universität wien

# MASTERARBEIT

Titel der Masterarbeit

„Integration of international, heterogeneous business communities into an e-Delivery network”

verfasst von

Philip Helger Bakk. rer. soc. oec.

angestrebter akademischer Grad

Diplom-Ingenieur (Dipl.-Ing.)

Wien, 2014

| | |
|---|---|
| Studienkennzahl lt. Studienblatt: | A 066 926 |
| Studienrichtung lt. Studienblatt: | Masterstudium Wirtschaftsinformatik |
| Betreut von: | Ao.Univ.Prof. Mag. Dr. Christian Huemer |

# Inhaltsverzeichnis

# 1 Einleitung

Die elektronische Rechnung (kurz e-Rechnung oder e-Invoice) ist ein Teil des elektronischen Beschaffungswesens (kurz e-Procurement), das sich mit der Beschaffung von Waren und Dienstleitungen über elektronische Kanäle beschäftigt. Damit sollen Kosteneinsparungen im Einkaufsprozess von Unternehmen realisiert werden. Laut der Billentis Marktstudie 2012 (anzufordern unter http://www.billentis.com/marktreport_e-rechnung_DE.htm; Stand 7.6.2014) gibt es bei flächendeckender Einführung der e-Rechnung alleine im österreichischen öffentlichen Sektor ein Einsparungsvolumen von rund 600 Millionen Euro. In größeren Ländern wie Deutschland oder Großbritannien ergibt sich sogar ein Einsparungspotential in Milliardenhöhe. In der Billentis Marktstudie 2009 wurde das durchschnittliche Einsparungspotential einer einzelnen e-Rechnung berechnet. Bei der Umstellung von Papierrechnung auf e-Rechnung kann ein Rechnungssteller rund 6,90€ (~62%) und ein Rechnungsempfänger sogar 11,60€ (~66%) pro Rechnung sparen. Die folgende Grafik aus der Billentis Marktstudie 2009 in schlüsselt die Einsparungen auf (Deutsche Bank Research 2010).

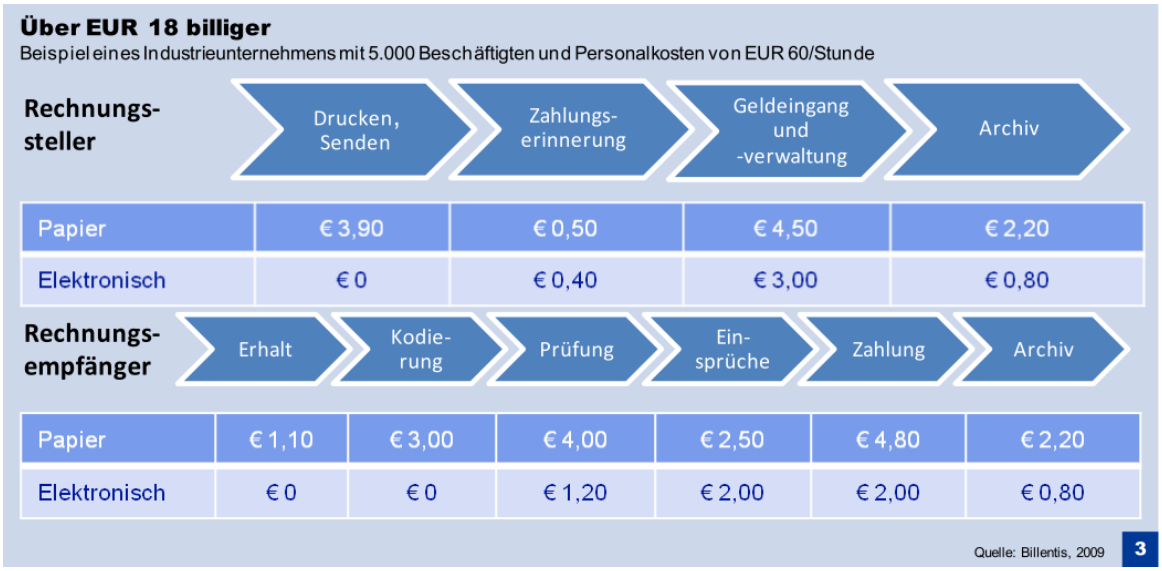


Abbildung 1: Kostenersparnis der e-Rechnung

Bei diesen großen Einsparungsmöglichkeiten darf aber nicht unerwähnt bleiben, dass durch die Einführung von elektronischem Datenaustausch auch Kosten entstehen. Diese betreffen nicht nur den Teil der EDV-Adaptionen, sondern auch Kosten für die Erstellung und Umsetzung von neuen Geschäftsprozessen sowie sonstige Anpassungen in der Organisation (z.B. Schulungskosten).

In Österreich ist die elektronische Rechnung spätestens seit dem 1. Jänner 2014 ein in der breiten Öffentlichkeit bekanntes Thema. Seit diesem Zeitpunkt nimmt der österreichische Bund nur noch strukturierte elektronische Rechnungen entgegen. Der Bund ist damit ein Vorreiter in Sachen e-Rechnung und gibt der Wirtschaft damit einen wichtigen Impuls. Der Bund listet dabei neben den Kosteneinsparungen noch andere Vorteile der e-Rechnung auf (Bundesministerium für Finanzen):

- Kürzere Bearbeitungszeit

    - Konzentration auf das Kerngeschäft durch elektronische Anlieferung der Rechnung
- Reduktion der Manipulationskosten
    - Kuvertierung, Frankierung, Aufgabe des Briefes bei der Post
- Reduktion von Versandkosten
    - Papier, Kuverts, Portogebühren
- Reduktion der Durchlaufzeit
    - schlankere Prozesse durch Wegfall der Transportwege und -zeiten und IT-unterstützte Rechnungsbearbeitung
- Reduktion von Medienbrüchen
    - Die eingebrachte elektronische Rechnung wird automatisch weiterbearbeitet
    - Mögliche Schreibfehler fallen weg
- Positiver Effekt auf die Umwelt
    - weniger Papierverbrauch, geringerer Transportaufwand

In vielen Ländern gibt es schon lange die Branche der „Service Provider“ bzw. "EDI-Provider", die aus den ehemaligen VAN-Betreibern (Value Added Network) entstanden ist. Service Provider sind im Zusammenhang mit elektronischer Beschaffung die Hubs, die Dokumente auf verschiedenen Kanälen und in verschiedenen Formaten empfangen, an die Bedürfnisse des Empfängers anpassen (zum Beispiel durch eine Transformation des Dokuments in ein anderes Format oder eine andere Syntax) und anschließend auf den vom Empfänger gewünschten Kanal an diesen liefern. Meistens verlangen Service Provider mehrere Cent pro übertragenes Dokument und sind damit noch immer viel günstiger als der vergleichbare Papier-Prozess.

Ein großes Problem der Service Provider ist die einfache automatische Vernetzung untereinander. Ist ein Unternehmen Kunde eines Service Providers, kann es im Normalfall auch nur Dokumente von diesem Service Provider empfangen. Um auch Dokumente von einem anderen Service Provider zu bekommen (weil z.B. ein Lieferant nur über diesen liefert), muss zumeist ein separater Vertrag abgeschlossen werden und der Aufwand steigt für alle beteiligten Parteien. Das europäische Projekt PEPPOL (Pan European Public Procurement On-Line; siehe Kapitel 4) wurde mit dem Ziel gestartet, eine technische und organisatorische Basis für den automatisierten Datenaustausch zwischen Service Providern im Bereich der elektronischen Beschaffung zu definieren und umzusetzen. Durch eine zentrale Registry, ein definiertes Austauschformat und standardisierte Prozesse können Service Provider problemlos Daten an Empfänger anderer Netzwerke versenden, ohne genau zu wissen, wer der Empfänger ist.

Nahezu zeitgleich mit PEPPOL wurde auch ERPEL (E-Business Registry Permitting Enterprise Liaisons; siehe Kapitel 3) entwickelt. Dabei handelt es sich um ein Forschungsprojekt der Technischen Universität Wien (TU). Die Zielsetzung von ERPEL ist die Lösung des Problems der Heterogenität beim Austausch von elektronischen Geschäftsdokumenten zwischen Geschäftspartnern. In ERPEL standen die praktische Umsetzung und die Unterstützung von KMUs durch die Integration in COTS-Software (Commercial off-the-shelf) im Vordergrund, während PEPPOL eine einheitliche europäische Lösung mit einer fundierten theoretischen Grundlage angestrebt hat.

Durch die Integration der PEPPOL-Infrastruktur mit dem ERPEL-Projekt, die in dieser Arbeit beschrieben und umgesetzt ist, soll die Reichweite beider Netzwerke erweitert werden. Dadurch gibt es auf der einen Seite eine größere Anzahl von potentiellen Geschäftspartnern auf Seiten von ERPEL und auf der anderen Seite können dadurch implizite Kosteneinsparungen realisiert werden, da andere spezifische Schnittstellen nicht implementiert werden müssen. Die umgesetzte Schnittstelle zwischen diesen Projekten hört auf den Namen ERPOL, und ist in Kapitel 5 ausführlich beschrieben.

## 1.1 Background und Motivation

Bereits 2006 wurden im Bundesrechenzentrum (BRZ) in Kooperation mit dem Bundesministerium für Finanzen (BMF) die ersten Studien und Analysen zum Thema E-Rechnung gestartet. Im Rahmen meiner beruflichen Tätigkeit war ich von Anfang an in dieses Projekt integriert. Ab 2007 wurde mit der Umsetzung einer nationalen Lösung begonnen, diese ging jedoch erst viel später live. Im Jahr 2009 wurde das Projekt PEPPOL gestartet, in dem ich als technische Unterstützung in den Arbeitspaketen WP5 (eInvoicing) und WP8 (Transport infrastructure) mitgewirkt habe. Österreich war in PEPPOL durch das Konsortium PEPPOL.AT vertreten, dass sich aus BMF, BBG (Bundesbeschaffungsgesellschaft) und BRZ zusammengesetzt hat. Parallel dazu wurde die Arbeit an der „E-Rechnung an den Bund“-Applikation (ER>B), der österreichischen nationalen Lösung, fortgesetzt und ging 2010 live. Allerdings war damals das Interesse der Wirtschaft am Senden von elektronischen Rechnungen nicht vorhanden. Damals wurde im Vergleich zu heute aber noch Finanz-Online als Identity Provider (das System zur Identifikation der Benutzer) verwendet. Heute kann nur noch nach einer Anmeldung am Unternehmensserviceportal (USP; https://www.usp.gv.at/Portal.Node/usp/public - Stand 7.6.2014) eine e-Rechnung an den Bund versendet werden.

Durch die intensive Mitarbeit des BRZ an allen Teilen der PEPPOL Komponenten gelang es uns schlussendlich auch diese für den österreichischen Bund umzusetzen, und diese sind noch heute im Einsatz. Ende 2012 wurde das IKT Konsolidierungsgesetz vom Parlament verabschiedet. Damit wurde der Bund verpflichtet, ab 1.1.2013 elektronische Rechnungen empfangen zu können, und ab 1.1.2014 müssen sämtliche Rechnungen des Bundes elektronisch einlangen. Aber auch dieses Gesetz wurde von der Wirtschaft lange ignoriert bzw. nicht ernst genommen. Erst in der zweiten Jahreshälfte 2013 kam es zu einem regelrechten Ansturm der Lieferanten des Bundes, da sie bis Ende 2013 mit der Umsetzung fertig sein mussten, um 2014 die Bezahlung ihrer Rechnungen sicherzustellen. Da der Bund Anfang 2014 in einer inoffiziellen Übergangsfrist auch noch vereinzelte Papierrechnungen entgegen genommen hat, kam es jedoch zu keinen gröberen organisatorischen Problemen. Aus heutiger Sicht kann die Einführung der E-Rechnung an den Bund als Erfolg gewertet werden. Per 3. Juni 2014 kam die 300.000ste e-Rechnung beim Bund an, und in Summe übermittelten bereits rund 14.800 Lieferanten des Bundes eine e-Rechnung (Quelle: ER>B-Newsletter 6.6.2014).

Meine Rolle bei ER>B ist die des technischen Umsetzungsverantwortlichen. D.h. ich konzeptioniere die ER>B-Software gemeinsam mit dem BMF und setze diese mit meinem BRZ-Kollegen um. Gleichzeitig bin ich mit meinen BRZ-Kollegen auch für die Supportanfragen der Lieferanten zuständig. Außerdem halte ich auch Vorträge zum Thema ER>B und bin Vertreter des BRZ im Arbeitskreis e-Billing der AustriaPro (die B2B-Standardisierungsplattform der Wirtschaftskammer Österreich). Die nächsten großen Herausforderungen befinden sich im

Umkreis von PEPPOL. Nach Ende des Projekts wurde der gemeinnützige OpenPEPPOL-Verein gegründet, der sich um die Fortführung der PEPPOL-Agenden kümmert. Im Zuge dessen wurden die Spezifikation sowohl auf inhaltlicher als auch auf technischer Ebene überarbeitet und verbessert und diese Änderungen müssen nun noch im Laufe des Jahres 2014 in die existierende PEPPOL-Infrastruktur des Bundes einfließen.

Da PEPPOL in Österreich noch eher stiefmütterlich behandelt wird, ist es mir mit dieser Arbeit ein besonderes Anliegen einerseits über PEPPOL zu informieren, und andererseits durch die praktische Umsetzung von ERPOL auch zu zeigen, dass der Aufwand der Anbindung an die PEPPOL-Infrastruktur in einem vernünftigen Rahmen umsetzbar ist. Vielleicht dient diese Arbeit anderen als Grundlage für eine eigene PEPPOL-Anbindung.

Das auf der TU von der Gruppe um Professor Christian Huemer gestartete ERPEL-Projekt entwickelte sich relativ zeitgleich zum PEPPOL-Projekt. Prof. Huemer ist Mitglied in internationalen Standardisierungsgremien (UN/CEFACT) und ebenfalls Mitglied im AustriaPro Arbeitskreis e-Billing, wodurch der Kontakt zu ERPEL zustande kam. ERPEL hat mich von Anfang an sehr interessiert, weil es sich im Vergleich zu PEPPOL um keine theoretische, sondern um eine praktische Lösung handelt. Durch die Zusammenarbeit mit österreichischen ERP-Software-Herstellern wurde in ERPEL von Anfang an auf die Umsetzbarkeit für COTS-Software (Commercial off-the-shelf – Software von der Stange) geachtet. Speziell in KMUs ist COTS-Software weit verbreitet und durch die Verfügbarkeit des ERPEL-Netzwerks erschließt sich den Benutzern ein großer Kreis von Rechnungsempfängern. In Kombination mit PEPPOL wird ERPEL ebenfalls mächtiger, da damit auch Rechnungsempfänger aus anderen europäischen Ländern automatisch verfügbar sind.

## 1.2 Methodologie

Der methodische Rahmen, der dieser Arbeit zu Grunde liegt ist das Design Science Paradigma nach Hevner. Demnach werden IT-Artefakte, die mit dem Ziel erstellt wurden ein konkretes Problem eines Unternehmens zu lösen, praktisch erzeugt und anschließend auf ihren Nutzen hin evaluiert. In diesem Paper wird u.a. auch das Design Science Paradigma mit dem Behavioural Science Paradigma verglichen. Im Vergleich zur Design Science versucht die Behavioural Science Theorien zu entwickeln und zu verifizieren, die das Verhalten von Personen und Unternehmen vorhersagt. Das Behavioural Science Paradigma ist durch die Suche nach der Wahrheit als theoretisches Gegenstück des Design Science Paradigmas zu betrachten (Hevner et al. 2004).

