned"/>
   <VocabularyElement id="urn:epcglobal:fmcg:disp:in_progress"/>
   <VocabularyElement id="urn:epcglobal:fmcg:disp:sold"/>
```

```
    <VocabularyElement id="urn:epcglobal:fmcg:disp:destroyed"/>
    <VocabularyElement id="urn:epcglobal:fmcg:disp:unknown"/>
   </VocabularyElementList>
  </Vocabulary>
</VocabularyList>
```

8.2.4. EPCClass

```
<VocabularyList>
 <Vocabulary type="urn:epcglobal:epcis:vtype:EPCClass">
  <VocabularyElementList>

     <!--http://rolls.mit.edu/EPCIS_REST/-->
   <VocabularyElement id="urn:epc:idpat:sgtin:0614141.107340.*"/>
   <VocabularyElement id="urn:epc:idpat:sgtin:0614141.107341.*"/>
   <VocabularyElement id="urn:epc:idpat:sgtin:0614141.107342.*"/>
   <VocabularyElement id="urn:epc:idpat:sgtin:0614141.107343.*"/>
   <VocabularyElement id="urn:epc:idpat:sgtin:0614141.107344.*"/>
   <VocabularyElement id="urn:epc:idpat:sgtin:0614142.107345.*"/>
   <VocabularyElement id="urn:epc:idpat:sgtin:0614142.107346.*"/>
   <VocabularyElement id="urn:epc:idpat:sgtin:0614142.107347.*"/>
   <VocabularyElement id="urn:epc:idpat:sgtin:0614142.107348.*"/>
   <VocabularyElement id="urn:epc:idpat:sgtin:0614142.107349.*"/>

     <!--[ACC07]-->
   <VocabularyElement id="urn:epc:id:sgtin:0069000.919923.*"/>
   <VocabularyElement id="urn:epc:id:sgtin:0069000.957110.*"/>
  </VocabularyElementList>
 </Vocabulary>
</VocabularyList>
```

8.3. EPCISDocument

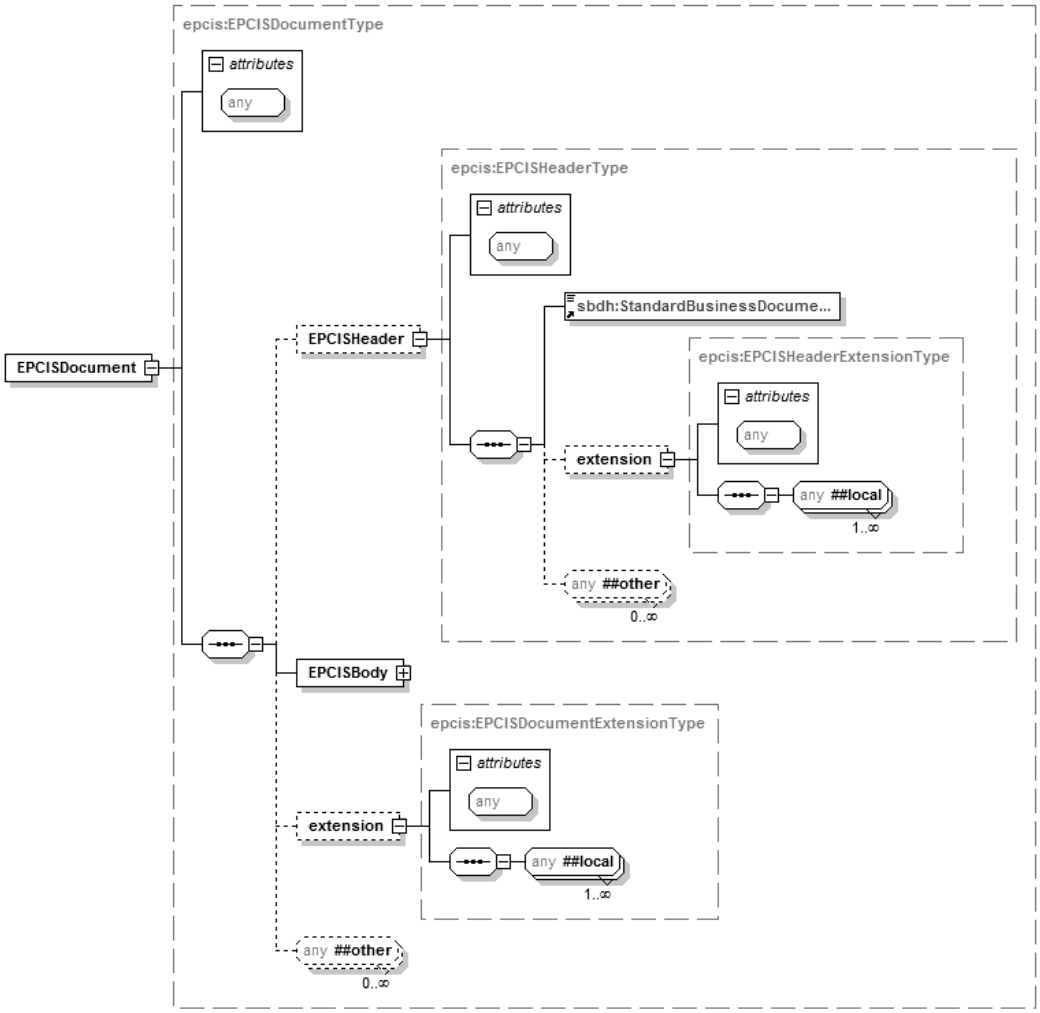

Abbildung 8.3.1 EPCISDocument nach [EPC07]

8.3.1. EPCISBodyType

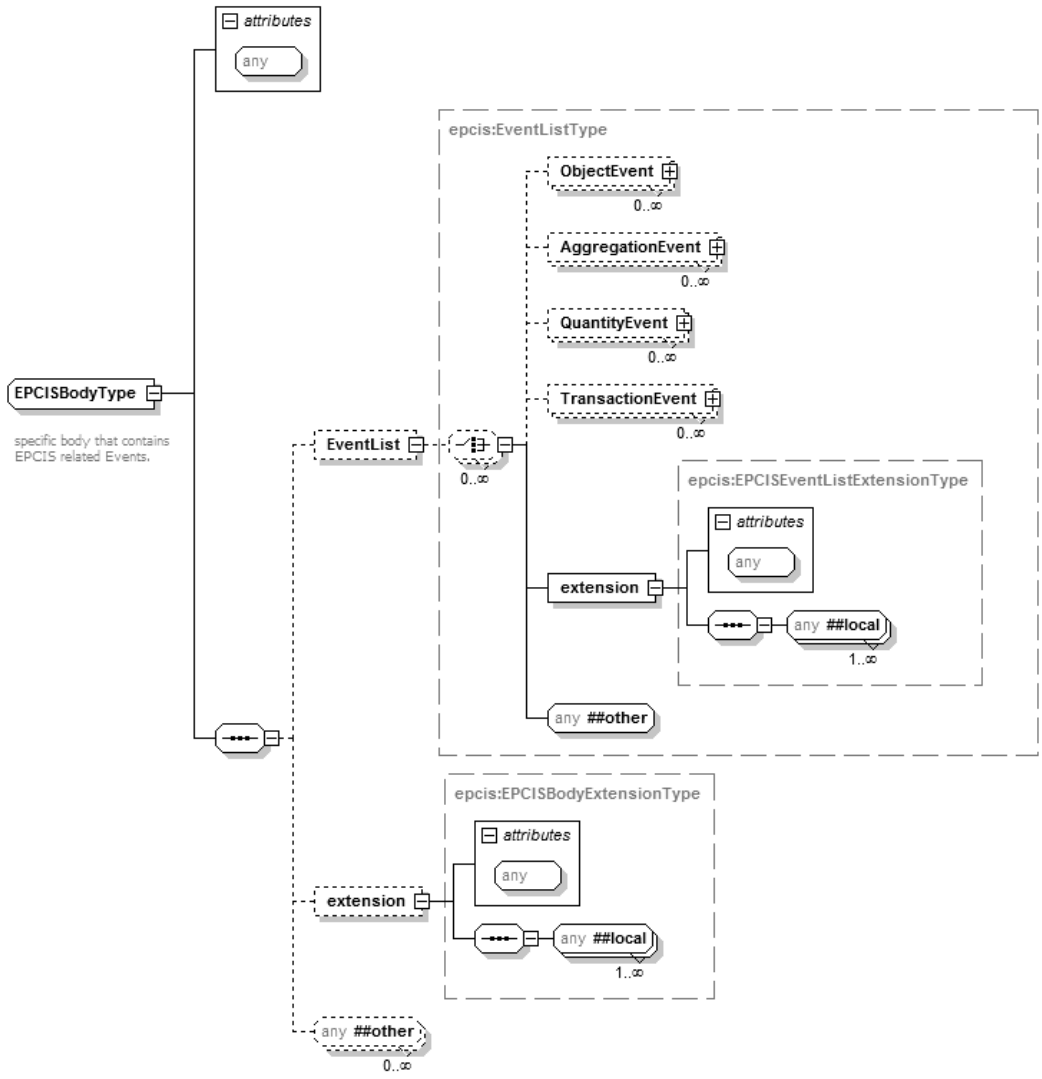

Abbildung 8.3.2 EPCISBodyType nach [EPC07]

8.3.2. ObjectEventType

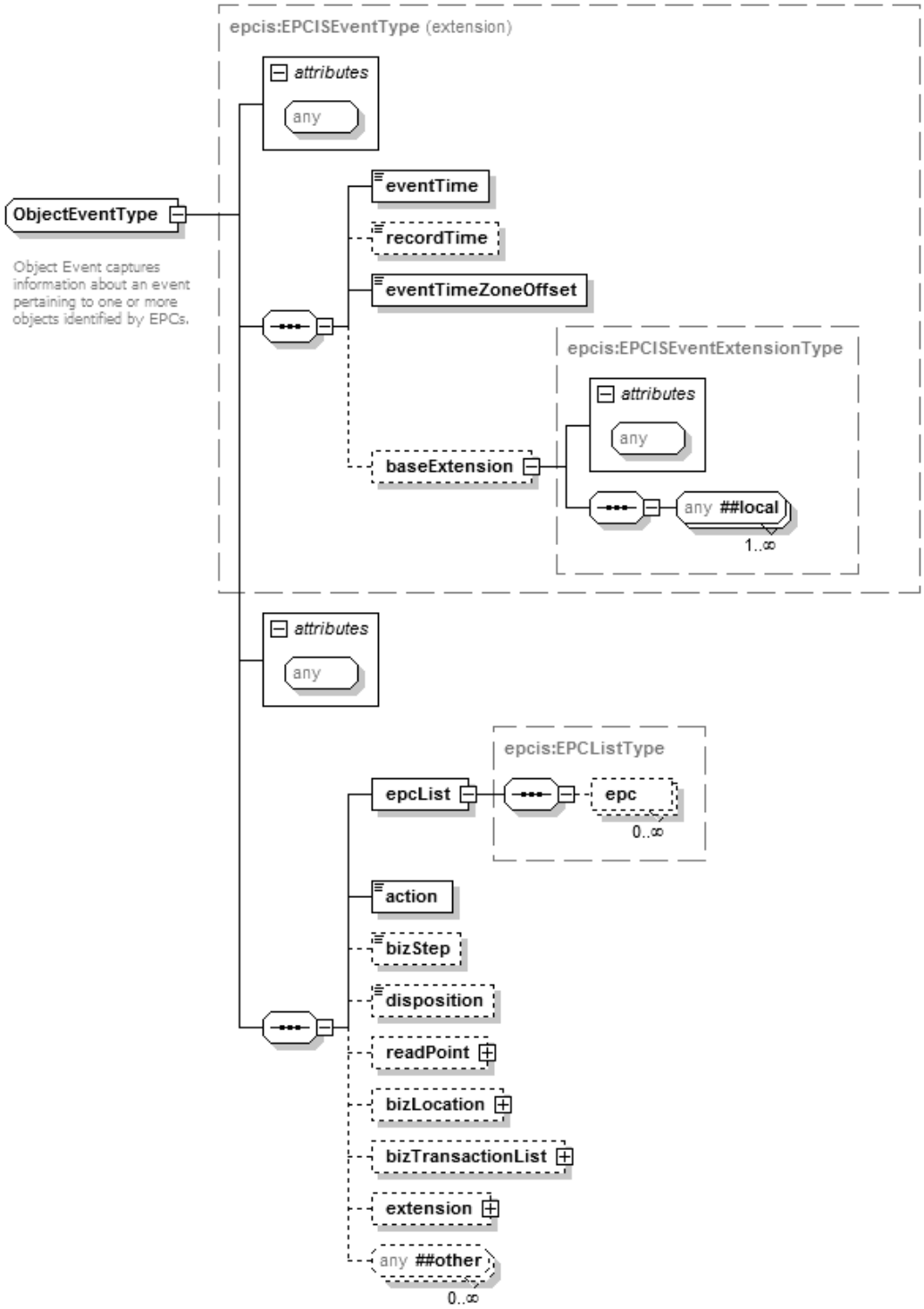

Abbildung 8.3.3 ObjectEventType nach [EPC07]

8.3.3. AggregationEventType

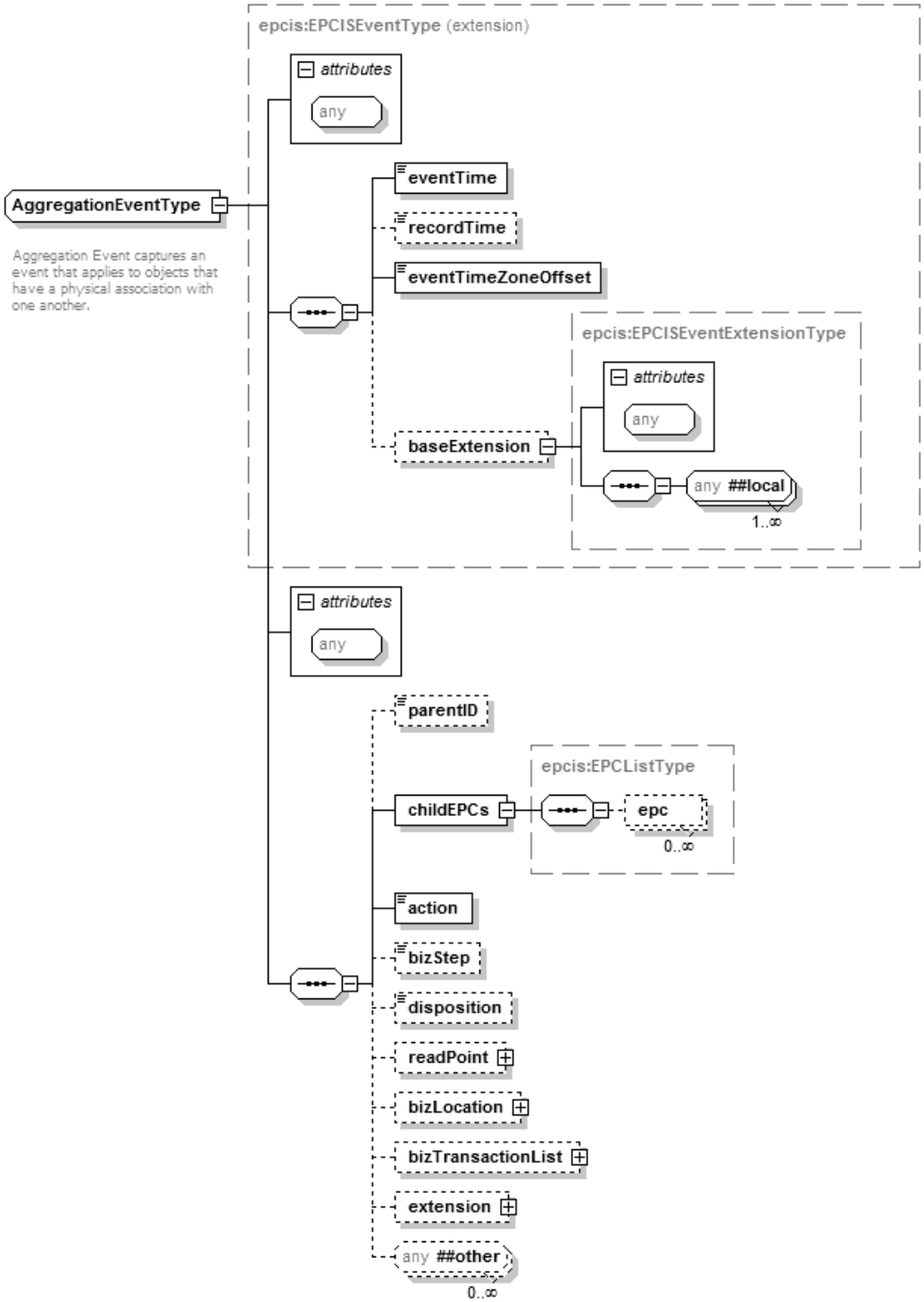

Abbildung 8.3.4 AggregationEventType nach [EPC07]

8.3.4. QuantityEventType

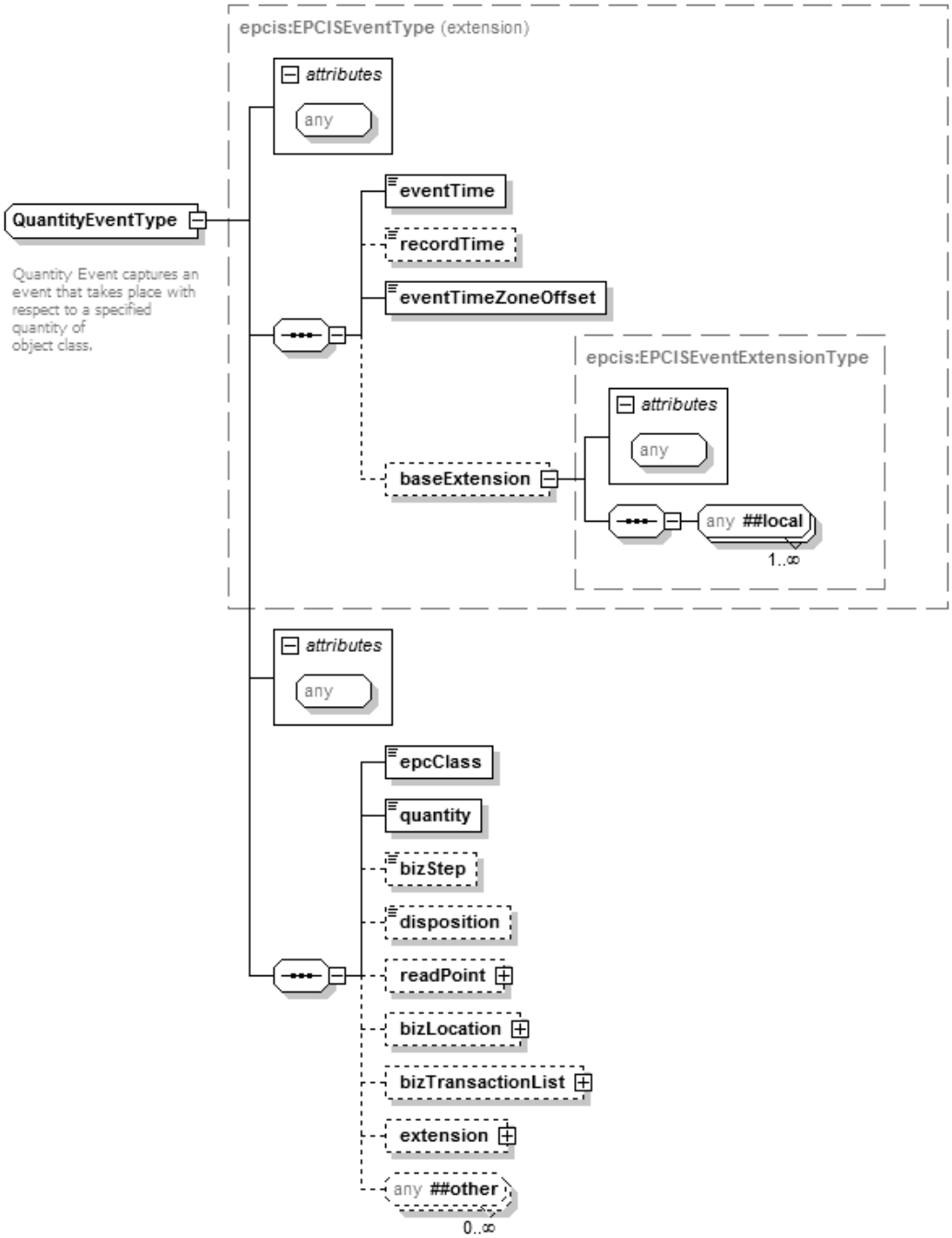

Abbildung 8.3.5 QuantityEventType nach [EPC07]

8.3.5. TransactionEventType

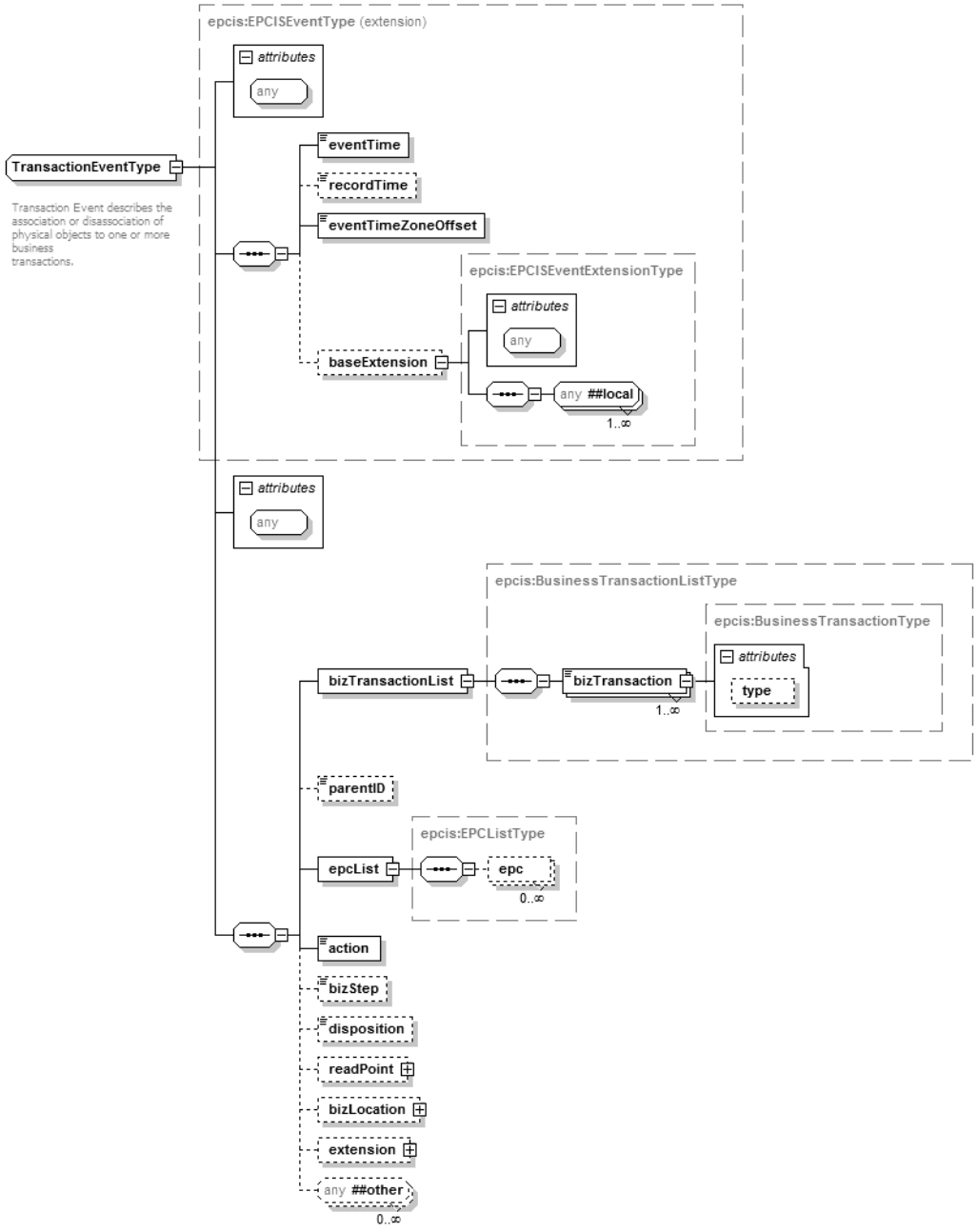

Abbildung 8.3.6 TransactionEventType nach [EPC07]

8.3.6. Beispiel

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<epcis:EPCISDocument
xmlns:epcis="urn:epcglobal:epcis:xsd:1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
creationDate="2005-07-11T11:30:47.0Z" schemaVersion="1">