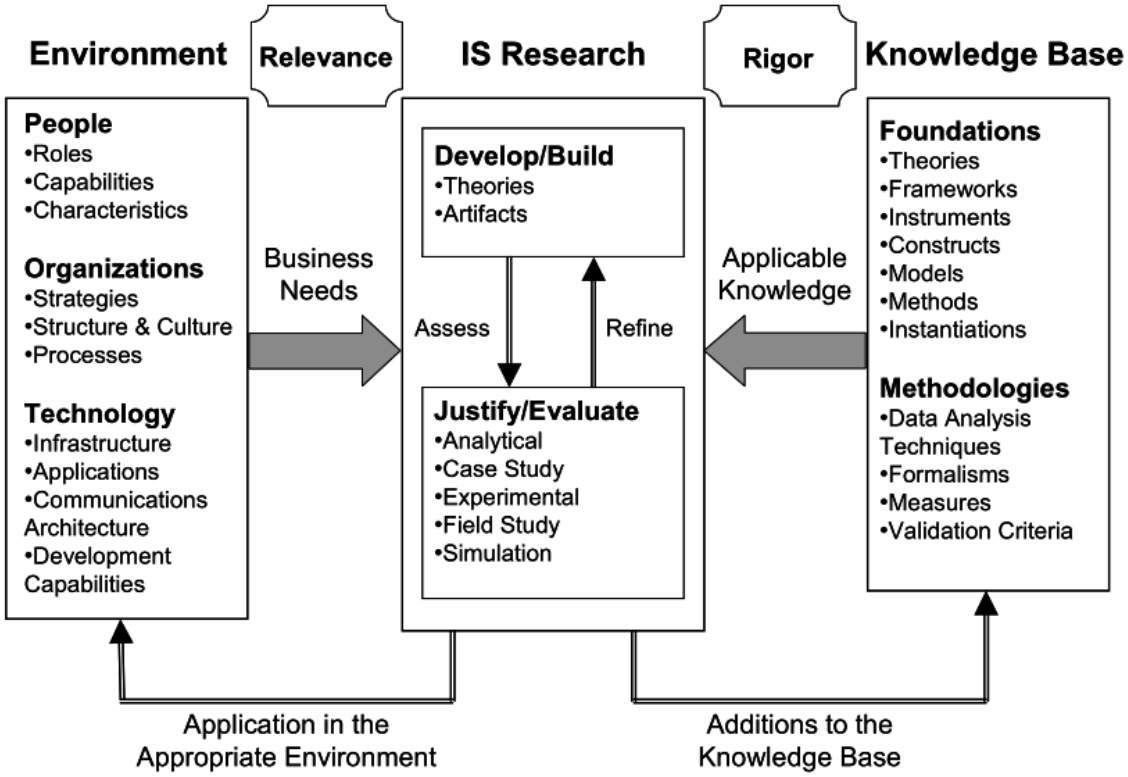


Abbildung 2: Das konzeptionelle Modell von (Hevner et al. 2004)

Abbildung 2 zeigt das konzeptionelle Modell auf dem Design Science basiert. Es gliedert sich in folgende drei große Bereiche:

- Links die organisatorische Umgebung, die sich aus Menschen, Organisationen und Technologien zusammensetzt. Gemeinsam definieren diese Komponenten das zu lösende Problem bzw. die Aufgabenstellung.
- Der mittlere Block zeigt den Bereich der Erforschung von Informationssystemen (IS Research). Für die Design Science sind die Verben „Build“ (bauen) und „Evaluate“ (auswerten) relevant, da durch sie der Nutzen für das zu lösende Problem bzw. die Aufgabenstellung identifiziert werden kann.
- Der rechte Block zeigt das Wissen, dass bei der Erforschung von Informationssystemen angewendet werden kann. Es setzt sich aus Basiskomponenten und Methoden zusammen.

Das fundamentale Prinzip der Design Science ist, dass das Wissen und das Verständnis eines Problems und seiner Lösung durch das Umsetzen und Anwenden eines Artefakts entstehen:

*„The fundamental principle of design-science research […] is that knowledge and understanding of a design problem and its solution are acquired in the building and application of an artifact.*“ (Hevner et al. 2004)

Diese Aussage ist die Basis für die sieben Richtlinien (Guidelines) der Design Science im Bereich der Erforschung von Informationssystemen (Hevner et al. 2004):

1. **Design as an Artifact**
   Es muss ein funktionierendes Artefakt erstellt werden.
   In dieser Arbeit wurde mit ERPOL ein funktionierendes Artefakt erstellt.
2. **Problem Relevance**
   Es muss eine relevante Problemstellung vorliegen.
   Die Entwicklung von ERPEL und PEPPOL hat gezeigt, dass es die Notwendigkeit der Harmonisierung im Bereich des elektronischen Datenaustauschs gibt. ERPOL kombiniert diese beiden Projekte und steigert damit die Problemlösungskompetenz beider Ansätze.
3. **Design Evaluation**
   Der Nutzen, die Qualität und die Effizienz des erstellten Artefakts muss anhand von Auswertungen überprüfbar sein.
   Durch den Use Case „Senden einer e-Rechnung an den Bund“ wurde bewiesen, dass die Übermittlung von Dokumenten von ERPEL über ERPOL und PEPPOL bis hin zu ER>B funktioniert.
4. **Research Contributions**
   Es soll klare und überprüfbare Ergebnisse im Bereich des Artefakts bzw. in dessen Herstellungsprozess geben. Das kann einerseits das Artefakt selbst sein oder andererseits ein Beitrag zur Wissensbasis bzw. zur Methodensammlung.
   In dieser Arbeit sind die wissenschaftlichen Beiträge einerseits ERPOL als Artefakt und andererseits die Ableitung des Wissens, das notwendig ist um sich an ERPEL bzw. PEPPOL anzubinden.
5. **Research Rigor**
   Design Science erfordert die Anwendung von strengen Regeln sowohl bei der Erstellung wie auch bei der Evaluierung des erstellten Artefakts. IT-Artefakte trotzen jedoch gerne exzessiven Formalismen.
   ERPOL wurde mit unterschiedlichen Formalisierungsgraden entworfen. Speziell die Schnittstellen wurden sehr genau spezifiziert und in mehreren Zyklen verbessert.
6. **Design as a Search Process**
   Die Suche nach dem besten bzw. optimalsten Design ist häufig ein sehr hartnäckiges Problem bei realen Aufgabenstellungen. Daher ist Design Science ein iterativer Prozess der aus den Phasen Generieren und Testen besteht.
   ERPOL wurde während der Entwicklungsphase immer wieder gegen eine Vielzahl von Testdokumenten geprüft und die Ergebnisse gegen ein Regelwerk überprüft. Dadurch konnten immer weitere Spezialfälle erfolgreich behandelt werden.
7. **Communication of Research**
   Das Ergebnis von Design-Science-Forschung muss sowohl für eine technische als auch für eine Management-orientierte Audienz aufbereitet werden.
   Diese Arbeit richtet sich in erster Linie an eine technische Audienz. Die Sicht auf die wirtschaftlichen und organisatorischen Aspekte wurde auch prioritär behandelt. Es ist das erklärte Ziel aus dieser Arbeit noch weitere Publikationen entstehen zu lassen.

## 1.3 Struktur der Arbeit

Die vorliegende Arbeit gliedert sich in thematisch aufeinander aufbauende Kapitel.

In Kapitel 2 werden grundlegende Technologien sowie relevante thematische Ansätze vorgestellt, die derzeit noch wenig bekannt sind. Diese bilden die Basis für die folgenden Abschnitte der Arbeit. Anschließend wird in Kapitel 3 das e-Delivery-Netzwerk ERPEL und in Kapitel 4 das europäische e-Procurement Projekt PEPPOL beschrieben. Bei beiden vorgestellten Projekten handelt es sich um in der Praxis verwendete Netzwerke zum Austausch von elektronischen Dokumenten im Bereich Beschaffung. Kapitel 5 beschreibt "ERPOL" - die Verbindung zwischen ERPEL und PEPPOL - die für diese Arbeit sowohl theoretisch spezifiziert als auch praktisch umgesetzt wurde. ERPOL besteht aus zwei Teilen: einer Registry-Schnittstelle und einer Dokumentenaustausch-Schnittstelle. Um die Praxistauglichkeit von ERPOL zu zeigen wurde eine Anbindung an "E-Rechnung an den Bund" (ER>B), die e-Rechnungs-Plattform des österreichischen Bundes, umgesetzt und in Kapitel 6 beschrieben. Abgeschlossen wird die Arbeit mit einer Zusammenfassung und Diskussion in der die offenen Punkte, entstandene Probleme, weiteres Forschungspotential und ein Ausblick enthalten sind.

# 2 Verwandte Technologien und relevante Ansätze

In diesem Kapitel werden Grundlagen der elektronischen Datenübertragung erläutert und verschiedene Arten von Dokumentenaustauschstandards beschrieben. Anschließend wird ein Überblick über die in Österreich derzeit relevanten Datenaustauschformate für elektronische Rechnungen gegeben und zum Schluss wird die XML-Validierungssprache Schematron detailliert erklärt, da diese für PEPPOL von Relevanz ist.

## 2.1 Übermittlung strukturierter Daten

Um Daten elektronisch automatisiert austauschen zu können bedarf es Regeln, die vom Sender und vom Empfänger gleichermaßen beachtet werden müssen. Diese umfassen die Struktur, Syntax und Semantik der auszutauschenden Daten (Format), die Art und Weise wie diese elektronisch übermittelt werden (Protokolle und Netzwerke) und die Geschäftsprozesse die den Ablauf des Datenaustauschs regeln.

Wenn Geschäftspartner Daten automatisiert und standardisiert austauschen wird dieser Prozess als Electronic Data Interchange (EDI) bezeichnet. Laut (National Institute of Standards and Technology 1996) kann EDI wie folgt definiert werden:

*"In EDI, the usual processing of received messages is by computer only. Human intervention in the processing of a received message is typically intended only for error conditions, for quality review, and for special situations. For example, the transmission of binary or textual data is not EDI as defined here unless the data are treated as one or more data elements of an EDI message and are not normally intended for human interpretation as part of on-line data processing."*

Um zu verhindern dass Übereinkünfte für jeden weiteren Geschäftspartner neu getroffen werden müssen, gibt es eine große Anzahl von Standardisierungsinitiativen die sich zum Ziel gesetzt haben, Standards für die Übertragung von strukturierten Daten – entweder generisch oder für bestimmte Geschäftsbereiche – zu schaffen. Geschäftsdokument-Standardisierung soll eine gemeinsame Semantik und Syntax für den Austausch zwischen Geschäftspartnern schaffen. Aus historischer Entwicklung heraus betrachtet, kann eine Unterscheidung zwischen Trennzeichen-separierten (z.B. UN/EDIFACT und ANSI X12) und Markup-basierten Sprachen (z.B. SGML, HTML, XML) getroffen werden. Vor allem seit dem Erscheinen von XML in den 1990ern ist ein eindeutiger Trend hin zu Markup-basierten Sprachen erkennbar. Das ging sogar soweit, dass einige Standards von einer Trennzeichen-basierten Syntax zu einer Markup-basierten Syntax übergegangen sind (Liegl et al. 2010).

Nach (Liegl et al. 2010) können Standards anhand ihrer technischen Eigenschaften in folgende Cluster eingeteilt werden, wobei einzelne Standards auch in mehreren Clustern vorkommen können:

1. **Top-Down-Ansätze**: Diese zielen darauf ab, möglichst alle Anforderungen zentral in einem Standard abzudecken, wodurch sich eine Übermenge aller Anforderungen ergibt. Das führt dazu, dass es sehr viele verschiedene – größtenteils optionale – Elemente gibt, wodurch sich aufgrund des Umfangs eine hohe Komplexität für die Umsetzung ergibt. Der bekannteste Standard der sich diesem Cluster zuordnen lässt ist UN/EDIFACT (United Nations Electronic Data Interchange for Administration, Commerce and

Transport). Dieser existiert seit 1987 und zeichnet sich im Unterschied zu den anderen Clustern durch zwei Releases pro Jahr aus. In den meisten anderen Clustern werden Aktualisierungen wesentlich seltener vorgenommen. Auf Grund der Größe und Komplexität des Standards ist er sehr teuer in der Umsetzung und wird daher fast ausschließlich von sehr großen Unternehmen genutzt.

2. **Bottom-Up-Ansätze**: Im Vergleich zu Top-Down-Ansätzen zielen diese darauf ab, lediglich die unbedingt notwendigen Elemente zu standardisieren und für alle optionalen Anwendungsfälle generische Erweiterungsmechanismen zur Verfügung zu stellen. Für Österreich ist ebInterface ein bekannter Vertreter dieses Ansatzes. ebInterface ist der nationale österreichische e-Rechnungs-Standard der in Kapitel 2.2.1 ausführlich beschrieben wird. Durch die Verwendung von weniger Elementen und dadurch geringerer Komplexität, kann er mit geringeren Kosten umgesetzt werden und ist daher besser für KMUs (Kleine und Mittlere Unternehmen) geeignet.
3. **Hybride Ansätze**: Diese wurden entworfen um die Vorteile von Top-Down- und Bottom-Up-Ansätzen miteinander zu vereinen. Ziel ist die Anforderungen verschiedener Interessensgruppen zu unterstützen und dabei trotzdem erweiterbar zu sein. Ein Vertreter des hybriden Ansatzes ist die von OASIS spezifizierte Sprache UBL (Universal Business Language), die auch das Datenaustauschformat von PEPPOL ist (siehe Kapitel 2.2.2). Auch das von ERPEL verwendete Dokumentenformat (siehe Kapitel 3.2) zählt zu den hybriden Ansätzen.
4. **Early Adopter der Markup-Technologie**: Damit sind Standards gemeint die sich zu einem sehr frühen Zeitpunkt von der Trennzeichen-basierten Syntax gelöst haben und sich den Markup-basierten Formaten zugewandt haben. Ein Beispiel für einen solchen Standard ist OAGIS (Open Application Group Integration Specification), die die Struktur sowohl für Geschäftsdokumente innerhalb eines Unternehmens als auch für den Austausch von Geschäftsdokumenten zwischen Unternehmen spezifiziert. In der Breite konnte sich dieser Standard aber nicht durchsetzen und er wird heute fast nur noch in der Automobilbranche verwendet. Das lässt den Schluss zu, dass eine schnelle Markteinführung nicht notwendigerweise zu einer hohen Akzeptanz führt. Ein anderer Early Adopter war xCBL (XML Common Business Library), der aber ebenfalls bereits 2003 eingestellt wurde und in UBL aufging.
5. **Integrierte Ansätze**: Die Definition eines Datenaustauschformats ist alleine nicht ausreichend um automatisiert elektronische Geschäftsdokumente austauschen zu können. Daher versuchen die integrierten Standards zusätzlich noch die Choreographie des Nachrichtenaustauschs (Geschäftsprozesse) sowie die Übertragungstechnologie zu spezifizieren. Der bekannteste Standard mit einem integrierten Ansatz ist ebXML (Electronic Business XML), das aus einer Zusammenarbeit von UN/CEFACT und OASIS entstand. ebXML besteht aus fünf Kernkomponenten mit dem Ziel, eine vollständige B2B-Infrastruktur für den automatischen Austausch von Geschäftsdokumenten zu definieren: Register (Registry), Nachrichtenaustausch (Messaging), Kollaborationsdefinitionen (Collaboration protocol profiles and agreements), Geschäftsprozessdefinitionen (Business process specifications) und den Datendefinitionen (Core Components). Durch den großen Umfang und die damit verbundenen hohen Umsetzungskosten konnte sich ebXML nie wirklich durchsetzen. Dies gilt auch für andere integrierte Ansätze.

6. **Auf Markup gewechselte Ansätze**: Auf Grund der schnellen und breiten Verbreitung von XML – auch im Zusammenhang mit serviceorientierten Architekturen (SOA) – sahen sich viele der existierenden Standards gezwungen von einer Trennzeichen-basierten Syntax auf XML zu wechseln, um besser in existierende Software-Lösungen integrierbar zu sein. Da sich aber viele Hersteller weigerten von den existierenden, gut getesteten Lösungen abzurücken, kam es dazu, dass die Standards sowohl in der klassischen Syntax als auch in XML verfügbar gemacht wurden. Als Beispiel für einen solchen Standard kann HL7 (Health Level Seven) genannt werden, ein Standard zum Austausch von Daten im Gesundheitsbereich. Auf Grund des gestiegenen Wartungsaufwands für die Pflege von verschiedenen Formaten, kann damit gerechnet werden, dass mittel- bis langfristig nur noch die Markup-basierte Version des Standards weitergepflegt wird.
7. **Syntaxneutrale Ansätze**: Das Ziel von syntaxneutralen Standards ist die Definition eines Dokumentenstandards auf einer generischen und konzeptionellen Ebene ohne dabei eine bestimmte Syntax und deren Limitationen zu berücksichtigen. Von dieser konzeptuellen Ebene können verschiedene Formate in verschiedenen Syntaxen abgeleitet werden. Alle Ableitungen haben dadurch eine gemeinsame semantische Basis. Ein bekannter Vertreter dieses Clusters ist CCTS (UN/CEFACT Core Components Technical Specification). Dieser Standard unterscheidet zwischen den generischen „Core Components“ und den Branchen-spezifischen „Business Information Entities“. Business Information Entities entstehen ausschließlich durch Einschränkung (und nie durch Erweiterung) von existierenden Core Components auf die kontextspezifischen Bedürfnisse, wodurch die gemeinsame semantische Basis erhalten bleibt. Sämtliche Core Components sind in der sog. „Core Components Library“ (CCL) zusammengefasst (http://www.unece.org/cefact/codesfortrade/unccl/ccl_index.html; Stand 7.6.2014). Auch das in PEPPOL (siehe Kapitel 4) verwendete UBL-Subset, basiert auf einer syntaxneutralen Spezifikation des CEN BII Arbeitskreises.
8. **Zusammenführende Ansätze**: Auf Grund der über die Jahrzehnte entstandenen Branchen-Standards, die alle mehr oder weniger das gleiche Ziel auf eine redundante Art und Weise verfolgen, versuchen sich Standards aus diesem Cluster als Übermenge von existierenden Standards zu positionieren. Einerseits ist es schwer möglich einen bereits existierenden alten Standard schnell abzulösen, und andererseits entstehen natürlich auch Fragen bzgl. der Kosten und des Nutzens eines solchen neuen Standards und dessen Umsetzung, da in vielen Fällen die Umsetzung des alten Standards ausreichend ist. Ein Beispiel für so eine Initiative ist UNIFI (Universal Financial Industry Message Scheme; ISO 20022), die sich mit der Zusammenführung von Dokumentenstandards im Bereich der Finanzbranche beschäftigt. Langfristig wird versucht einen neuen breiten Standard zu entwickeln, während kurzfristig sowohl der alte als auch der neue Standard nebeneinander existieren.

## 2.2 Existierende e-Rechnungs-Formate

Es gibt bereits viele verschiedene Formate für die Übermittlung von elektronischen Dokumenten im Bereich der Beschaffung. Da es speziell elektronische Rechnungsformate in großer Zahl gibt, wird nun ein kurzer Überblick über die in Österreich relevanten XML-Formate für e-Rechnungen gegeben.

### 2.2.1 ebInterface

ebInterface ist ein von der AUSTRIAPRO (die B2B-Standardisierungsplattform innerhalb der Wirtschaftskammer Österreich; http://www.austriapro.at; Stand 29.3.2014) in Kooperation mit der Wirtschaft entwickelter Standard für elektronische Rechnungen in Österreich. Die erste Version wurde bereits im Jänner 2005 veröffentlicht und seitdem ist viel Arbeit in den Standard geflossen. Die derzeit aktuelle Version ist ebInterface 4.1, die im Dezember 2013 veröffentlicht wurde. Sämtliche Versionen des Standards stehen unter http://www.ebinterface.at (Stand 29.3.2014) zum Download zur Verfügung. Darüber hinaus gibt es unter http://www.ebinterface.org (Stand 29.3.2014) ein Entwickler-Forum in dem öffentlich über die verschiedenen Teile des Standards – sowohl inhaltlich als auch technisch – diskutiert werden kann.

Bei ebInterface handelt es sich um einen XML-Dialekt mit dem ausschließlich elektronische Rechnungen abgebildet werden können. Im Vergleich zu vielen anderen e-Rechnungs-Formaten handelt es sich bei ebInterface um eine sog. „Bottom-Up"-Spezifikation, d.h. es wurden nur die Felder spezifiziert die auch wirklich verwendet werden – diese müssen jedoch auch vollständig verstanden werden. Das Pendant dazu ist eine „Top-Down"-Spezifikation bei der sämtliche denkbare Felder deklariert und anschließend Subsets definiert werden.

Neben den XML-Schema-Dateien werden auch ein XSLT-Script zur Visualisierung sowie eine ausführliche Dokumentation der einzelnen Felder angeboten. ebInterface ist in Österreich relativ weit verbreitet und wird von den meisten Branchen-Software-Produkten bereits seit einiger Zeit unterstützt. ebInterface ist auch eines jener Formate, die vom österreichischen Bund im Rahmen der „E-Rechnung an den Bund" unterstützt werden. Das ist insofern relevant, als das der österreichische Bund seit dem 1.1.2014 ausschließlich strukturierte elektronische Rechnungen und damit keine Papier-Rechnungen mehr annimmt. Dadurch sind sämtliche Firmen die als Lieferant für den Bund tätig sind, verpflichtet sich mit dem Thema „strukturierte elektronische Rechnungen" auseinanderzusetzen. Die „E-Rechnung an den Bund" war der Anlass für die Spezifikation von ebInterface 4.1, da die zum Zeitpunkt des Gesetzesbeschluss aktuelle ebInterface Version 4.0 nicht alle für den Bund relevanten Anforderungen erfüllte.

### 2.2.2 UBL

Die Universal Business Language (UBL) ist eine von der internationalen OASIS Organisation (https://www.oasis-open.org/; Stand 29.3.2014) spezifizierte XML-Bibliothek für Geschäftsdokumente. Die von UBL spezifizierten Dokumententypen werden u.a. von PEPPOL als Datenaustauschformate verwendet und von der „E-Rechnung an den Bund" akzeptiert. Die aktuelle Version ist UBL 2.1 und kann von https://www.oasis-open.org/committees/ubl/ (Stand 29.3.2014) heruntergeladen werden.

PEPPOL spezifizierte eine eigene Untermenge von UBL 2.0 u.a. für elektronische Bestellungen und Rechnungen, während bei der „E-Rechnung an den Bund" die UBL-Rechnung und -Gutschrift zum Einsatz kommen (beide UBL-Versionen werden unterstützt). UBL 2.1 ist abwärtskompatibel zu UBL 2.0 wodurch ein leichter Umstieg von 2.0 auf 2.1 gewährleistet wurde.

Im Gegensatz zu ebInterface ist UBL eine Bibliothek die eher „Top down" spezifiziert wurde. Das heißt, dass es in UBL Felder für alle möglichen Anwendungsfälle gibt und dass es nahezu unmöglich ist, den kompletten Funktionsumfang von UBL vollständig zu unterstützen.

### 2.2.3 ZUGFeRD

Bei ZUGFeRD (die Abkürzung steht für „Zentraler User Guide des Forums elektronische Rechnung Deutschland“; http://www.ferd-net.de/front_content.php?idcat=231&lang=3 Stand 29.3.2014) handelt es sich um einen in Deutschland entwickelten Standard für die Übermittlung von elektronischen Rechnungen. Im Vergleich zu ebInterface und UBL handelt es sich dabei aber nicht ausschließlich um eine XML-Datei, sondern um eine in einer PDF-Datei eingebettete XML-Datei. Die Idee hinter dieser technisch komplizierteren Lösung ist, dass es immer sowohl eine menschenlesbare als auch eine für Maschinen lesbaren Form der Rechnung in einem Dokument gibt.

Das verwendete XML-Format basiert auf dem internationalen Standard „Cross Industry Invoice“ (CII) der UN/CEFACT und bietet drei verschiedene Profile, mit unterschiedlicher Detailtiefe (FeRD 2013):

- **Basic:** die einfachste Variante, die alle für die Buchung und den Zahlungsverkehr einfacher Rechnungen notwendigen Felder enthält.
- **Comfort:** enthält die Felder vom Profil *Basic* und alles was für die möglichst vollautomatische Abwicklung von Buchungen, Zahlungen und Rechnungsprüfungen notwendig ist.
- **Extended:** enthält den Inhalt des Profils *Comfort* und weitere branchenübergreifende Felder.

Die mit ZUGFeRD verwendete PDF-Datei muss dem PDF/A-3-Standard (ISO 19005-3:2012) entsprechen. Bei PDF/A handelt es sich um ein Subset von PDF das speziell für die Langzeitarchivierung entworfen wurde. Version 3 des PDF/A Standards existiert seit 2012 und bietet erstmals die Möglichkeit andere Dateien an eine PDF/A-Datei anzuhängen.

## 2.3 Schematron

Bei Schematron handelt es sich um eine XML-basierte Sprache mit deren Hilfe XML-Dokumente kontextspezifisch überprüft werden können. Schematron ist als ISO/IEC 19757 international standardisiert, und die Spezifikation kann von der ISO Webseite unter http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833_ISO_IEC_19757-3_2006(E).zip (Stand 21.10.2012) unentgeltlich heruntergeladen werden. Sowohl ERPEL als auch PEPPOL setzen zur erweiterten Validierung Schematron ein. Durch die Verwendung von XPath in Schematron, können Regeln definiert werden, die in DTDs und XML-Schemas nicht möglich sind. Schematron wurde so generisch spezifiziert, dass nicht nur XPath, sondern auch andere Abfragesprachen verwendet werden können. In der Praxis wird jedoch hauptsächlich die XPath-Version verwendet, da die vorhandene Implementierung XSLT-basiert ist. Deshalb sind auch sämtliche Beispiele in dieser Arbeit XPath-basierend.

Schematron definiert Regeln, die auf bestimmte XML-Knoten angewandt werden. XML Schema hingegen definiert nur die Struktur und die Datentypen einer XML-Datei, bietet aber nur sehr rudimentäre Unterstützung für die Verwendung von Abhängigkeiten zwischen Elementen innerhalb eines Dokuments. Anhand eines einfachen Beispiels soll die Anwendung von Schematron beschrieben werden. Das Beispiel ist angelehnt an das offiziellen ISO Schematron Tutorium von Dave Pawson. Auf der Tutorium-Website

http://www.dpawson.co.uk/schematron/index.html (Stand 21.10.2012) befindet sich eine sehr gute und ausführliche Erklärung des gesamten ISO-Schematron-Standards.

### 2.3.1 Beispiel

Die Ausgangsbasis für die Anwendung von Schematron-Regeln ist immer ein XML-Dokument. Als Beispiel wird eine einfache XML-Datei verwendet, die ein Buch mit drei Kapiteln enthält. Jedes Kapitel soll eine eindeutige ID, einen Titel und einen Inhalt besitzen:

```
01 <?xml version="1.0" encoding="utf-8"?>
02 <doc xmlns="http://www.example.org">
03 <chapter id="c1">
04   <title>chapter title</title>
05   <para>Chapter content</para>
06 </chapter>
07
08 <chapter id="c2">
09   <title>chapter 2 title</title>
10   <para>Content</para>
11 </chapter>
12
13 <chapter>
14   <title>Title</title>
15 </chapter>
16 </doc>
```

Als erstes soll eine Regel definiert werden, die überprüft, ob jedes Kapitel einen Titel hat. Doch bevor die Regel selbst erklärt wird, zeigt das folgende Listing eine leere Schematron-Datei, die noch keine Regeln enthält.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <schema xmlns="http://purl.oclc.org/dsdl/schematron"
3         queryBinding='xslt'>
4   <title>Test ISO schematron file.</title>
5   <ns prefix='ex' uri='http://www.example.org'/>
6  <!-- Die Regeln müssen hier eingefügt werden -->
7 </schema>
```

Als XML Namespace muss `http://purl.oclc.org/dsdl/schematron` verwendet werden. Das `queryBinding`-Attribut (Zeile 3) definiert, mit welcher Abfragesprache die Regeln verarbeitet werden sollen. Im Beispiel-Schematron handelt es sich um den Standardwert XSLT 1.0. Das `title`-Element (Zeile 4) ist ein optionales und deskriptives Element, das den Titel des Regelwerks enthält. Nach dem Titel können über das `ns`-Element beliebig viele XML-Namespace Präfixes definiert werden, auf die in Regeln referenziert werden kann. Im oberen Beispiel wird in Zeile 5 das Präfix „ex“ für die Namespace-URI „http://www.example.org“ verwendet. Anschließend müssen die Regeln definiert werden.

```
1 <pattern>
2   <rule context="/ex:doc/ex:chapter">
3     <assert test="ex:title">A chapter requires a title</assert>
4   </rule>
5 </pattern>
```

Das `pattern`-Element ist eine Klammer um einen Satz von zusammengehörigen Regeln. In der Praxis gibt es häufig pro Schematron-Datei genau ein `pattern`-Element. Darin enthalten ist im

Beispiel genau eine Regel, die im `rule`-Element spezifiziert ist (Zeile 2-4). Über das `context`-Attribut wird angegeben, in welchem Kontext die Regel angewendet werden soll. In der Beispiel-Regel ist das der XPath-Ausdruck `/ex:doc/ex:chapter`, der alle direkten `chapter`-Kindelemente unterhalb des Wurzelelements `doc` auswählt. Sowohl das `doc`- als auch das `chapter`-Element müssen beiden im durch das Namespace-Präfix „ex“ vorgegebenen Namensraum liegen. Welchem XML-Namensraum das Präfix „ex“ entspricht, wurde bereits oben im `ns`-Element deklariert.

Für das Beispiel-XML-Dokument ergeben sich aus der Kontext-Regel alle drei `chapter`-Elemente. Würde man den Wert des `context`-Attributs auf `/ex:doc/ex:chapter[@id]` ändern, d.h. wenn man nur `chapter`-Elemente betrachtet die ein Attribut mit dem Namen `id` haben, so würde die Regel nur noch auf die ersten beiden `chapter`-Elemente zutreffen, und nicht mehr auf das dritte, da dieses kein `id`-Attribut hat.

Die eigentliche Überprüfung befindet sich im `assert`-Element. Es beschreibt eine Annahme, die für jeden Knoten im aktuellen Kontext überprüft wird. Trifft diese Annahme nicht zu, so wird ein Fehler erzeugt. Im Beispiel lautet die Überprüfung im Attribut `test` `ex:title`. Es wird also die Annahme getroffen, dass direkt unter jedem `chapter`-Element ein `title`-Element vorhanden sein muss. Die Annahme ist so definiert, dass sie auch zutreffen würde, wenn es mehr als ein `title`-Element unterhalb von `chapter` gibt.

Der Ausdruck im `test`-Attribut muss immer den Rückgabewert `boolean` (`true` oder `false`) besitzen. In unserem Beispiel handelt es sich aber um den Rückgabewert `node-set` (Menge von Knoten), welcher durch eine in XSLT spezifizierte, automatische Umwandlung den Wert `true` ergibt, wenn die Knotenmenge mindestens einen Knoten enthält. Wenn ein `node-set` keinen einzigen Knoten enthält, so entspricht das dem `boolean`-Wert `false`.

Als Alternative zum `assert`-Element gibt es das `report`-Element. Dieses enthält die gleichen Attribute wie das `assert`-Element und kann auch an den gleichen Stellen verwendet werden. Der einzige Unterschied zwischen dem `assert`- und dem `report`-Element ist die Test-Semantik. Bei einem `assert`-Element soll der Kontextknoten die Bedingung erfüllen, während er beim `report`-Element die Bedingung nicht erfüllen soll (Hedler et al. 2011).

Zusätzlich zu den hier vorgestellten Basis-Konstrukten gibt es noch weitere Sprachelemente wie etwa Variablen, abstrakte Regeln, Inklusionen etc. Für Details sei auf die offizielle Referenz-Dokumentation der ISO verwiesen.

### 2.3.2 Validierungsprozess

Um eine Schematron-Validierung auf ein XML-Dokument anzuwenden, sind mehrere Schritte notwendig. Die Referenzimplementierung basiert auf XSLT da es in vielen Programmiersprachen verwendet werden kann. Zuerst muss die Schematron-Datei in eine XSLT-Datei transformiert werden. Dafür gibt es drei vordefinierte XSLT-Skripten, die sequentiell angewendet werden müssen. Das finale Ergebnis der Anwendung des generierten XSLT-Skript auf ein XML-Dokument ist ein "Schematron Validation Report Language" (SVRL)-Dokument - ebenfalls ein XML-Dialekt.

1. Das erste Skript `iso_dsdl_include.xsl` löst alle inkludierten Elemente auf. Dabei kann es sich um Schematron-, XML Schema Instance- oder XLink-Includes handeln.

Außerdem werden Ableitungen durch Textersetzungen aufgelöst. Das Ergebnis ist wiederum ein Schematron-Skript. Dieser Schritt muss nicht ausgeführt werden, wenn weder Inklusionen noch Ableitungen verwendet werden.

2. Das zweite Skript `iso_abstract_expand.xsl` muss auf das Ergebnis von Schritt 1 angewendet werden. Es löst abstrakte Regeln auf. Das Ergebnis von diesem Skript ist ebenfalls ein Schematron-Skript. Falls keine abstrakten Regeln enthalten sind, kann dieser Schritt übersprungen werden.
3. Das dritte Skript `iso_svrl_for_xslt1.xsl` (für XSLT 1) oder `iso_svrl_for_xslt2.xsl` (für XSLT 2) generiert aus dem in Schritt 2 erzeugten Schematron eine XSLT-Datei, die auf XML-Dokumente angewendet werden kann, um die eigentliche Überprüfung durchzuführen. Das Ergebnis von diesem Schritt kann gespeichert werden, da es für eine bestimmte Schematron-Datei immer ident ist. Soll also ein und derselbe Schematron-Regelsatz auf verschiedene XML-Dokumente angewendet werden, so kann stets die in diesem Schritt erzeugte XSLT-Datei zum Einsatz kommen.
   Dieser Schritt eignet sich sehr gut um einmalig als Pre-processing ausgeführt zu werden. Die Anwendung der Schritte 1-3 ist bei großen Schematron-Dateien sehr zeit- und speicheraufwendig, und sollte daher so selten wie möglich ausgeführt werden.
4. Das Ergebnis aus Schritt 3 kann nun auf das zu überprüfende XML-Dokument angewandt werden. Das Ergebnis ist das SVRL-Dokument, in dem die Validierungsergebnisse zusammengefasst sind.

Alternativ dazu gibt es die Möglichkeit Schematron mit einer nativen Implementierung auszuführen. Beispielsweise gibt es mit "ph-schematron" (https://github.com/phax/ph-schematron/ - Stand 01.09.2014) eine native Java OSS Implementierung, die deutlich bessere Performance bei der Validierung bietet, als die herkömmlichen XSLT-Umsetzungen.

### 2.3.3 Validierungs-Ergebnisse

Das Ziel von Schematron ist die Validierung von XML-Dokumenten. Das Ergebnis dieser Validierung kann entweder ein Einfaches "ist gültig" oder "ist ungültig" sein, oder ein detailliertes Ergebnis in dem aufgelistet ist, welche Regeln aufgerufen wurden und welche Assertions fehlgeschlagen sind.

Im einfachen Fall müssen nur jene Regeln ausgeführt werden, bis es zur ersten fehlgeschlagenen Assertion kommt. In diesem Fall wäre das XML-Dokument also ungültig. Für ein gültiges XML-Dokument müssen aber in jedem Fall alle Assertions getestet werden, egal ob die einfache oder die komplexe Validierung durchgeführt wird.

Innerhalb von Schematron wird die "Schematron Validation Report Language" (SVRL) als XML-Dialekt zur Darstellung der Ergebnisse verwendet. SVRL ist ein einfacher Ablaufreport der Validierung. Er enthält jedes ausgeführte Pattern, jede Regel sowie alle fehlgeschlagenen Assertions und alle erfolgreichen Reports.