 <EPCISBody>
  <EventList>

   <ObjectEvent>
    <eventTime>2005-04-03T20:33:31.116-06:00</eventTime>
    <eventTimeZoneOffset>-06:00</eventTimeZoneOffset>
     <epcList>
      <epc>urn:epc:id:sgtin:0614141.107346.2017</epc>
     </epcList>

     <action>OBSERVE</action>
     <bizStep>urn:epcglobal:epcis:bizstep:fmcg:shipped</bizStep>
     <disposition>
      urn:epcglobal:epcis:disp:fmcg:unknown</disposition>

     <readPoint>
      <id>urn:epc:id:sgln:0614141.07346.1234</id>
     </readPoint>
     <bizLocation>
      <id>urn:epcglobal:fmcg:loc:0614141073467.A23-49</id>
     </bizLocation>

     <bizTransactionList>
      <bizTransaction type="urn:epcglobal:fmcg:btt:po">
       http://transaction.acme.com/po/12345678</bizTransaction>
     </bizTransactionList>
    </ObjectEvent>

    <ObjectEvent>
     <eventTime>2005-04-04T20:33:31.116-06:00</eventTime>
     <eventTimeZoneOffset>-06:00</eventTimeZoneOffset>
     <epcList>
     <epc>urn:epc:id:sgtin:0614141.107346.2018</epc>
     </epcList>
...
    </ObjectEvent>

   </EventList>
  </EPCISBody>

</epcis:EPCISDocument>
```