```
01 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
02 <schematron-output xmlns="http://purl.oclc.org/dsdl/svrl">
03   <active-pattern />
04   <fired-rule context="AAA" />
```

```
05   <successful-report test="BBB" location="AAA">
06     <text>BBB element is present.</text>
07   </successful-report>
08   <failed-assert test="@name" location="AAA">
09     <text>AAA misses attribute name.</text>
10   </failed-assert>
11   <active-pattern />
12   <fired-rule context="AAA" />
13   <successful-report test="BBB" location="AAA">
14     <text>BBB element is present.</text>
15   </successful-report>
16   <active-pattern />
17   <fired-rule context="AAA" />
18   <failed-assert test="@name" location="AAA">
19     <text>AAA misses attribute name.</text>
20   </failed-assert>
21 </schematron-output>
```

Die dargestellte SVRL-Datei enthält drei aufgerufene Patterns (im Element `active-pattern`, Zeilen 3, 11 und 16), mit jeweils einer aufgerufenen Regel (im Element `fired-rule`, Zeilen 4, 12 und 17) sowie jeweils zwei fehlgeschlagene Assertions (im Element `failed-assert`, Zeilen 8-10 und 18-20) und erfolgreiche Reports (im Element `successful-report`), Zeilen 5-7 und 13-15). Da mindestens ein `failed-assert`- bzw. `successful-report`-Element vorhanden ist, war die Validierung in diesem Fall nicht erfolgreich.

# 3 ERPEL

Bei ERPEL (E-Business Registry Permitting Enterprise Liaisons) handelt es sich um ein Forschungsprojekt der TU Wien mit dem Ziel, das Hauptproblem im B2B E-Commerce zu lösen: Die Heterogenität von IT-Lösungen zwischen Geschäftspartnern (Huemer et al. 2012).

Die Heterogenität kann in folgende drei Aspekte unterteilt werden:

1. Die Suche nach Geschäftspartnern.
2. Die Formate der ausgetauschten Dokumente.
3. Die Übertragung der elektronischen Dokumente zwischen Geschäftspartnern.

Das ERPEL-Projekt fokussiert sich auf den Bereich E-Procurement (elektronische Beschaffung) und bietet dort folgende Lösungen für alle genannten Aspekte an:

1. Durch eine, um semantische Aspekte angereicherte, Registry wird die Suche nach Geschäftspartnern vereinfacht („ERPEL Registry“).
2. Ein einheitlicher Dokumenttyp für ausgetauschte Dokumente.
3. Ein einheitliches Protokoll und eine zentrale Koordinierungsstelle für den Austausch von Geschäftsdokumenten („Business Service Interface“ und „Messaging Hub“).

## 3.1 Überblick

Im Vergleich zu anderen Projekten verfolgt ERPEL keinen rein technischen Ansatz, sondern betrachtet auch die fachliche Ebene. Abbildung 3 zeigt die Gesamtarchitektur von ERPEL. Auf fachlicher Ebene werden „Business Processes“ definiert (Kapitel 3.3), die den Austausch der „Business Documents“ beschreiben (Kapitel 3.2). Auf der Infrastrukturebene befinden sich die „ERPEL Registry“ (Kapitel 3.4) und die „BSIs“ (Kapitel 3.5). Alle Komponenten wurden auf die rechtlichen Grundlagen und auf die im Einsatz befindlichen ERP-Systeme der Projektpartner abgestimmt.

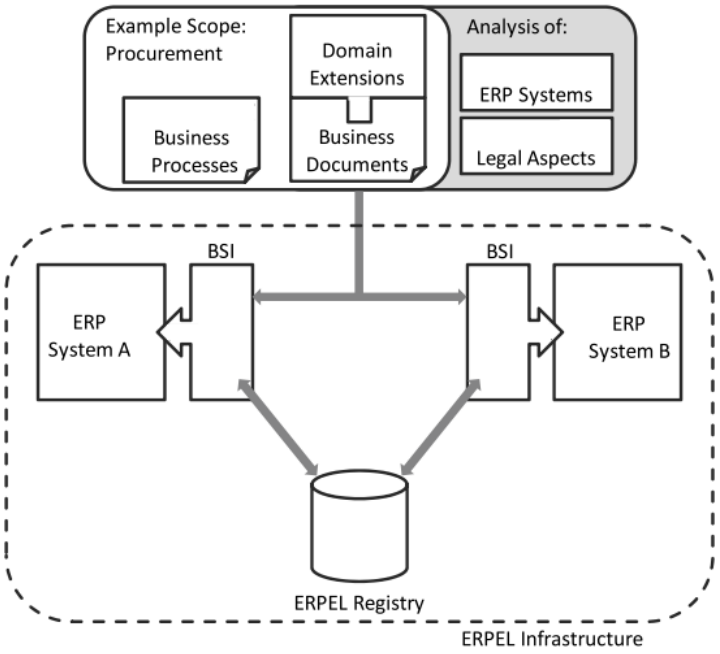


**Abbildung 3: ERPEL Architektur Überblick (TU Vienna, Institute of Software Technology and Interactive Systems 2009)**

Die Informationen zu einem Großteil der in diesem Kapitel beschriebenen Elemente sind im ERPEL Wiki abrufbar, welches jedoch nicht öffentlich zugänglich ist, sondern nur für Projektpartner zur Verfügung steht.

ERPEL ist für alle Firmen offen, die daran teilnehmen möchten. Ursprünglich war eine Schnittstelle zum „Firmen A-Z“ der WKO (http://firmen.wko.at - Stand 23.7.2014) geplant, um die Registry mit den Daten existierender Firmen zu füllen. Auf Grund von organisatorischen Hindernissen wurde diese Schnittstelle bis heute noch nicht umgesetzt. Deshalb sind bisher nur Pilotpartner als Unternehmen in der Registry enthalten.

## 3.2 Geschäftsdokumente

ERPEL spezifiziert einen eigenen XML-basierten E-Procurement-Dokumenttyp für die einheitliche Übermittlung elektronischer Geschäftsdokumente. Der ERPEL-Dokumenttyp wurde „Bottom-up“ definiert. Es wurde demnach nur die Schnittmenge der Felder definiert, die von allen Branchen benötigt werden und ist dementsprechend klein. Andere Spezifikationen wie ebXML oder UBL sind „Top-down“ definiert, enthalten also die Übermenge aller branchenspezifischen Elemente. Abbildung 4 zeigt den Unterschied zwischen „Top-down“ und „Bottom-up“ Spezifikation.

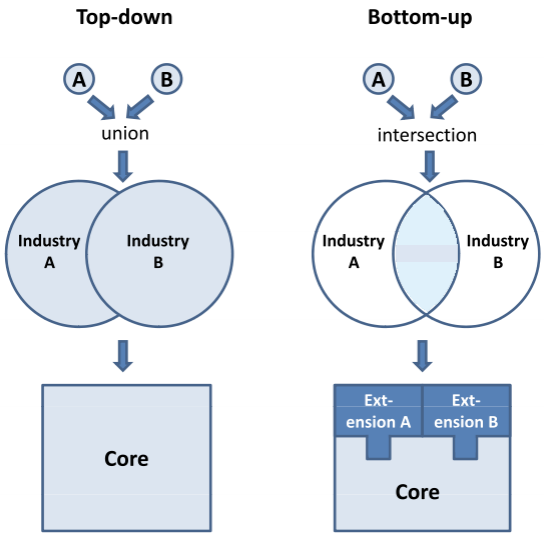


Abbildung 4: Top-down und Bottom-up Spezifikation (Huemer et al. 2012)

Um branchenspezifische Erweiterungen abbilden zu können, gibt es in ERPEL ein eigenes standardisiertes Erweiterungsschema. Das Besondere am ERPEL Ansatz ist, dass es genau einen Dokumenttyp und daher auch nur eine zentrale XML Schemadefinition für alle Dokumente gibt. Die spezifischen Ausprägungen für die unterschiedlichen Szenarien werden als Sub-Dokumenttypen bezeichnet. Abbildung 5 zeigt den Zusammenhang zwischen dem ERPEL-Dokumenttyp, den Sub-Dokumenttypen und den optionalen Erweiterungen.

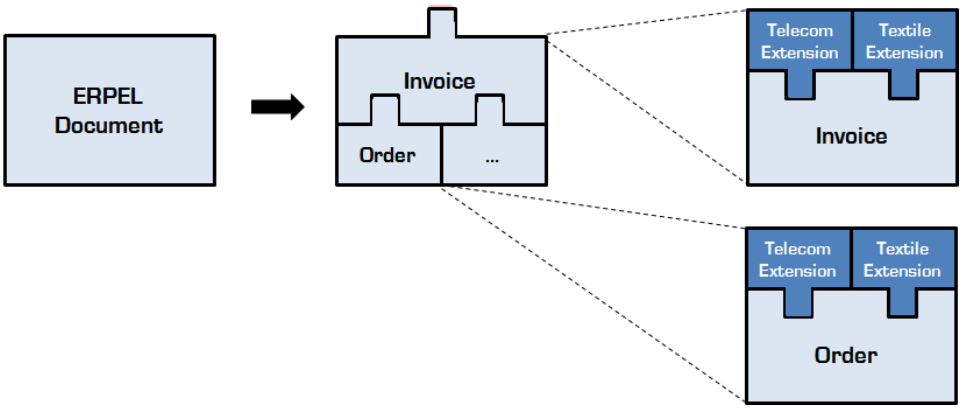


Abbildung 5: ERPEL Dokumenten Struktur (Liegl et al. 2013)

ERPEL unterstützt 6 verschiedene Sub-Dokumenttypen, die primär im Bereich E-Procurement zum Einsatz kommen. Die Ausgangsbasis des zentralen Dokumenttyps ist bei ebInterface, dem offiziellen E-Rechnungsstandard der WKO, zu finden. Daher gibt es sehr viele Übereinstimmungen und eine Transformation einer ERPEL Invoice von und nach ebInterface ist dementsprechend einfach. Die unterstützten Sub-Dokumenttypen sind (Liegl et al. 2013):

- Anfrage – „Request for Quote“:
  Im ERP-Kontext und aus der Sicht des Lieferanten beschreibt es eine Anfrage von genau einem Kunden an einen oder mehrere Lieferanten. Es ist eine Aufforderung ein Angebot zu legen.

- Angebot – „Offer“:
  Im ERP-Kontext und aus der Sicht des Kunden ist Offer üblicherweise ein Einzeldokument von genau einem Lieferanten.
- Bestellung – „Order“:
  Eine Bestellung unterteilt sich in eine Abruf-, Kontrakt- oder Normalbestellung.
- Auftragsbestätigung – „Order Confirmation“:
  Diese bezieht sich auf die Subtypen gemäß Bestellung.
- Lieferschein – „Dispatch Advice“:
  Dieser unterteilt sich in einen Teil-, Sammel- oder Normallieferschein.
- Rechnung – „Invoice“:
  Die Rechnungs-Subtypen sind gemäß ebInterface:
  - Invoice – Rechnung
  - FinalSettlement – Endabrechnung
  - InvoiceForAdvancePayment – Vorauszahlung
  - InvoiceForPartialDelivery – Rechnung für Teillieferung
  - SubsequentCredit – Nachentlastung
  - CreditMemo – Gutschrift
  - SubsequentDebit – Nachbelastung
  - SelfBilling – Gutschriftverfahren

In Abbildung 6 ist die übliche Abfolge eines Dokumentenaustauschs der ERPEL Sub-Dokumenttypen dargestellt.

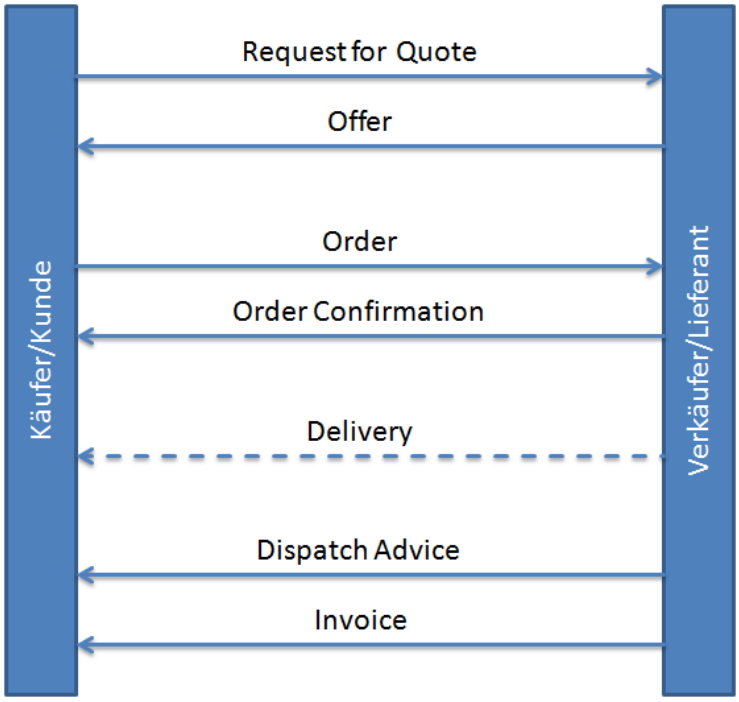


Abbildung 6: Übliche Abfolge beim Austausch der ERPEL Dokumenttypen (Liegl et al. 2013)

### 3.2.1 Technische Umsetzung

Alle ERPEL-Dokumente müssen valide XML-Instanzen sein. Die Elemente sind im XML Namespace `http://erpel.at/schemas/1p0/documents` deklariert. Die Unterscheidung,

um welchen Sub-Dokumenttyp es sich handelt, ist dem Attribut `DocumentType` aus dem XML-Wurzelelement zu entnehmen. Der folgende Ausschnitt aus einem ERPEL-Dokument zeigt, dass es sich um eine Rechnung handelt:

```
<erpel:Document erpel:DocumentType="Invoice"
erpel:IsGrossPrice="true" erpel:DocumentCurrency="EUR"
erpel:GeneratingSystem="String"
xmlns:erpel="http://erpel.at/schemas/1p0/documents">
     <erpel:BusinessProcessID>4</erpel:BusinessProcessID>
     <erpel:DocumentNumber>5</erpel:DocumentNumber>
     <erpel:DocumentDate>2011-04-12</erpel:DocumentDate>
…
</erpel:Document>
```

Da das verfügbare Regelwerk von XML Schema relativ klein ist, gibt es zusätzlich noch für alle Sub-Dokumenttypen eigene Schematron-Regeln (siehe Kapitel 2.3), die zusätzlich angewandt werden sollen, um die Integrität der Dokumente zu gewährleisten. Beispiel einer einzelnen Schematron-Regel, die überprüft, ob das `DocumentType`-Attribut für eine Rechnung gültig ist:

```
<pattern>
     <rule context="/erpel:Document">
          <assert test="@erpel:DocumentType='Invoice' or
@erpel:DocumentType='InvoiceFinalSettlement' or
@erpel:DocumentType='InvoiceForAdvancePayment' or
@erpel:DocumentType='InvoiceForPartialDelivery' or
@erpel:DocumentType='InvoiceSubsequentCredit' or
@erpel:DocumentType='InvoiceCreditMemo' or
@erpel:DocumentType='InvoiceSubsequentDebit' or
@erpel:DocumentType='InvoiceSelfBilling'">
            Regel 28: Der DocumentType entspricht nicht einer
Invoice' or 'FinalSettlement' or 'InvoiceForAdvancePayment' or
'InvoiceForPartialDelivery' or 'SubsequentCredit' or 'CreditMemo'
or 'SubsequentDebit' or 'SelfBilling'</assert>
     </rule>
</pattern>
```

Für die Bestellung sieht die entsprechende Schematron-Regel wie folgt aus:

```
<pattern>
     <rule context="/erpel:Document">
          <assert test="@erpel:DocumentType='Offer'">
            Regel 21: Der DocumentType entspricht nicht einer
Offer</assert>
     </rule>
</pattern>
```

Optional werden auch elektronische Signaturen nach dem XMLDSig-Standard unterstützt. Es gibt keine Vorgaben bzgl. des Kanonikalisierungs-Algorithmus, des Hash-Algorithmus und der Transformations-Operationen.

## 3.3 Prozesse

Neben den Dokumenttypen wurden in ERPEL auch Standard-Prozesse für den Bereich E-Procurement entwickelt. Das Ziel war es, einfache, auch für KMUs umsetzbare, Standard-Geschäftsprozesse zu definieren, die von allen Beteiligten umgesetzt werden können bzw.

bereits umgesetzt sind. Zur technischen Definition der Prozesse wurden standardisierte Choreographien, das ist die genaue Beschreibung des Austauschs der technischen Dokumente, entwickelt. Die Spezifikation der Semantik der ausgetauschten Datenelemente für den Bereich E-Procurement befindet sich in den in Kapitel 3.2 beschriebenen Sub-Dokumenttypen. Die ERPEL Registry (siehe Kapitel 3.4) enthält pro Teilnehmer eine Liste der unterstützten Geschäftsprozesse. Der Austausch der elektronischen Dokumente erfolgt über das standardisierte Business Service Interface (BSI; siehe Kapitel 3.5) (TU Vienna, Institute of Software Technology and Interactive Systems 2009).

### 3.3.1 Theoretische Grundlagen

Sowohl zur Erstellung von Prozessen als auch zur Definition von Dokumenttypen können Technologien eingesetzt werden, die im BSopt Projekt (Business Semantics on top of Process Technology; http://www.bsopt.at/ - Stand 23.7.2014) entwickelt wurden. Zur Erstellung der Prozesse kann die von der TU Wien im Rahmen des BSopt Projekts entwickelte Domain Specific Language (DSL) namens „Business Choreography Language“ (BCL) angewendet werden.

BCL basiert auf den Konzepten der „UN/CEFACT Modelling Methodology“ (UMM) sowie den Best Practices anderer Notationen wie z.B. BPMN. Für ERPEL wurde sie um Konzepte der semantischen Geschäftsprozesse erweitert. BCL basiert auf folgenden Konzepten, die sowohl dem UMM zugrundeliegenden UML-Profil als auch BPMN entnommen sind (Motal et al. 2009):

- **Business collaboration**: Ein von zwei oder mehreren Geschäftspartnern gemeinsam ausgeführter Geschäftsprozess, der aus einer oder mehreren *business transactions* besteht.
- **Business transaction**: Definiert einen Informationsaustausch zwischen genau zwei Geschäftspartnern und basiert auf *business transaction patterns*.
- **Business transaction patterns**: In UMM wird zwischen zwei unidirektionalen und vier bidirektionalen Mustern unterschieden, die sich durch die Antwortdokumente bzw. den *Quality of Service Parametern* unterscheiden.
- **Quality of service parameters**: Definieren Sicherheits- und Timeout-Einstellungen sowie technisch notwendige Bestätigungen
- **Business documents**: Enthalten die eigentlichen Geschäftsinformationen die innerhalb einer *business transaction* ausgetauscht werden.
- **Shared states**: Bilden den gemeinsamen Status aller involvierten Geschäftspartner während der Ausführung einer *business collaboration* ab.
- **Re-use of business transactions**: Die Wiederverwendung von *business transactions* in verschiedenen *business collaborations*.
- **Role mapping**: Die Abstraktion von spezifischen Geschäftspartnern durch Rollen, um die Wiederverwendung von *business transactions* zu erleichtern.
- **Timer events**: Definiert die absolute bzw. relative Verzögerung einer Aktivität in einer business collaboration.
- **Compensations**: Spezielle *business transactions* die im Fehlerfall zum Rückgängigmachen bereits ausgeführter *business transactions* dienen.
- **Event-based XORs**: Im Vergleich zum herkömmlichen XOR wird nicht anhand der Daten entschieden, welche Verzweigung im Prozess genommen werden soll, sondern anhand eines externen Ereignisses.

### 3.3.2 Referenzprozess

ERPEL hat für die Domäne E-Procurement einen Referenzprozess zwischen Lieferant und Käufer definiert. Der Prozess enthält den optionalen Austausch von Anfragedokument, Angebot, Bestellung, Auftragsbestätigung und Lieferschein, sowie den verpflichtenden Austausch der Rechnung.

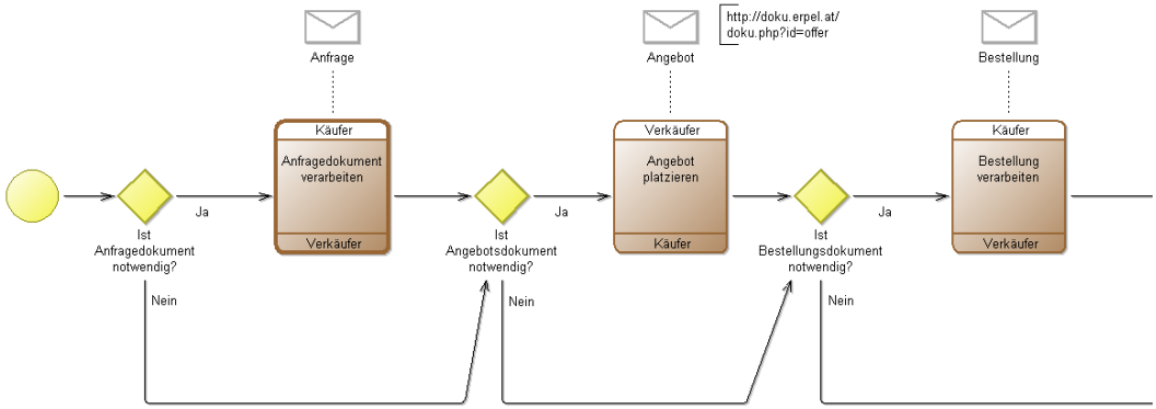


Abbildung 7: ERPEL Referenzprozess Teil 1 (Liegl et al. 2013)

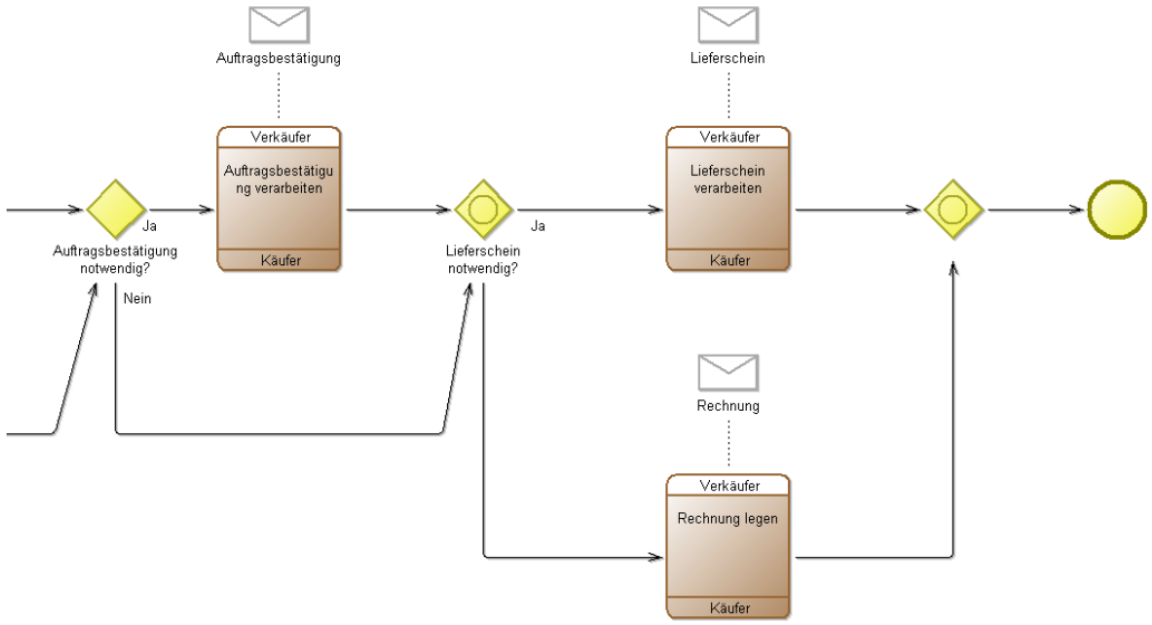


Abbildung 8: ERPEL Referenzprozess Teil 2 (Liegl et al. 2013)

Abbildung 7 und Abbildung 8 zeigen den Prozess grafisch in der BPMN 2.0 Notation. Fast alle Entscheidungen sind exklusive ODER, d.h. es wird immer nur der eine oder der andere Pfad gegangen. Bei der Entscheidung „Lieferschein notwendig?“ wird der Pfad zur Rechnung immer genommen, während der Pfad zum Lieferschein optional ist und der Austausch der Dokumente grundsätzlich nebenläufig erfolgen kann (Liegl et al. 2013).

## 3.4 Registry

ERPEL speichert die Daten aller am Netzwerk teilnehmenden Firmen und Personen. Neben den Geschäftsdaten und den Details der Firma (Name, Anschrift, Kontakt, Rechtform, Bankverbindung etc.) werden auch Repräsentationsdetails, Produkte, Bewertungen und

Verbindungen zu anderen Firmen innerhalb des ERPEL Netzwerks gespeichert. Die ERPEL Registry ist eine kontinuierliche Weiterentwicklung von existierenden Registries, wie etwa der ebXML Registry oder UDDI. Beide Registry-Typen sowie die Motivation eine neue Registry zu spezifizieren, werden im Folgenden näher erläutert (TU Vienna, Institute of Software Technology and Interactive Systems 2009).

### 3.4.1 ebXML Registry

„Die Vision von ebXML ist es, einen einzigen globalen Marktplatz zu erzeugen, auf dem sich Geschäftstreibende finden, Partner werden und dann miteinander Geschäfte machen können. All diese Operationen sollen durch den automatischen Austausch von standardisierten XML Dokumenten durchgeführt werden.“ (Hofreiter et al. 2002)

Um die Daten der Geschäftstreibenden zu speichern, wurde von der ebXML Initiative das „ebXML Registry Information Model“ (ebRIM) entwickelt, dass von der ebXML Registry implementiert wird. Dabei handelt es sich um ein generisches XML-basiertes Metamodell für Registry-Dienste. Genau diese Generizität ist eines der Hauptprobleme der ebXML Registry: so können etwa Produkte und Dienstleistungen nicht standardmäßig abgebildet werden. Stattdessen muss die Registry so angepasst werden, dass die Interoperabilität nicht mehr sichergestellt werden kann. Daher ist die ebXML Registry nur als technisches Framework zum Aufbau einer B2B Registry zu verwenden. Sie ist nicht als ein „Out-of-the-box“ Produkt für den elektronischen Geschäftsverkehr einsetzbar (TU Vienna, Institute of Software Technology and Interactive Systems 2009).

### 3.4.2 UDDI

UDDI ist das Registry-Service der Web Service Technologien, das sich in drei Teile gliedert: White Pages (Firmendaten), Yellow Pages (Klassifikation der Leistungen eines Anbieters) und Green Pages (die eigentlichen Web Service Schnittstellendaten). Seit die meisten UDDI Registries, die von großen Anbietern betrieben wurden, rund um 2005 abgeschaltet wurden, ist es sehr ruhig um UDDI geworden. Das Scheitern von UDDI kann u.a. auf folgende drei Gründe zurückgeführt werden (TU Vienna, Institute of Software Technology and Interactive Systems 2009):

- UDDI war nur für einfache Dienste geeignet, und konnte keine Abhängigkeiten zwischen Services darstellen. In der Realität ist der Austausch von verschiedenen Nachrichten zwischen unterschiedlichen Partnern im Rahmen von Choreographien aber ein immanenter Bestandteil.
- UDDI ist ein rein technischer Ansatz und lässt viele wirtschaftliche Aspekte außen vor.
- UDDI hatte große Probleme mit vielen gefälschten Einträgen, da es keine Überprüfung der eingetragenen Werte gab.

### 3.4.3 Architektur

ERPEL hat sich zum Ziel gesetzt eine Registry zu entwickeln, die den Anforderungen von modernem B2B entspricht, ohne dabei die Fehler von ebXML und UDDI machen. Daher wurden anstelle von technischen Services ganze Geschäftsprozesskollaborationen als integraler Bestandteil in die Registry inkludiert. Neben den technischen Eigenschaften von Einträgen wurde auch ganz besonderer Wert auf die geschäftlichen Eigenschaften gelegt. Zusätzlich wird zur Vermeidung von gefälschten Einträgen ein zweistufiger Vertrauensmechanismus herangezogen.

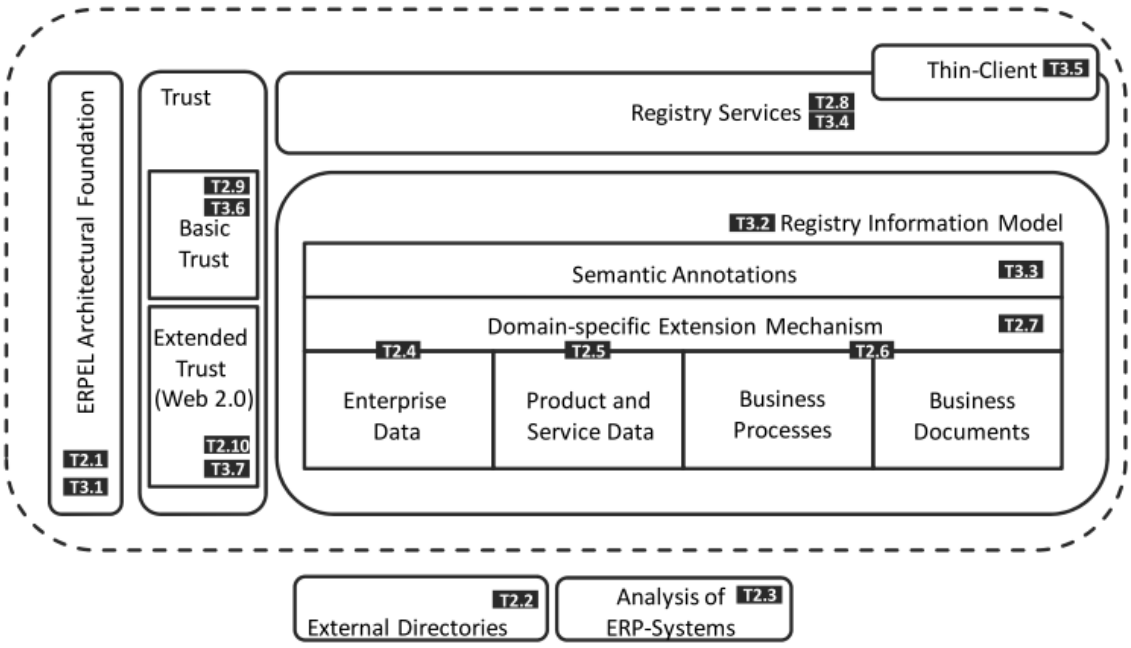


**Abbildung 9: ERPEL Registry Architektur (TU Vienna, Institute of Software Technology and Interactive Systems 2009)**

Die Kernkomponente der ERPEL-Registry ist das „Registry Information Model“ (T3.2). Es beruht auf den vier Säulen Unternehmensdaten (T2.4), Produkte und Dienstleistungen (T2.5), Geschäftsprozesse (T2.6) und Geschäftsdokumente (T2.6). Mit Hilfe dieser Elemente ist es möglich neue Geschäftspartner zu finden und elektronische Transaktionen durchzuführen. Um komplexere Anforderungen von bestimmten Geschäftszweigen abbilden zu können, wurde ein generischer Erweiterungsmechanismus eingeführt (T2.7) (TU Vienna, Institute of Software Technology and Interactive Systems 2009).

Mit Hilfe von semantischen Annotationen (T3.3) können syntaktische Unterschiede zwischen Elementen mit unterschiedlichem Namen aber gleichem Inhalt überbrückt werden. Die auf OWL DL basierende *GoodRelations*-Ontologie wurde für die Verwendung in der ERPEL-Registry vorgeschlagen. Mit einem einheitlichen Vokabular adressiert sie die Beschreibung der Relationen zwischen

- Web Ressourcen,
- Angebote die von diesen Web Ressourcen ausgehen,
- Juristischen Einheiten,
- Preisen,
- Geschäftsbedingungen und
- den Produkt- und Service-Ontologien (Huemer et al. 2012)

Zur Abbildung der Daten wurden existierende Verzeichnisdienste (T2.2), wie etwas das Firmenregister der Wirtschaftskammer, und existierende Software-Produkte (T2.3) untersucht, um eine möglichst vollständige und korrekte Abbildbarkeit zu erreichen. Dies gilt sowohl für die Unternehmensdaten als auch für die Produkt- und Dienstleistungsbeschreibungen.

Zur Absicherung der Daten und zur Vermeidung von Fake-Einträgen, werden diese mit großen, vertrauenswürdigen Verzeichnissen abgeglichen. Dadurch soll sichergestellt werden, dass nur

gültige Einträge in der Registry gespeichert werden. Diese Maßnahme ist in Abbildung 5 als „Basic Trust“ dargestellt. Die „Extended Trust“ Ebene soll mit Hilfe von Mitteln sozialer Netzwerke umgesetzt werden. Durch das gegenseitige Vertrauen von Geschäftspartnern soll ein Vertrauensnetzwerk („Network of trust“) aufgebaut werden, das den Teilnehmern im ERPEL Netzwerk die Möglichkeit gibt, Rückschlüsse über die Vertrauenswürdigkeit der anderen Unternehmen zu ziehen („Friend-of-a-Friend“ Ansatz) (TU Vienna, Institute of Software Technology and Interactive Systems 2009). Diese erweiterte Vertrauensebene wurde theoretisch erarbeitet, jedoch bisher noch nicht in die Praxis umgesetzt, da es sich dabei um ein sehr komplexes Thema handelt.

Der Zugriff auf die Registry erfolgt entweder manuell über eine Webseite oder über eine Web Service-Schnittstelle. Manche Daten – wie z.B. die Produktdaten – können nicht über die Webseite, sondern nur das Web Service geändert werden. Das Web Service wird auch für die automatische Replikation der Daten aus ERP-Systemen heraus verwendet. Das ERP System bleibt das führende System. Eventuelle Änderungen an den Stamm- oder Produktdaten sollten jedoch in die ERPEL-Registry repliziert werden. Dazu stehen die üblichen CRUD Operationen (*create*, *read*, *update* und *delete*) nach einer Authentifizierung über Benutzername und Passwort zur Verfügung.

## 3.5 Messaging Plattform

ERPEL stellt eine einheitliche Lösung zur elektronischen Übermittlung von Geschäftsdokumenten zur Verfügung: ein Business Service Interface (BSI) als Adapter zu existierenden Software-Lösungen, den Messaging Hub als zentrale Datendrehscheibe und das Protokoll zur Kommunikation zwischen einem BSI und dem Messaging Hub.

### 3.5.1 Business Service Interface

Ein Business Service Interface (BSI) ist ein Adapter zu existierenden Software-Lösungen (z.B. zu ERP-Systemen). Auf der einen Seite muss ein BSI die standardisierte Web-Service Schnittstelle zum Messaging Hub (*Deliver* und *Fetch*) implementieren, auf der anderen Seite muss er an die jeweilige Software-Lösung angepasst sein. Pro Version einer angebundenen Software-Lösung muss das BSI nur einmal implementiert werden, kann aber mehrfach verwendet werden.

Da die meisten angebundenen Software-Produkte das ERPEL-Dokumentenformat nicht nativ implementieren, ist das BSI auch ein geeigneter Ort zur Durchführung einer Dokumententransformation von oder in das ERPEL-Dokumentenformat. Ein BSI kann in jeder beliebigen Programmiersprache umgesetzt werden, solange den Protokoll-Anforderungen des Messaging Hubs entsprochen werden kann.

### 3.5.2 Messaging Hub

Sämtlicher Datenaustausch im ERPEL-Netzwerk erfolgt über einen zentralen Messaging Hub. Die folgende Abbildung zeigt eine schematische Darstellung des Messaging Hubs im Datenaustausch:

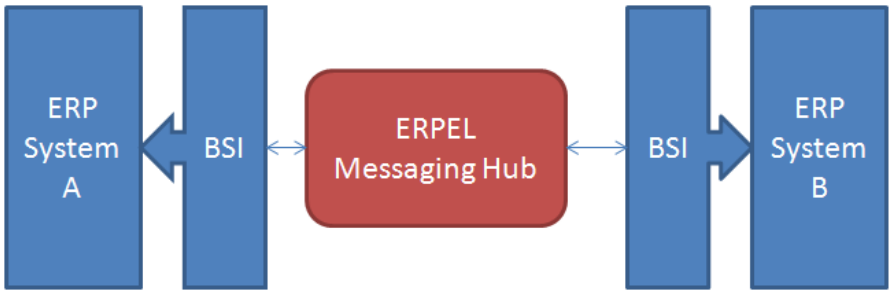


Abbildung 10: ERPEL Messaging Hub im Datenaustausch

Der Messaging Hub ist eine zentrale Komponente und daher gibt es exakt eine Instanz im ERPEL-Netzwerk. Der Messaging Hub dient als Drehscheibe für den Austausch von jeglichen Geschäftsdokumenten zwischen Geschäftspartnern – ein direkter Dokumentenaustausch zwischen Geschäftspartnern ohne Einbeziehung des Messaging Hubs ist nicht angedacht. Er erlaubt auch eine temporäre Speicherung der Daten, falls der Empfänger zum Zeitpunkt der Datenübertragung nicht erreichbar ist. In diesem Fall wird das Dokument solange vom Messaging Hub vorgehalten, bis der Empfänger es abruft. Die Speicherung eines Dokuments passiert jeweils im Kontext eines Benutzers in sogenannten *MessageBoxen* die schematisch einem E-Mail-Konto gleichzusetzen sind (Liegl et al. 2013).

Der Messaging Hub bietet zwei atomare Operationen an: *Deliver* und *Fetch*. Mit Hilfe von *Deliver* werden Nachrichten von einem BSI an den Messaging Hub übertragen. Die *Fetch* Operation dient dem Abholen von Nachrichten durch einen BSI. Beide Operationen liefern synchron ein Ergebnis zurück, welche als technische Antwort dient. Der Messaging Hub prüft nach Einlangen eines Dokuments, ob der Empfänger im System bekannt ist, und fügt im Erfolgsfall das übertragene Dokument in die MessageBox des Empfängers ein und eine automatisch generierte Übermittlungsbestätigung (*DeliverAckOfDelivery*) in die MessageBox des Senders. Falls der Empfänger unbekannt ist, wird eine automatisch generierte Fehlermeldung (*ControlFailure*) in die MessageBox des Senders gespeichert. Wurde ein Geschäftsdokument erfolgreich übertragen ist also sowohl für den Sender als auch für den Empfänger jeweils ein Dokument in der MessageBox abrufbereit. Sobald der Empfänger das Original-Geschäftsdokument erfolgreich abgerufen hat, wird wiederum eine automatisch generierte Abrufbestätigung (*DeliverAckOfPickUp*) in die MessageBox des Senders gelegt, die dieser dann wieder *fetchen* (abholen) kann. Die folgende Grafik zeigt den Zusammenhang im Erfolgsfall (Liegl et al. 2013):

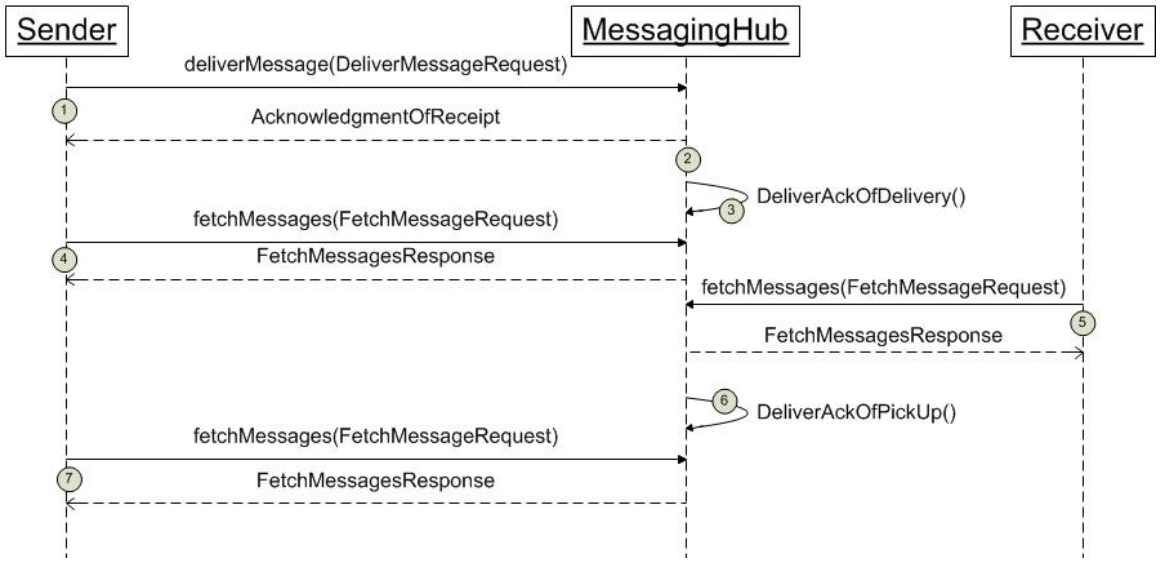


**Abbildung 11: Ablauf einer erfolgreichen Dokumentenübermittlung in ERPEL (Liegl et al. 2013)**

Neben den Operationen zum Datenaustausch bietet der Messaging Hub auch eine Web Service Operation zum Validieren von beliebigen ERPEL-Geschäftsdokumenten an. Diese bietet registrierten Teilnehmern des ERPEL-Netzwerks die Möglichkeit, Geschäftsdokumente vor dem Absenden gegen das XML Schema und die erweiterten Schematron-Regel zu prüfen. Falls ein zu validierendes Dokument ungültig ist, wird eine ausführliche Fehlerbeschreibung in der jeder einzelne Fehler detailliert beschrieben ist, an den Aufrufer übermittelt (Liegl et al. 2013).

Als zentrale Komponente im ERPEL-Netzwerk muss der Messaging Hub redundant ausgelegt sein, da ein Ausfall den gesamten Datenaustausch lahm legen würde. Auch ergeben sich an den Messaging Hub besondere Anforderungen bzgl. Performance und Antwortzeiten, da er mit allen speziellen BSIs kommuniziert und nur bedingt Einfluss auf deren Parametrisierung hat. Als Ausbaustufe zur Kommunikation mit anderen Netzwerken ist der Messaging Hub auch die geeignete Stelle zur Durchführung von Transformationen zwischen verschiedenen Dokumenttypen.

### 3.5.3 Protokoll

Der Austausch von Nachrichten innerhalb des ERPEL-Netzwerks erfolgt über ein im Rahmen des Projekts entwickeltes Protokoll namens „ERPEL document exchange protocol“. Die Ziele bei der Spezifikation des Protokolls waren a) der sichere und verlässliche Dokumentenaustausch, b) das Routing der Dokumente und c) die Archivierung der Dokumente. Als Datenübertragung wurde Web Services über SMTP (also Web Services „per E-Mail“) gewählt, um speziell KMUs eine einfache Möglichkeit zu geben, am ERPEL-Netzwerk teilzunehmen. Die Idee dahinter war, dass KMUs in den meisten Fällen bereits E-Mail Technologie im Einsatz haben. Für den Einsatz von konventionellen Web Services über HTTP(S) ist im Normalfall eine eigene Infrastruktur notwendig, was für viele kleine Betriebe mit hohen Kosten und dem Aufbau von neuem Know-How verbunden wäre.

Der verlässliche Datenaustausch wurde in diesem Protokoll nicht über WSRM (Web Service Reliable Messaging) gelöst, da es hier softwareseitig zu große Interoperabilitätsprobleme gibt, sondern über Business Signals (Bestätigungsnachrichten auf Applikationsebene). Sobald ein

Dokument an einen Empfänger zugestellt wurde, und wenn dieser das Dokument abholt, wird dem Sender eine automatisch generierte Nachricht in seine *MessageBox* gelegt, die er zu jedem Zeitpunkt abrufen kann. Um eine Veränderung der Daten bei der Übertragung zu erkennen, werden die Dokumente elektronisch vom Sender signiert. Dadurch kann der Empfänger, wenn er in Besitz des Public Keys des Senders ist, die Signatur überprüfen. Es werden immer nur die Geschäftsdokumente signiert – auf Protokollebene gibt es keine elektronische Signatur. Die Archivierung der Dokumente erfolgt ebenfalls zentral über den Messaging Hub. Dieser legt die Dokumente derzeit auf einer Amazon S3 Instanz ab – für die Zukunft wird derzeit die Verwendung von Amazon Glacier evaluiert, einer speziellen Lösung für Langzeitarchivierung.

## 3.6 Authentifizierung

Zur Authentifizierung von Teilnehmern im ERPEL-Netzwerk gibt es ein komplexes System aus *AppKey*, *UserKey* und *UserId*. Ein *AppKey* wird pro teilnehmendem Hersteller vergeben, um eventuellen Software-Inkompatibilitäten zentral begegnen zu können. Die *UserId* entspricht der E-Mail-Adresse des Benutzers und ist dessen Login-Name für das ERPEL-Netzwerk. Der *UserKey* ist eine pro *UserId* automatisiert erzeugte Kennung für den Datenaustausch über Web Services. Pro *UserKey* gibt es genau eine *UserId*.

Um Dokumente über den Messaging Hub austauschen zu können (*Deliver* und *Fetch*), müssen *AppKey*, *UserKey* und *UserId* mitgegeben werden. Für die Dokumentenvalidierung muss nur der *AppKey* mitgegeben werden (der *UserKey* wird derzeit noch nicht ausgewertet). Für den lesenden Zugriff auf die ERPEL Registry reicht die Angabe des *AppKey*, während die schreibenden Zugriffe *AppKey*, *UserKey* und *UserId* benötigen. In allen Fällen in denen *UserKey* und *UserId* notwendig sind, müssen diese per HTTP-BasicAuth im Header gesendet werden (Liegl et al. 2013).