Abbildung 8.3.7 Beispiel EPCISDokument [EPC07]

8.4. Beispiele des Xquery-basierter Ansatz

```
<PatientRecords>
      <Patient Name=”Aaron”>
            <Personal>
                  <SSN> 999-99-9999 </SSN>
                  <DoB>
                        <Month>January</Month>
                        <Date>01</Date>
                        <Year>1991</Year>
                  </DoB>
            </Personal>
            <Medical>
                  <Doctor> Brian </Doctor>
                  <Diagnosis> Cancer </Diagnosis>
                  <Prescription> Chemo medicine </Prescription>
                  <Bill> 500.00 </Bill>
            </Medical>
      </Patient>

      <Patient Name=”Christy”>
            <Personal>
                  <SSN> 444-44-4444 </SSN>
                  <DoB>
                        <Month>February</Month>
                        <Date>02</Date>
                        <Year>1972</Year>
                  </DoB>
            </Personal>
            <Medical>
                  <Doctor> David </Doctor>
                  <Diagnosis> Diabetes </Diagnosis>
                  <Prescription> Insulin </Prescription>
                  <Bill> 100.00 </Bill>
            </Medical>
      </Patient>

      <Patient Name=”Emily”>
            <Personal>
                  <SSN> 222-22-2222 </SSN>
                  <DoB>
                        <Month>March</Month>
                        <Date>03</Date>
                        <Year>1983</Year> </DoB>
            </Personal>
            <Medical>
                  <Doctor> Fred </Doctor>
                  <Diagnosis> SARS </Diagnosis>
                  <Prescription> Unknown </Prescription>
                  <Bill> 1000.00 </Bill>
            </Medical>
      </Patient>
</PatientRecords>
```

Abbildung 8.4.1 hospital.xml [CGR03]

```
<Staff>
      <Employee Name=”Brian”>
            <Personal> <SSN> 666-66-6666 </SSN> </Personal>
            <StaffInfo>
                  <Position> Doctor </Position>
                  <AccountableTo> David </AccountableTo>
            </StaffInfo>
      </Employee>

      <Employee Name=”David”>
            <Personal> <SSN> 555-55-5555 </SSN> </Personal>
            <StaffInfo>
                  <Position> Doctor </Position>
                  <AccountableTo />
            </StaffInfo>
      </Employee>

      <Employee Name=”Fred”>
            <Personal> <SSN> 777-77-7777 </SSN> </Personal>
            <StaffInfo>
                  <Position> Doctor </Position>
                  <AccountableTo> David </AccountableTo>
            </StaffInfo>
      </Employee>

      <Employee Name=”Greg”>
            <Personal> <SSN> 888-88-8888 </SSN> </Personal>
            <StaffInfo>
                  <Position> Nurse </Position>
                  <AccountableTo>
                        <Doctor>David</Doctor>
                        <Doctor>Brian</Doctor>
                        <Doctor>Fred</Doctor>
                  </AccountableTo>
            </StaffInfo>
      </Employee>

</Staff>
```

Abbildung 8.4.2 office.xml [CGR03]

```
<access-rights>
let $office := document ( ”office.xml” )
let $hospital := document ( ”hospital.xml” )

for $doc1 in $hospital//Patient return
      for $doc2 in $office//Employee
            where $doc2//Position = ”Doctor” and
                  $doc1//Doctor = $doc2/@Name return

<Rules>
      <Rule>
            <Subject>{$doc2/@Name}</Subject>
            <Object>
                  <File>hospital.xml</File>
                  <Path>//Patient[{ $doc1/@Name }]</Path>
            </Object>
            <Right>READ</Right>
      </Rule>

      <Rule>
            <Subject>{$doc2/@Name}</Subject>
            <Object>
                  <File>hospital.xml</File>
                  <Path>//Patient[{ $doc1/@Name } ]//DoB</Path>
            </Object>
            <Right>READ</Right>
      </Rule>

      <Rule>
            <Subject>{$doc2/@Name}</Subject>
            <Object>
                  <File>hospital.xml</File>
                  <Path>
                        //Patient[{ $doc1/@Name }]/Medical
                  </Path>
            </Object>
            <Right>OVERWRITE</Right>
      </Rule>

</Rules>
</access-rights>
```

Abbildung 8.4.3 Regel 1 [CGR03]

8.5. Beispiele für XSLT

```
importorg.apache.xalan.*;

public class Counter{
int i;
int j;

public Counter(){}

public void init
(org.apache.xalan.extensions.XSLProcessorContextcontext,
org.apache.xalan.templates.ElemExtensionCallextElem){}

public int count(){i=i+1;returni;}
public void nextcount(){i=i+1;}
public int getcount(){returni;}
public int index(){j=j+1;returnj;}
public void setindex(int z){j=z;}
public void setcount(int j){i=j;}
}
```

Abbildung 8.5.1 Counter.java [See08]

8.6. Wertschöfungskette

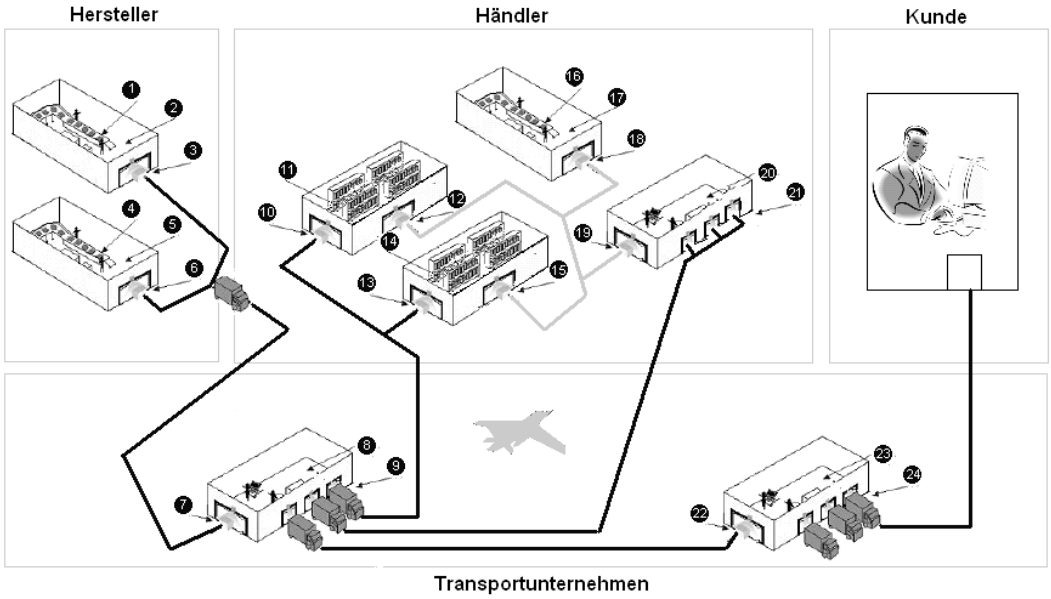

Abbildung 8.6.1 Wertschöpfungskette

Legende Abbildung 8.6.1:

1. Produktionslinie A mit integrierter RFID Auszeichnung
2. Verpackung
3. Warenausgang Produktionslinie A
4. Produktionslinie B mit integrierter RFID Auszeichnung
5. Verpackung
6. Warenausgang Produktionslinie B
7. Wareneingang
8. Zwischenlagerung und Versandvorbereitung
9. Laderampen mit RFID-Reader
10. Lagereingang
11. Lager für eigene Produktionslinie
12. Lagerausgang
13. Lagereingang
14. Lager für Kommissionsware
15. Lagerausgang
16. Produktionslinie C mit integrierter RFID Auszeichnung
17. kleines Zwischenlager
18. Warenausgang Produktionslinie C
19. Wareneingang Verpackungs- und Versandabteilung
20. Verpackungs- und Versandabteilung
21. Laderampen mit RFID-Reader
22. Wareneingang
23. Zwischenlagerung und Versandvorbereitung
24. Laderampen mit RFID-Reader

8.7. Anwendungsfälle

8.7.1. PolicySet 1

```
<?xml version="1.0" encoding="UTF-8"?>

<PolicySet
PolicySetId="spock"
PolicyCombiningAlgId="http://www.inf.tu-dresden.de/eal/0.2/PolicyCombining"
xmlns:ns2="http://www.inf.tu-dresden.de/eal/0.2/schema"
xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os">

 <Target>
  <Subjects>
   <Subject>
    <SubjectMatch
     MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      spock
     </AttributeValue>
     <SubjectAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:role"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </SubjectMatch>
   </Subject>
  </Subjects>

  <Resources/>
  <Actions/>
  <Environments/>
 </Target>

 <Policy
  PolicyId="spock/po/1234"
  RuleCombiningAlgId=
  "http://inf.tu-dresden.de/aidxacml/1.0/combiners/rule/eal-core-intersect">

  <Target>
   <Subjects>
    <Subject>
     <SubjectMatch
      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <AttributeValue
       DataType="http://www.w3.org/2001/XMLSchema#string">spock
      </AttributeValue>
      <SubjectAttributeDesignator
       AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:role"
       DataType="http://www.w3.org/2001/XMLSchema#string"/>
     </SubjectMatch>
    </Subject>
   </Subjects>

   <Resources>
    <Resource>
     <ResourceMatch MatchId=
      "http://inf.tu-dresden.de/aidxacml/1.0/matchers/eal-core-match">
      <AttributeValue
       DataType="http://www.inf.tu-dresden.de/eal/0.2/schema">
       <ns2:eventSet>
       <ns2:eventSubset visibleAttributes="*">
        <ns2:basicSet>ObjectEvent</ns2:basicSet>
        <ns2:basicSet>AggregationEvent</ns2:basicSet>
        <ns2:basicSet>TransactionEvent</ns2:basicSet>
        <ns2:conditionSet attribute="bizTransactionList.bizTransaction">
         <ns2:condition>
          <ns2:predicate operator="string-equal">
           http://transaction.acme.com/spock/po/1234
          </ns2:predicate>
         </ns2:condition>
        </ns2:conditionSet>
```

```
        </ns2:eventSubset>
       </ns2:eventSet>
      </AttributeValue>
      <ResourceAttributeDesignator
       AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
       DataType="http://www.inf.tu-dresden.de/eal/0.2/schema"/>
     </ResourceMatch>
    </Resource>
   </Resources>
  </Target>

   <Rule Effect="PartialPermit" RuleId="spock/po/1234">
   <Target>
    <Subjects>
     <SubjectMatch
      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
       <AttributeValue
        DataType="http://www.w3.org/2001/XMLSchema#string">spock
       </AttributeValue>
       <SubjectAttributeDesignator
        AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:role"
        DataType="http://www.w3.org/2001/XMLSchema#string"/>
     </SubjectMatch>
    </Subject>
   </Subjects>

    <Resources>
     <Resource>
      <ResourceMatch
      <ResourceMatch MatchId=
       "http://inf.tu-dresden.de/aidxacml/1.0/matchers/eal-core-match">
       <AttributeValue
        DataType="http://www.inf.tu-dresden.de/eal/0.2/schema">
        <ns2:eventSet>
         <ns2:eventSubset visibleAttributes="*">
          <ns2:basicSet>ObjectEvent</ns2:basicSet>
          <ns2:basicSet>AggregationEvent</ns2:basicSet>
          <ns2:basicSet>TransactionEvent</ns2:basicSet>
          <ns2:conditionSet attribute="bizTransactionList.bizTransaction">
           <ns2:condition>
            <ns2:predicate operator="string-equal">
             http://transaction.acme.com/spock/po/1234
            </ns2:predicate>
           </ns2:condition>
          </ns2:conditionSet>
         </ns2:eventSubset>
        </ns2:eventSet>
       </AttributeValue>
       <ResourceAttributeDesignator
        AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
        DataType="http://www.inf.tu-dresden.de/eal/0.2/schema"/>
      </ResourceMatch>
     </Resource>
    </Resources>
   </Target>

  </Rule>
 </Policy>
</PolicySet>
```