# 4 PEPPOL

Bei PEPPOL (Pan-European Public Procurement On-Line) handelt es sich um ein europäisches Projekt dessen Ziel es ist, einen einheitlichen Austausch von elektronischen Beschaffungsdokumenten für Regierungsstellen in Europa umzusetzen. PEPPOL ist ein Teil des Rahmenprogramms für Wettbewerbsfähigkeit und Innovation (CIP) der Europäischen Kommission und inkludiert Projektpartner aus 11 europäischen Ländern, darunter auch Österreich. 2008 gestartet hatte es eine initiale Laufzeit bis April 2012, wurde aber bis August 2012 verlängert. Die offizielle Projektwebseite befindet sich unter http://www.peppol.eu - dort befinden sich nach dem Ende des Projekts immer noch viele Dokumente und Informationen. Österreich wurde repräsentiert durch die ARGE „PEPPOL.AT“ mit den Mitgliedern

- Bundesministerium für Finanzen (BMF)
- Bundesrechenzentrum (BRZ)
- Bundesbeschaffungsgesellschaft (BBG)

Das Projekt hatte den klaren Fokus, einen Austausch von E-Procurement-Dokumenten mit Regierungsstellen umzusetzen. Die vom Projekt entwickelte Software-Infrastruktur (mit dem Namen „BUSDOX“ – Details siehe Kapitel 4.4) ist jedoch in keinem Bereich auf Regierungsstellen eingeschränkt, sodass auch ein Dokumentenaustausch zwischen Nicht-Regierungsstellen kein Hindernis darstellt. Im Kontext von PEPPOL werden Kunden als „Contracting Authorities“ (CA) und Lieferanten als „Economic Operators“ (EO) bezeichnet. Im Gegensatz dazu sind „Service Provider“ (SP) Firmen, die Dienstleistungen im Kontext des elektronischen Dokumentenaustauschs durchführen.

Das Ziel von PEPPOL war es existierende Software-Komponenten miteinander zu verbinden, und nicht eine komplett neue Infrastruktur, die bestehende Lösungen ablösen soll, zu erstellen. D.h. existierende Service Provider sollen dazu bewegt werden dem PEPPOL Netzwerk beizutreten, da dadurch eine große Anzahl von potentiellen PEPPOL-Teilnehmern im Netzwerk verfügbar sind ohne dass jeder einzelne Teilnehmer aktiv beitreten muss. Mit jedem Service Provider, der dem PEPPOL Netzwerk beitritt, kommen also viele potentielle Teilnehmer zum Netzwerk hinzu.

PEPPOL ist nach dem Ende des Projekts in dem Non-Profit-Verein „OpenPEPPOL“ mit Sitz in Brüssel aufgegangen. Die offizielle Website des Vereins ist nach wie vor http://www.peppol.eu. Der Beitritt zu OpenPEPPOL ist kostenpflichtig. Die Höhe des Mitgliedsbeitrags richtet sich nach den Services die man in Anspruch nehmen möchte. Die Grundvoraussetzung für den Beitritt zu OpenPEPPOL ist die Übermittlung der unterzeichneten Transport Infrastructure Agreements (TIA). Diese enthalten die einzuhaltenden Regeln und Vorschriften für die Verwendung der PEPPOL Transport Infrastruktur.

## 4.1 Struktur

PEPPOL deckt einen großen Teil des Beschaffungsprozesses ab, aber nicht alle Elemente davon. Die folgende Abbildung zeigt die von PEPPOL unterstützten Prozessteile gelb hervorgehoben: „eAttestation (VCD)“, „eCatalogue“, „eSignature“, „PEPPOL Transport Infrastructure“, „eOrdering“ und „eInvoicing“:

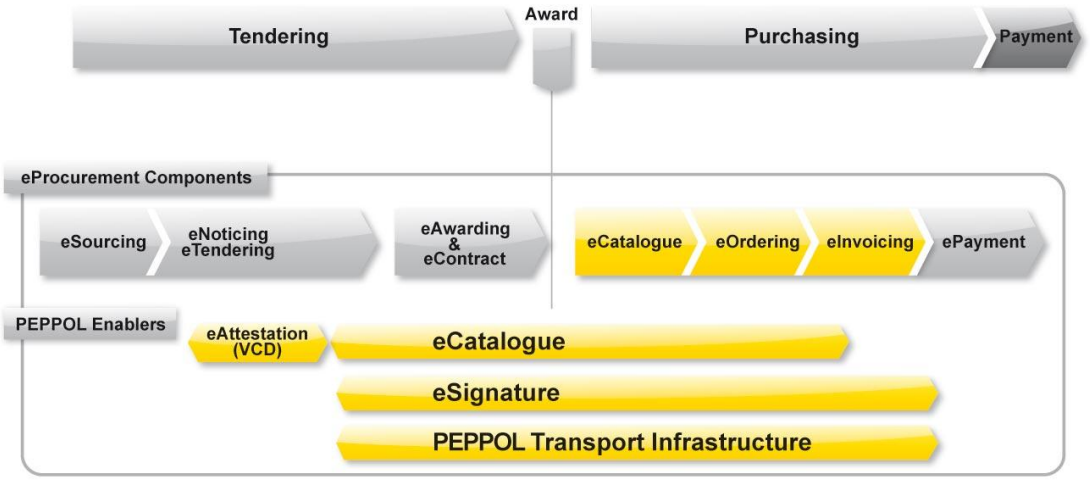


Abbildung 12: PEPPOL E-Procurement Elemente (PEPPOL 2012)

Die linke Seite der Grafik zeigt „Pre-Award“ (Vor Zuschlag) Komponenten, in der Mitte ist der „Award“ (Zuschlag) und rechts davon befinden sich die „Post-Award“ (Nach Zuschlag) Komponenten. Einige der Komponenten sind auch übergreifend über beiden Phasen hinweg einsetzbar. Ein Überblick über die einzelnen Komponenten befindet sich in den Kapiteln 4.2 Pre-Award (Vor Zuschlag), 4.3 Post-Award (Nach Zuschlag) und 4.4 Transport Infrastruktur.

Ursprünglich war PEPPOL in 8 Arbeitspakete (sog. „Work Packages“, WP) unterteilt:

- WP 1: eSignature
- WP 2: Virtual Company Dossier (VCD)
- WP 3: eCatalogue
- WP 4: eOrdering
- WP 5: eInvoicing
- WP 6: Project management
- WP 7: Awareness, Training, Consensus Building
- WP 8: Transport Infrastructure

In den Arbeitspaketen WP 2, WP 3, WP 4, WP 5, WP 7 und WP 8 gab es eine aktive österreichische Beteiligung. Aus den anderen beiden Arbeitspaketen wurden nur die Ergebnisse bzw. Vorgaben übernommen. Anfang 2010 wurde aus Effizienzgründen beschlossen die Arbeitspakete WP 3, WP 4 und WP5 umzustrukturieren, und zu jeweils einem „Pre-Award“ (WP 3 neu) und zu einem „Post-Award“ (WP 4 neu) Arbeitspaket zusammenzufassen, wodurch die Gesamtanzahl der Arbeitspakete auf 7 reduziert wurde.

## 4.2 Pre-Award (Vor Zuschlag)

### 4.2.1 VCD

Die PEPPOL „Pre-Award“ Komponenten enthalten das Virtual Company Dossier (VCD) und den „Pre-Award Catalogue“. Das VCD System dient zur Definition und Abfrage der nationalen Kriterien für Ausschreibungen. Das ist sinnvoll, da es in jedem europäischen Land unterschiedliche Kriterien für Ausschreibungen gibt, und damit die Barrieren für die Teilnahme an grenzüberschreitenden Ausschreibungen gesenkt werden sollen. Es erlaubt einem Anbieter

bei einer Ausschreibung die automatische Sammlung und Abholung der gesetzlichen notwendigen Nachweise über eine zentrale Plattform. Falls bei einzelnen Dokumenten Kosten anfallen (z.B. für einen Strafregisterauszug), so sind diese trotzdem zu entrichten. Die Kriterien werden über Ontologien modelliert, und müssen für jedes Land separat gepflegt werden.

Das VCD System bieten dabei Entscheidungsunterstützung durch einen Vorschlag der benötigten Nachweise und teilweise die automatisierte Abholung aus den jeweiligen Systemen (z.B. Firmenbuch). Die endgültige Festlegung der eingebrachten Dokumente obliegt aber dem Ausschreibungs-Teilnehmer.

### 4.2.2 Pre-Award Katalog

Im Vergleich zum VCD ist der elektronische Pre-Award Katalog ein direkter Bestandteil der übermittelten Ausschreibungsunterlagen. Ein E-Katalog kann sowohl vor dem Zuschlag als auch nach dem Zuschlag übermittelt werden, wobei sich aber die Inhalte (der Detailgrad) des Katalogs unterscheiden können. Der Pre-Award Katalog kann als Basis für den Post-Award Katalog betrachtet werden, und entfaltet bereits rechtliche Bindungskraft (Stefano et al. 2012).

Die folgende Grafik soll die Verwendung von E-Katalogen im PEPPOL Kontext verdeutlichen:

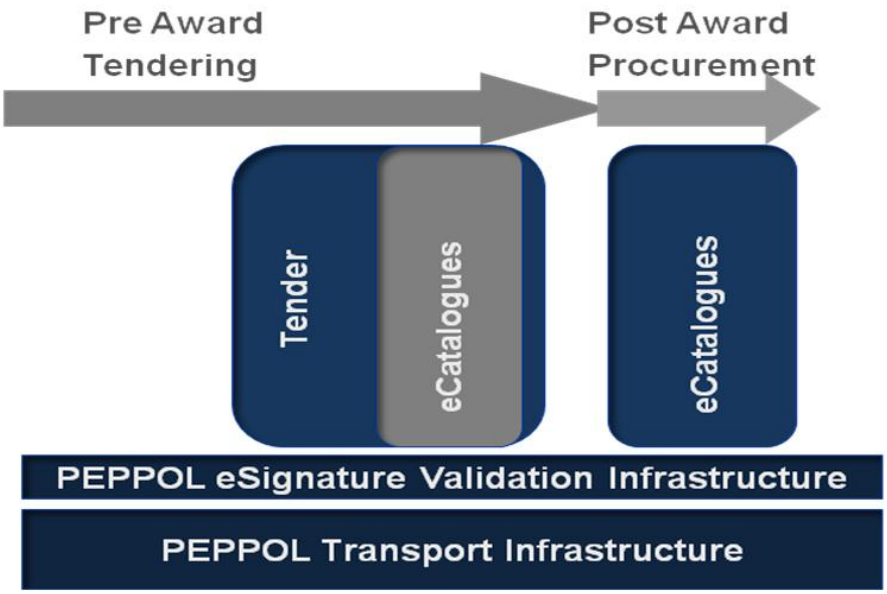


Abbildung 13: E-Kataloge in PEPPOL (Stefano et al. 2012)

## 4.3 Post-Award (Nach Zuschlag)

Die Post-Award Komponenten von PEPPOL umfassen die Prozesse, die nach dem Zuschlag erfolgen. Von PEPPOL unterstützt werden der E-Katalog, die E-Bestellung und die E-Rechnung. Für die Zahlung bzw. den Leistungsnachweis wird von PEPPOL keine Unterstützung angeboten. Der Datenaustausch von Dokumenten basiert auf standardisierten Prozessen, die beim CEN ISSS WS/BII Workshop (CENBII) erarbeitet wurden (nach Beendigung der ersten Phase des CEN ISSS WS/BII wurde 2010 Phase 2 gestartet, die auch heute, im September 2012, noch aktiv ist). CEN ist die europäische Normungskommission, deren Normen in Österreich durch die Austrian Standards verpflichtend umgesetzt werden müssen. Dieser CEN Workshop hat einen Satz von Standardprozessen im Beschaffungsumfeld technologieunabhängig definiert, die partiell von PEPPOL aufgegriffen und mit einem Syntax-Mapping versehen wurden. Diese Prozesse - im

CENBII Jargon "Profiles" genannt - enthalten neben der Chorographie auch eine Definition von unterstützten Datenfeldern und deren Semantik. Eine Choreographie besteht aus einzelnen Kollaborationen, welche wiederum eine wiederverwendbare Aggregation von Transaktionen ist, die für die Übermittlung von gleichstrukturierten Dokumenten zwischen Geschäftspartnern steht.

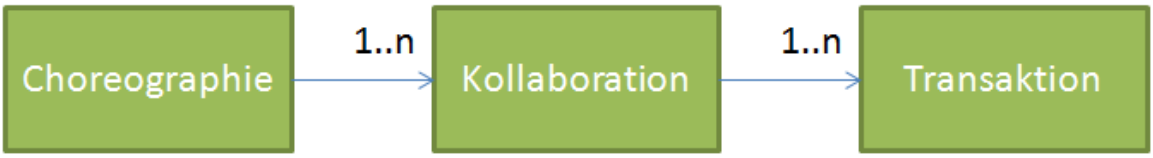


Abbildung 14: Aufbau einer CENBII Choreographie

Für PEPPOL mussten einige Prozessdetails angepasst werden, um sie praxistauglich zu machen. Die so entstandenen PEPPOL Standardprozesse wurden in "PEPPOL Business Interoperability Specifications" zusammengefasst und publiziert. PEPPOL verwendet für alle Dokumente ein UBL-Mapping. UBL steht für Universal Business Language und ist ein von der OASIS standardisierter XML-Dialekt zum Austausch von elektronischen Beschaffungsdokumenten. Die Projektwebseite befindet sich unter https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ubl (abgerufen am 29.9.2012).

### 4.3.1 Business Interoperability Specifications (PEPPOL BIS)

PEPPOL beschreibt jeden Prozess mit einer "Business Interoperability Specification", oder kurz PEPPOL BIS. Die Inhalte einer PEPPOL BIS sind nach (Borresen und Pedersen 2010):

- der rechtliche Rahmen der Spezifikation
- der organisatorische Rahmen der Spezifikation
- die Choreographie der abgedeckten Geschäftsprozesse
- die innerhalb der Geschäftsprozessen durchzuführenden elektronischen Transaktionen (Übermittlung von Dokumenten)
- die Geschäftsregeln die bei der Ausführung der Geschäftsprozesse eingehalten werden müssen
- die Inhalte der ausgetauschten Dokumente
- die technische Umsetzung der Spezifikation inkl. einer semantischen Spezifikation
- das Zusammenspiel der Spezifikation mit der Transport Infrastruktur und der E-Signatur

Jedes PEPPOL BIS ist um das "European Interoperability Framework" (EIF) 2.0 herum organisiert. Die Ziele des EIF sind:

- es soll als Basis für die nahtlose europäische Interoperabilität im Bereich der öffentlichen Zustellung dienen, und dadurch einen verbesserten öffentlichen Dienst auf europäischer Ebene bereitstellen
- die Bereitstellung von "Pan-European eGovernment Services" (PEGS) soll durch eine erweiterte, grenz- und sektorüberschreitende Interoperabilität unterstützt werden
- existierende nationale Interoperabilitäts-Frameworks sollen in einer paneuropäischen Dimension unterstützt werden

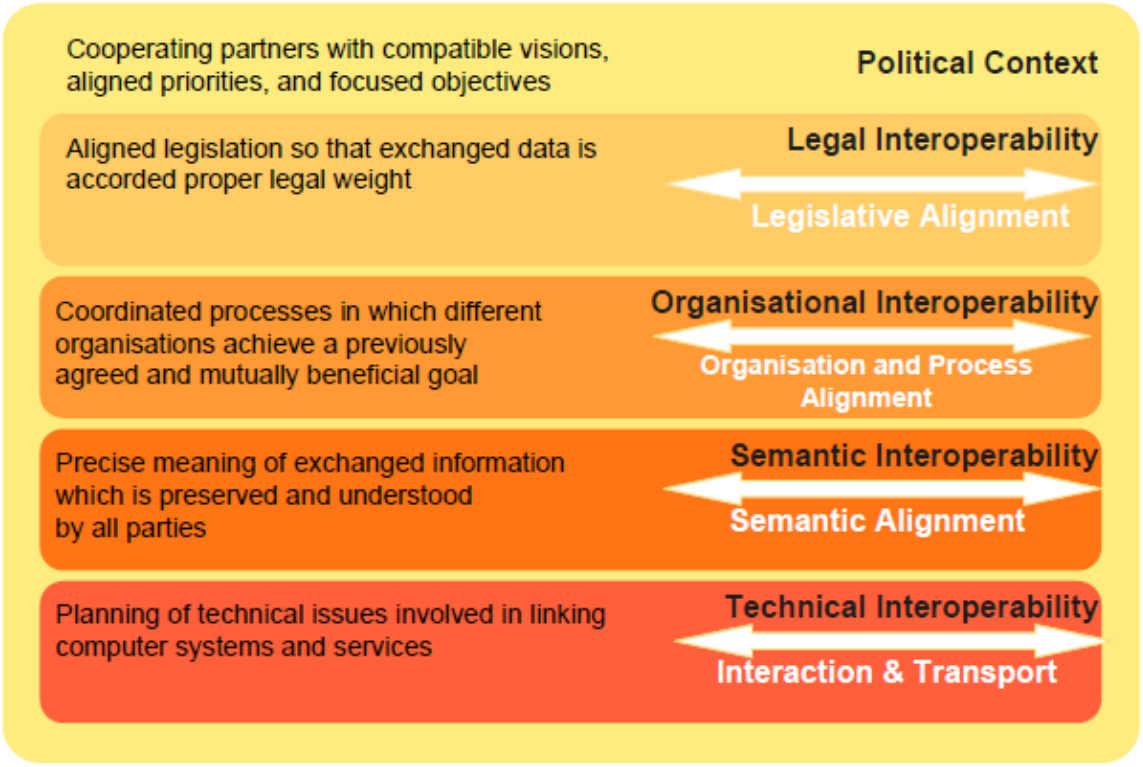


Abbildung 15: EIF 2.0 Interoperabilitätsebenen (EUROPEAN COMMISSION 2010)

Das EIF-Modell ist, wie dargestellt, in vier Ebenen unterteilt. Für PEPPOL war diese Einteilung zu grobgranular weshalb die EIF "Organisational Interoperability"-Ebene in eine „organisatorische Interoperabilitätsbene“ und eine „Prozessinteroperabilitätsebene“ untergliedert wurde. Außerdem wurde die EIF "Technical Interoperability"-Ebene in eine „Prozess- und Semantikebene“, eine „eSignature-Validierungsebene“ und eine „Transportebene“ untergliedert. Dadurch stehen in PEPPOL in Summe sieben Ebenen zur Verfügung.

Folgende PEPPOL BISs wurden erarbeitet:

- **BIS 1a - Catalogue only**: Ein Lieferant sendet einen elektronischen Katalog an einen Käufer. Dieser muss den Katalog entweder akzeptieren oder ablehnen. Das entspricht der CENBII Kollaboration "BiiColl09" (Stefano und Bertini 2012).
- **BIS 3a - Basic Order only**: Ein Käufer übermittelt eine elektronische Bestellung an einen Lieferanten. Es wird keine Bestätigung oder Ablehnung des Lieferanten auf elektronischem Wege unterstützt. Das entspricht der CENBII Kollaboration "BiiColl01" (Skulason et al. 2012).
- **BIS 4a - Basic Invoice only**: Ein Lieferant übermittelt eine elektronische Rechnung an einen Käufer. Es wird keine Bestätigung oder Ablehnung des Käufers auf elektronischem Wege unterstützt. Das entspricht der CENBII Kollaboration "BiiColl04" (Skulason und Pedersen 2012a).
- **BIS 5a - Billing**: Ein Lieferant übermittelt eine elektronische Rechnung an einen Käufer (BiiColl04). Falls die Rechnung vom Käufer nicht akzeptiert wird (diese Mitteilung erfolgt nicht im Prozess sondern "extern"- z.B. via Telefon), so hat der Lieferant die Möglichkeit eine Korrektur-Rechnung bzw. eine Gutschrift elektronisch zu übermitteln (BiiColl05)

(Skulason und Pedersen 2012b).
Der Name "Billing" ist leider irreführend, da keine Zahlung im Prozess involviert ist. Es handelt sich eigentlich um einen "Invoice only" Prozess, der etwas komplexer ist als der in BIS 4a Definierte.

- **BIS 6a - Procurement**: Ein Käufer übermittelt eine elektronische Bestellung an einen Lieferanten (BiiColl01), die dieser entweder akzeptieren oder ablehnen muss (BiiColl03). Falls die Bestellung akzeptiert wurde, so kann zu einem späteren Zeitpunkt vom Lieferanten an den Käufer eine elektronische Rechnung übermittelt werden (BiiColl04), bzw. im Fehlerfall auch eine Korrektur-Rechnung oder eine Gutschrift (BiiColl05) (Skulason und Pedersen 2012c).
- **BIS 12a – eCatalogues as part of a Tender**: Im Zuge einer Ausschreibung übermittelt der Lieferant dem Käufer seine Unterlagen für die Ausschreibung, in der auch der elektronische Katalog enthalten ist. Der Käufer bestätigt elektronisch dass er die Ausschreibungsunterlagen erhalten hat (BiiColl022) (Stefano et al. 2012).

Das kleine "a" im Namen einer BIS steht für "UBL Mapping". Falls es zu einem späteren Zeitpunkt ein Mapping auf ein anderes Datenformat als UBL geben sollte (z.B. UN/CEFACT), so wird nur dieser Buchstabe geändert, da die rechtlichen und organisatorischen Regeln nicht geändert werden dürfen.

Die Kollaborationen in der obigen Liste dienen zur Verdeutlichung des verwendeten Konzepts von CENBII. Die Kollaboration "BiiColl04" (Invoice only) wird in BIS 4a, BIS 5a und BIS 6a verwendet, und besteht aus einer einzigen Transaktion "SubmitInvoice". D.h. an allen Stellen, an denen diese Kollaboration verwendet wird, muss genau ein Dokument mit der stets gleichen Semantik und denselben Regeln ausgetauscht werden. Die Kollaboration "BiiColl09" besteht je nach Ablauf aus zwei unterschiedlichen Transaktionen: "SubmitCatalogue" (Katalog übermitteln) ist immer die erste Transaktion und als Antwort muss entweder die Transaktion "RejectCatalogue" (Katalog ablehnen) oder die Transaktion "AcceptCatalogue" (Katalog akzeptieren) durchgeführt werden (Stefano und Bertini 2012).

## 4.4 Transport Infrastruktur

Die Transport Infrastruktur ist eine zentrale Komponente von PEPPOL und wurde im WP 8 entwickelt. Sie definiert die Protokolle, Schnittstellen und Datenformate für den sicheren Datenaustausch zwischen den Teilnehmern. Die Infrastruktur trägt den Namen BUSDOX (Business Document Exchange) und wurde im Rahmen des PEPPOL Projekts entwickelt. Um Sicherheit für die langfristige Pflege der Spezifikationen zu gewährleisten wurden diese an OASIS übergeben. Dafür wurde das „OASIS BUSDOX Technical Committee“ (BDX TC) gegründet, welches jedoch durch eine geänderte Charta zum BDXR TC (das „R“ steht für „revised“) wurde. Die offizielle Projektwebseite ist unter https://www.oasis-open.org/committees/bdxr/ zu finden (abgerufen am 17.09.2012). Ziel dieses TCs ist die Wartung und Weiterentwicklung der technischen Infrastruktur zur Übermittlung von elektronischen Dokumenten im Bereich E-Procurement. Als Merkmale werden das „4-corner model“, sowie die sichere und zuverlässige Übertragung von elektronischen Dokumenten genannt. (OASIS 2012)Die Architekturprinzipien von BUSDOX sind laut (Pedersen und Andersen 2011):

- Verbinde existierende Systeme, und ersetze sie nicht

- Ermögliche einen sicheren und verlässlichen Dokumentenaustausch
- Baue auf die Erfahrungen und Erkenntnisse existierender Applikationen
- Verwende ausschließlich offene, frei verfügbare Technologien
- Verwende nur Internet-Technologien
- Benutze die CEN ISSS WS/BII (1+2) CWAs als Basis für die Umsetzung
- Versuche existierende Infrastruktur möglichst effizient zu nutzen
- Sei föderiert und skalierbar.

BUSDOX selbst basiert vollständig auf anderen offenen Standards und setzt keine proprietären Technologien voraus. Der Dokumentenaustausch von BUSDOX ist im Gegensatz zu PEPPOL technologieneutral spezifiziert und kann daher mit beliebigen Datenformaten umgehen. PEPPOL spezifiziert im Gegensatz dazu, dass nur XML-Dokumente über die Transport Infrastruktur ausgetauscht werden können.

Die PEPPOL Transport Infrastruktur besteht sowohl aus zentralen Komponenten (Public Key Infrastructure (PKI) und dem Service Metadata Locator (SML)), dezentralen Komponenten (Service Metadata Publisher (SMP) und Access Point (AP)) und Web Service Profilen (Secure Trusted Asynchronous Reliable Transport (START) und Light Weight Message Profile (LIME)).

Die zentralen Komponenten existieren genau einmal in der PEPPOL Infrastruktur im Gegensatz zu den dezentralen Komponenten, die je angebundenem Service Provider existieren (wobei es theoretisch sein kann, dass sich mehrere Service Provider eine SMP Instanz teilen). Zusätzlich gibt es noch einige Hilfskomponenten die eingesetzt werden können, für die es jedoch keine Verpflichtung zum Einsatz gibt: die Validierung von Dokumenten sowie die Visualisierung/Darstellung von Dokumenten. Zu allen aufgezählten Komponenten gibt es frei verfügbare Open Source Lösungen in Java und teilweise auch in .NET.

### 4.4.1 Das „4-corner model“

Das „4-corner model“ bezeichnet den Prozess zum Datenaustausch zwischen zwei bis vier Parteien. Dabei können sowohl Sender als auch Empfänger jeweils einen Service Provider ihrer Wahl verwenden, der für die technische Übertragung verantwortlich ist. Die 4 Ecken sind daher: Sender, Service Provider des Senders, Service Provider des Empfängers und der Empfänger. Hat entweder nur der Sender oder nur der Empfänger einen Service Provider, so spricht man von einem „3-corner model“ und wenn weder Sender noch Empfänger einen Service Provider nutzen gar nur von einem „2-corner model“.

Das „4-corner model“ ist keine Erfindung von PEPPOL, sondern eine in der Branche übliche Bezeichnung für diese Topologie des Datenaustauschs. Daher ist dieses Modell auch nicht spezifisch für PEPPOL und wird von anderen Datenaustauschnetzwerken ebenfalls verwendet.

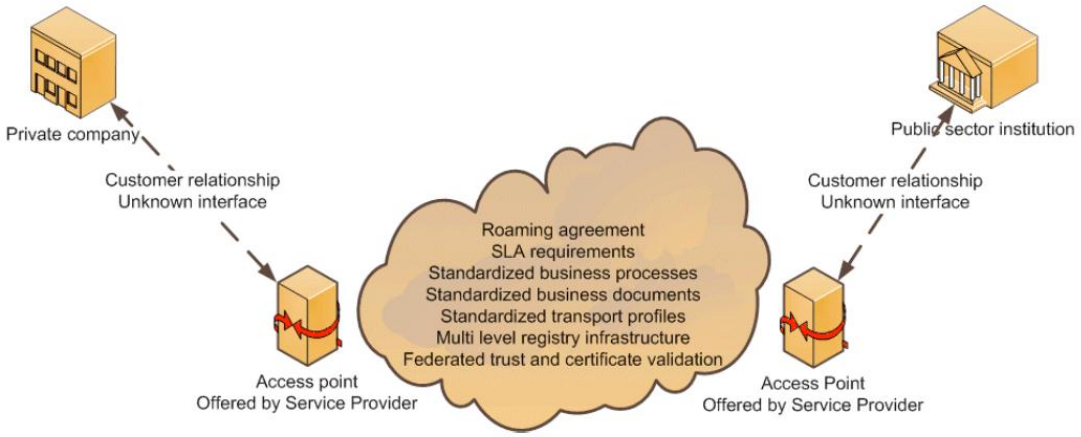


Abbildung 16: Das 4-corner model (OASIS 2011)

In Abbildung 16 wird das vollständige „4-corner model“ dargestellt. Es wird gezeigt, dass der Dokumentenaustausch bidirektional ist, d.h. beide Endpunkte („Private company“ und „Public sector institution“) können Dokumente senden und empfangen. Die Wolke in der Mitte ist im Fall von PEPPOL die Transport Infrastruktur wobei ersichtlich ist, dass neben den technischen Komponenten auch organisatorische Aspekte wie etwa SLAs (Service Level Agreements), Standardisierung (für Prozesse und Dokumente) und bilaterale Absprachen wichtig sind.

Die beiden Service Provider sind als „Access Point“ deklariert, wohingegen die Endpunkte dies nicht sind. Das kommt daher, dass nur die beiden Service Provider jeweils einen PEPPOL Access Point implementieren müssen. Es wird davon ausgegangen, dass es bereits eine existierende Geschäftsbeziehung eines Endpunkts mit (mindestens) einem Service Provider gibt.

Theoretisch kann es sich bei den beiden dargestellten Service Providern auch um ein und denselben handeln, wodurch die Wolke in der Mitte wegfallen würde und es ein „3-corner model“ wäre. Zur Verdeutlichung des Modells wird jedoch davon ausgegangen, dass es sich um zwei verschiedene Service Provider handelt.

### 4.4.2 Identifikation

Ein wichtiges Element von PEPPOL ist die eindeutige Referenz von Elementen über „Identifier“ (ID). Im Vergleich zu anderen Projekten war es von vornherein das Ziel keine neuen Identifikationen zu vergeben, sondern existierende so gut es geht wiederzuverwenden. Da es aber keine einzelne existierende Liste von IDs gab, die alle Anforderungen erfüllt, wurde ein generisches, zusammengesetztes Format für die Definition von Identifikationen entwickelt. Es besteht aus einem Schema und einem Wert. Das verwendete Schema definiert die Syntax für den Wert. PEPPOL bietet 3 Schemata für verschiedene Typen von IDs an:

- `iso6523-actorid-upis` für Teilnehmer
- `busdox-docid-qns` für Dokumententypen und
- `cenbii-procid-ubl` für Prozesse

Werden PEPPOL-IDs gespeichert, bzw. wird eine textuelle Repräsentation benötigt, so ist diese für alle Schemata gleich: Schema und Wert werden durch zwei Doppelpunkte voneinander getrennt.