8.7.2. PolicySet 2

```
<?xml version="1.0" encoding="UTF-8"?>

<PolicySet
PolicySetId="spock"
PolicyCombiningAlgId="http://inf.tu-
dresden.de/aidxacml/1.0/combiners/policy/eal-core-intersect"
xmlns:ns2="http://www.inf.tu-dresden.de/eal/0.2/schema"
xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os">

 <Target>
  <Subjects>
   <Subject>
    <SubjectMatch
      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      spock
     </AttributeValue>
     <SubjectAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:role"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </SubjectMatch>
   </Subject>
  </Subjects>

  <Resources/>
  <Actions/>
  <Environments/>
 </Target>

 <Policy
  PolicyId="spock/po/1234"
  RuleCombiningAlgId=
  "http://inf.tu-dresden.de/aidxacml/1.0/combiners/rule/eal-core-intersect">

  <Target>
   <Subjects>
    <Subject>
     <SubjectMatch
       MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <AttributeValue
       DataType="http://www.w3.org/2001/XMLSchema#string">spock
      </AttributeValue>
      <SubjectAttributeDesignator
       AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:role"
       DataType="http://www.w3.org/2001/XMLSchema#string"/>
     </SubjectMatch>
    </Subject>
   </Subjects>

   <Resources>
    <Resource>
     <ResourceMatch MatchId=
      "http://inf.tu-dresden.de/aidxacml/1.0/matchers/eal-core-match">
      <AttributeValue
       DataType="http://www.inf.tu-dresden.de/eal/0.2/schema">
       <ns2:eventSet>
        <ns2:eventSubset visibleAttributes="*">
         <ns2:basicSet>ObjectEvent</ns2:basicSet>
         <ns2:basicSet>AggregationEvent</ns2:basicSet>
         <ns2:basicSet>TransactionEvent</ns2:basicSet>
         <ns2:conditionSet attribute="bizTransactionList.bizTransaction">
          <ns2:condition>
           <ns2:predicate operator="string-equal">
            http://transaction.acme.com/spock/po/1234
           </ns2:predicate>
          </ns2:condition>
         </ns2:conditionSet>
         <ns2:conditionSet attribute="parentId">
          <ns2:condition>
           <ns2:predicate operator="epc-match">100</ns2:predicate>
```

```
          </ns2:condition>
         </ns2:conditionSet>
         <ns2:conditionSet attribute="epcList.epc">
          <ns2:condition>
           <ns2:predicate operator="epc-match">1</ns2:predicate>
          </ns2:condition>
          <ns2:condition>
           <ns2:predicate operator="epc-match">2</ns2:predicate>
          </ns2:condition>
          <ns2:condition>
           <ns2:predicate operator="epc-match">3</ns2:predicate>
          </ns2:condition>
          <ns2:condition>
           <ns2:predicate operator="epc-match">4</ns2:predicate>
          </ns2:condition>
         </ns2:conditionSet>
        </ns2:eventSubset>
       </ns2:eventSet>
      </AttributeValue>
     <ResourceAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
      DataType="http://www.inf.tu-dresden.de/eal/0.2/schema"/>
     </ResourceMatch>
    </Resource>
   </Resources>
  </Target>

   <Rule Effect="PartialPermit" RuleId="spock/po/1234">
   <Target>
    <Subjects>
     <Subject>
      <SubjectMatch
       MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
       <AttributeValue
        DataType="http://www.w3.org/2001/XMLSchema#string">spock
       </AttributeValue>
       <SubjectAttributeDesignator
        AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:role"
        DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </SubjectMatch>
     </Subject>
    </Subjects>

    <Resources>
     <Resource>
      <ResourceMatch MatchId=
       "http://inf.tu-dresden.de/aidxacml/1.0/matchers/eal-core-match">
       <AttributeValue
        DataType="http://www.inf.tu-dresden.de/eal/0.2/schema">
        <ns2:eventSet>
         <ns2:eventSubset visibleAttributes="*">
          <ns2:basicSet>ObjectEvent</ns2:basicSet>
          <ns2:basicSet>AggregationEvent</ns2:basicSet>
          <ns2:basicSet>TransactionEvent</ns2:basicSet>
          <ns2:conditionSet attribute="bizTransactionList.bizTransaction">
           <ns2:condition>
            <ns2:predicate operator="string-equal">
             http://transaction.acme.com/spock/po/1234
            </ns2:predicate>
           </ns2:condition>
          </ns2:conditionSet>
          <ns2:conditionSet attribute="parentId">
           <ns2:condition>
            <ns2:predicate operator="epc-match">100</ns2:predicate>
           </ns2:condition>
          </ns2:conditionSet>
          <ns2:conditionSet attribute="epcList.epc">
           <ns2:condition>
            <ns2:predicate operator="epc-match">1</ns2:predicate>
           </ns2:condition>
           <ns2:condition>
            <ns2:predicate operator="epc-match">2</ns2:predicate>
```

```
            </ns2:condition>
            <ns2:condition>
             <ns2:predicate operator="epc-match">3</ns2:predicate>
            </ns2:condition>
            <ns2:condition>
             <ns2:predicate operator="epc-match">4</ns2:predicate>
            </ns2:condition>
           </ns2:conditionSet>
          </ns2:eventSubset>
         </ns2:eventSet>
        </AttributeValue>
        <ResourceAttributeDesignator
         AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
         DataType="http://www.inf.tu-dresden.de/eal/0.2/schema"/>
       </ResourceMatch>
      </Resource>
     </Resources>
    </Target>

  </Rule>
 </Policy>
</PolicySet>
```

8.8. Struktur des XSLT-Stylesheets für PIE-PolicySets

8.8.1. Vordefinierte Struktur für XSLT-Stylesheet

```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
xmlns:eal="http://inf.tu-dresden.de/eal/1.0/schema"
xmlns:xalan="http://xml.apache.org/xalan"
xmlns:java="http://xml.apache.org/xslt/java" exclude-result-prefixes="xalan
java"
xmlns:my-counter="de.tu.dresden.inf.senera.pie.epcfinder.Counter"
xmlns:my-epcfinder="de.tu.dresden.inf.senera.pie.epcfinder.Epcfinder"
extension-element-prefixes="my-counter my-epcfinder">

<xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>


<!--XSLT Javaerweiterung XSLT für den EPC-Finder-->
<xalan:component
 prefix="my-epcfinder" elements="init" functions="findallEpc">

  <xalan:script lang="javaclass"
   src="xalan://de.tu.dresden.inf.senera.pie.epcfinder.EPCFinder">
  </xalan:script>

</xalan:component>

<my-epcfinder:init/>

<!-- Select useraccount, biztransactiontype,  biztransactionid-->
<xsl:param name="useraccount" select="substring-before(substring-after
(string(//bizTransaction), 'com/'), '/')"/>
<xsl:param name="biztransactiontype" select="substring-before(substring-
after(substring-after(string(//bizTransaction), 'com/'), '/'), '/')"/>
<xsl:param name="biztransactionid" select="substring-after(substring-after
(substring-after(string(//bizTransaction), 'com/'), '/'), '/')"/>


<!-- Set policiesetid, policyid,  ruleid-->
<xsl:param name="policiesetid" select="$useraccount"/>
<xsl:param name="policyid" select="$useraccount"/>
<xsl:param name="ruleid" select="concat(concat($biztransactiontype,'_'),
$biztransactionid)"/>
```

```
<!-- CombiningAlgorithmen -->
<xsl:variable name="policycombiningalgId">
 http://www.inf.tu-dresden.de/eal/0.2/PolicyCombining
</xsl:variable>
<xsl:variable name="rulecombiningalgId">
 http://inf.tu-dresden.de/aidxacml/1.0/combiners/rule/eal-core-intersect
</xsl:variable>


<!-- Generation PIEPolicySet -->
<xsl:template match="/">

<!-- PolicySet -->
 <PolicySet PolicySetId="{$policiesetid}" PolicyCombiningAlgId="{
$policycombiningalgId}">

<!-- select description for integration TransactionEvent -->
 <xsl:for-each select="TransactionEvent">
  <Description>TransactionEvent</Description>
 </xsl:for-each>

<!-- or select select description for integration AggregationEvent -->
 <xsl:for-each select="AggregationEvent">
  <Description>AggregationEvent</Description>
 </xsl:for-each>

  <xsl:call-template name="TargetPolicySet"/>

<!-- Policy -->
  <Policy PolicyId="{$policyid}" RuleCombiningAlgId="{$rulecombiningalgId}">
   <Target/>

<!-- Rule -->
   <Rule Effect="PartialPermit" RuleId="{$ruleid}">
    <xsl:call-template name="TargetRule"/>
   </Rule>
  </Policy>
 </PolicySet>
</xsl:template>


<!-- Generation Target for PolicySet -->
<xsl:template name="TargetPolicySet">
 <Target>
  <Subjects>
   <Subject>
    <SubjectMatch
     MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      <xsl:value-of select="$useraccount"/>
     </AttributeValue>
     <SubjectAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:role"
      DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </SubjectMatch>
   </Subject>
  </Subjects>
 </Target>
</xsl:template>


<!-- Generation Target for Policy -->
<xsl:template name="TargetRule">
 <Target>
  <Resources>
   <Resource>
    <ResourceMatch MatchId=
     "http://inf.tu-dresden.de/aidxacml/1.0/matchers/eal-core-match">
     <AttributeValue DataType="http://www.inf.tu-dresden.de/eal/0.2/schema">

<!-- Select Generation EAL for TransactionEvent -->
```

```
      <xsl:for-each select="TransactionEvent">
       <xsl:call-template name="EALTransactionEvent"/>
      </xsl:for-each>

<!-- or Select Generation EAL for AggregationEvent -->
      <xsl:for-each select="AggregationEvent">
       <xsl:call-template name="EALAggregationEvent"/>
      </xsl:for-each>

     </AttributeValue>
     <ResourceAttributeDesignator
      AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
      DataType="http://www.inf.tu-dresden.de/eal/0.2/schema"/>
    </ResourceMatch>
   </Resource>
  </Resources>
 </Target>
</xsl:template>


<!-- Generation EAL for TransactionEvent -->
<xsl:template name="EALTransactionEvent">
<!--   ...
       Generation EAL for TransactionEvent
       ...-->
</xsl:template>


<!-- Generation EAL for AggregationEvent -->
<xsl:template name="EALAggregationEvent">
<!--   ...
       Generation EAL for AggregationEvent
       ...-->
</xsl:template>

</xsl:stylesheet>
```

8.8.2. Beispieledefinitonen für ConditionSet

8.8.2.1. EPC-ConditionSet mit EPC-Finder

Alle EPC-Objekte des zugrunde liegenden EPCIS-Events und deren aggregierten EPCs, die der EPC-Finder ermittelt, werden zur Formulierung der Berechtigung verwendet. Das erstellte ConditionSet erlaubt, den Zugriff auf die im predicate-Element des conditionSet-Elementes angegeben EPCs.

```
<xsl:variable name="permitdepth">
 Wert für Aggregationstiefe einfügen
</xsl:variable>

<xsl:variable name="hiddenepcclass">
 Stringliste für nicht zu berücksichtigende EPC-Objektklassen einfügen
</xsl:variable>

 <eal:conditionSet attribute="epcList.epc">
  <xsl:for-each select="//epc">
   <xsl:variable name="epc" select="string(.)"/>

   <eal:condition>
    <eal:predicate operator="epc-match">
     <xsl:value-of select="$epc"/>
    </eal:predicate>
   </eal:condition>

   <xsl:value-of select="my-epcfinder:findallEpc($epc)"
    disable-output-escaping="yes"/>
```

```
  </xsl:for-each>
 </eal:conditionSet>
```

8.8.2.2. Time-ConditionSet

Zur Formulierung der Berechtigung wird der Erstellungszeitpunkt, die *eventTime*, des EPCIS-Events verwendet. Das erstellte conditionSet erlaubt den Zugriff auf die EPCIS-Events, deren eventTime größer oder gleich der angegebenen eventTime ist.

```
<xsl:variable name="time" select="string(//eventTime)"/>

 <eal:conditionSet Attribute="eventTime">
  <eal:condition>
   <eal:predicate Operator="greaterThanOrEqual">
    <xsl:value-of select="$time"/>
   </eal:predicate>
  </eal:condition>
 </eal:conditionSet>
</eal:conditionSet>
```

8.8.2.3. BizLocation-ConditionSet

Das erstellte conditionSet spezifiziert den Zugriff auf die angegebenen BizLocations. Alle EPC-Objekte mit der angegebenen BizLocations sind für den Berechtigten sichtbar.

```
<eal:conditionSet Attribute="bizLocation">
 <eal:condition>
  <eal:predicate operator=”string-equal”>Wert einfügen</eal:predicate>
 </eal:condition>
</eal:conditionSet>
```

8.8.2.4. BizStep-ConditionSet

Das erstellte conditionSet spezifiziert den Zugriff auf die angegebenen BizSteps. Alle EPC-Objekte mit den angegebenen BizSteps sind für den Berechtigten sichtbar.

```
<eal:conditionSet Attribute="bizStep">
 <eal:condition>
  <eal:predicate operator=”string-equal”>Wert einfügen</eal:predicate>
 </eal:condition>
</eal:conditionSet>
```

8.9. PIE-Datenbankschema

```
BEGIN;

-- Drop Table;
DROP TABLE EPC;
DROP TABLE AggEPC;
DROP TABLE PIERule;
DROP TABLE GRforTransaction;
DROP TABLE Transaction;
DROP TABLE TransactionEPC;

-- EPC and aggregate EPCs
CREATE TABLE epc (
id bigint PRIMARY KEY auto_increment,
epc VARCHAR(500) UNIQUE);

CREATE TABLE aggepc (
epcid bigint NOT NULL references EPC(id),
aggepc VARCHAR(500) NOT NULL,
idx int NOT NULL,
INDEX (epcid));

-- all generated PIERules with Timestamp
CREATE TABLE PIERule (
id bigint PRIMARY KEY auto_increment,
action VARCHAR(6) NOT NULL CHECK (action IN ('ADD','DELETE')),
biztransaction VARCHAR(500) NOT NULL,
user VARCHAR(500) NOT NULL,
transaction VARCHAR(5) NOT NULL,
grtime TIMESTAMP NOT NULL);

-- generate PIERule for Transaction
CREATE TABLE GRforTransaction (
id bigint PRIMARY KEY auto_increment,
transaction VARCHAR(5) UNIQUE
--,generateR CHAR(1) NOT NULL CHECK (agright IN ('Y','N')));

-- EPCs for Transactions
CREATE TABLE BizTransaction (
id bigint PRIMARY KEY auto_increment,
biztransaction VARCHAR(500) UNIQUE);

CREATE TABLE TransactionEPC (
transid bigint NOT NULL references Transaction(id),
epc VARCHAR(500) NOT NULL,
idx int NOT NULL,
INDEX (transid));

COMMIT;
```

9. Literaturverzeichnis

[ACC07] Accada: www.accada.org/../epcis_demo_data.sql, 2007

[BEA06] BEA Systems, Inc.: BEAWebLogic RFID EnterpriseServer™: Understanding the Event, Master Data,and Data ExchangeServices v.2.0, Oktober 2006

[BGMP03] Manish Bhide, Ajay Gupta, Mukesh Mohanai, Sandeep Pandey: Dynamic Access Control Framework Based On Event: A Demonstration, IEEE Computer Society, 2003

[CGR03] Chris Clifton, Siddhartha K. Goel, Arnon Rosenthal: Derived Access Control Specification for XML, ACM Workshop on XML Security, Oktober 2003

[EAL08a] Eberhard Grummt: Die Event Addressing Language v1.0, SAP-internes Arbeitspapier, Oktober 2008

[EDA08] Wikipedia: Event-driven architecture, http://en.wikipedia.org/wiki/Event_Driven_Architecture, September 2008

[EDI08] GS1 Germany, EDI-Team: EDI-Standard – ein Leitbild, http://www.gs1-germany.de/internet/content/e39/e466/e468/datei/ean/ebusiness_edi/leitbild_edi_und_eancom.pdf, Oktober 2008

[EPC07] EPCglobal Inc: EPC Information Services (EPCIS) Version 1.0.1 Specification, http://www.epcglobalinc.org/standards/epcis/epcis_1_0_1-standard-20070921.pdf, Sebtember 2007

[EPC07b] EPCglobal Inc.: EPCIS-Introduction: Presented at Paris JAG Meeting in 2007, http://www.epcglobalinc.org/standards/epcis/epcis_1_0-presentation-20070619.pdf, 2007

[FOS08] Fosstrak: Fosstrak EPCIS Project, http://www.fosstrak.org/epcis/index.html, 2008

[GS07] Eberhard Grummt, Martin Schöffel: Verteilte Autorisationin RFID-Ereignissystemen, P. Horster (Hrsg.); D•A•CH Security, 2007

[GS106] GS1 Germany GmbH: EPC-Informationsservice(EPCIS) und Umsetzung im EPC-Showcase, http://www.gs1-germany.de//content/e39/e466/e468/datei/epc_rfid/epcis_epcshowcase.pdf, August 2006

[GS107] Sehorz DI Engen: EPCglobal ratifiziert globalenKommunikationsstandard für sicherenEcht-Zeit Datenaustausch, GS1 Austria Information (Hrsg.); Information Zeitung für Artikelnummerierung, Warenwirtschaft und elektronischen Datenaustausch, Ausgabe 2.2007, Juli 2007

[GS108] GS1 Austria GmbH: Rückverfolgbarkeit mit dem GS1 Rückverfolgbarkeits-Standard, http://www.gs1austria.at/html/documents/RV_Broschuere.pdf, 2008

[Hen08] Sebastian Hennebrüder: Java Datenbank Entwicklung / Einführung, http://www.laliluna.de/java-datenbank-entwicklung_de.html, Hibernate, EJB 2, JDBC Tutorials, 2008

[Jes00] Jesse Ralf: Java 2, 5. Ausgabe, S.16, moderne industrie Buch AG&Co.KG, ISBN: 3-8266-8100-2, 2000

[Kaa07] Dr. rer. nat. Jürgen Kaack: Wertschöpfungskette / Szenarien zeigen, wo es Alternativen gibt, http://www.mittelstandswiki.de/Wertschöpfungskette, Oktober 2007

[Kuh08] Frank Kuhlmann: EPCIS und EANCOM®:Standards gehen Hand in Hand, www.isis-specials.de/profile_pdf/1g204_ed1_rfid0208.pdf, ISIS Medien (Hrsg.), 2008

[Leh07] Kathrin Lehmann: Modelle und Techniken für eine effiziente und lückenlose Zugriffskontrolle in Java-basierten betrieblichen Anwendungen, Dissertation, 2007

[Lip07] IT Wissen: Das große Online-Lexikon der Informationstechnologie, http://itwissen.info/fileadmin/user_upload/EBOOKS/2007_02_Warenverkehr.pdf, Klaus Lipinski (Hrsg.); DATACOM-Buchverlag GmbH, 2007

[Man06] Sal Mangano: XSLT Kochbuch, 2.Ausgabe, O'Reilly, ISBN: 3-89721-457-1, 2006

[Mic06] Brenda M. Michelson: Event-Driven Architecture Overview: Event-Driven SOA Is Just Part of the EDA Story, http://dx.doi.org/10.1571/bda2-2-06cc.pdf, Februar 2006

[Mos05] Tim Moses, Entrust Inc.: eXtensible Access Control Markup Language (XACML) Version 2.0, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf, OASIS-Standard, Februar 2005

[OLW08] Oates Richard, Langer Thomas, Wille Stefan, Lueckow Torsten, Bachlmayr Gerald: Spring&Hibernate / Eine praxisbezogene Einführung, 2. Ausgabe, Carl Hanser, ISBN: 978-3-446-41213-2, 2008

[OT08] openTRANS: www.opentrans.de, 2008

[Pat03] Tony Patton: Grundlagen von XQuery: einfaches Abrufen von XML-Daten, http://www.zdnet.de/anwendungsentwicklung_grundlagen_von_xquery_einfaches_abrufen_von_xml_daten_story-20000201-39117872-1.htm, Dezember 2003

[Sch08] Martin Schöffel: Verteilte Zugriffsentscheidungen in TraceabilityNetworks, Diplomarbeit, Juni 2008

[See08] M. Seeboerger-Weichselbaum: Gut kombimiert, entwickler magazin; Ausgabe 2.2008, Februar 2008

[SKO08] Volker Schmitz, Oliver Kelkar, Boris Otto: openTRANS® Version 2.0 final draft, Fraunhofer IAO, Stuttgart; Universität Duisburg-Essen BLI, August 2008

[XSLT08] Wikipedia: http://de.wikipedia.org/wiki/XSL_Transformation, Dezember 2008

10. Abbildungsverzeichnis

11. Tabellenverzeichnis

12. CD-ROM zum PIE-Prototypen

Inhalt der CD-ROM:

- NetBeans-Projekt:
  - PIE
  - HibernateDB
- Tomcat lib-Dateien
- Tomcat webapps
- Diplomarbeit