```
<Schema>::<Wert>
```

Die Trennung in Schema und Wert ist notwendig, damit BUSDOX generisch bleibt, und keine PEPPOL Spezifika abbildet (Helger und Rasmussen 2012).

#### 4.4.2.1 Teilnehmer Identifikation

Das Schema für Teilnehmer-IDs in PEPPOL ist immer `iso6523-actorid-upis`. Dieses Schema definiert den Wert einer ID als Zusammensetzung eines externen ID-Schemas und der eigentlichen ID, d.h. der Wert einer PEPPOL ID ist ebenfalls ein komplexer Wert. In diesem Zusammenhang kann das globale Schema `iso6523-actorid-upis` auch als „Meta-Schema" bezeichnet werden. Der Wert beginnt mit „iso6523" um darauf hinzuweisen, dass die Werte an die von ISO-6523 definierte Liste angelehnt sind. Eine inoffizielle Liste aller ISO-6523 Werte kann unter http://www.oid-info.com/doc/ICD-list.pdf (abgerufen am 17.9.2012) gefunden werden, jedoch werden nicht alle diese Werte von PEPPOL unterstützt. Die offizielle ISO-6523 Webseite befindet sich unter http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=25773 (abgerufen am 17.9.2012). „actorid" bezeichnet die ID für einen Teilnehmer und „upis" steht für „Universal Participant Identifier Scheme" und soll die Universalität ausdrücken.

PEPPOL stellt eine Liste von unterstützten Schemata bereit, die verwendet werden können. In der derzeit gültigen Liste werden 60 verschiedene ID-Schemata unterstützt. Darunter sind sowohl internationale Schemata wie z.B. DUNS (Dun and Bradstreet) und GLN (Global Location Number), als auch nationale Schemata wie z.B. die österreichische UID Nummer. Jedem dieser Schemata ist von PEPPOL eine eindeutige Nummer zugeordnet. So steht etwa „0088" für GLN und „9914" für die österreichische UID Nummer. Die vollständige Liste der Zuordnungen steht als XML-Datei und als Genericode-Codeliste (Genericode ist ein OASIS Standard zur Definition von Codelisten in XML) unter https://joinup.ec.europa.eu/svn/peppol/dev/public/codelists/infrastructure/identifier/latest (abgerufen am 17.9.2012) zur Verfügung. In der textuellen Repräsentation wird die Nummer der verwendeten Liste von der ID durch einen einzelnen Doppelpunkt getrennt.

Beispiel: die österreichische UID Nummer „ATU12345678" hat als Teil einer PEPPOL-ID den Wert „9914:ATU12345678". Gemeinsam mit dem generischen Schema `iso6523-actorid-upis` ergibt sich daraus folgende vollständige PEPPOL Teilnehmer-ID:

```
iso6523-actorid-upis::9914:ATU12345678
```

Um im PEPPOL Netzwerk Dokumente austauschen zu können, muss aber nur der zweite Teil der ID (9914:ATU12345678) bekannt gegeben werden, da der erste Teil (`iso6523-actorid-upis`), wie bereits erwähnt, immer konstant ist.

Wichtig ist, dass Teilnehmer-IDs in UBL Dokumenten unterschiedlich zu handhaben sind als bei Verwendung in der PEPPOL Infrastruktur. Kurz gesagt darf in UBL nicht die PEPPOL Nummer für die referenzierte Liste verwendet werden, sondern es muss die textuelle Schema-ID (z.B.

„AT:VAT“ für die österreichische UID-Nummer) verwendet werden, die in den bereitgestellten Codelisten zu finden ist (Helger und Rasmussen 2012).

#### 4.4.2.2 Dokumententypen Identifikation

Das Schema für die Identifikation von Dokumententypen (z.B. Bestellung oder Rechnung) in PEPPOL ist immer `busdox-docid-qns`[1]. Der Wert einer Dokumententypen-ID setzt sich aus der Namespace URI, dem lokalen Namen des Wurzelelements des übertragenen Dokumententyps und einer „Sub Type ID“ zusammen. Da in PEPPOL nur XML-Dokumente übertragen werden, ist dieser Ansatz gültig, wenn auch nicht allgemeingültig, da er fast nur auf XML-Dokumente anwendbar ist. In PEPPOL ist die Sub Type ID spezifiziert als Aggregation einer „Transaction ID“, einer optionalen Liste von „Extension IDs“ und einer Versionsnummer.

Eine Rechnung im UBL Format hat den Namespace URI „urn:oasis:names:specification:ubl:schema:xsd:Invoice-2“ und den lokalen Wurzelelementnamen „Invoice“. Gemeinsam mit der Transaction ID „urn:www.cenbii.eu:transaction:biicoretrdm010:ver1.0“, der Extension ID „urn:www.peppol.eu:bis:peppol4a:ver1.0“ und der Versionsnummer „2.0“ ergibt sich folgender Wert der ID (als eine lange Zeile, ohne Leerzeichen und Zeilenumbrüche):

```
urn:oasis:names:specification:ubl:schema:xsd:Invoice-
2::Invoice##urn:www.cenbii.eu:transaction:biicoretrdm010:ver1.0:#
urn:www.peppol.eu:bis:peppol4a:ver1.0::2.0
```

Alles nach dem Trennzeichen „##“, definiert in welcher Business-Transaktion und in welchem Prozess diese ID zum Einsatz kommt. Kombiniert mit dem PEPPOL Schema für Dokumententypen entsteht folgende vollständige PEPPOL Dokumententyp-ID:

```
busdox-docid-
qns::urn:oasis:names:specification:ubl:schema:xsd:Invoice-
2::Invoice##urn:www.cenbii.eu:transaction:biicoretrdm010:ver1.0:#
urn:www.peppol.eu:bis:peppol4a:ver1.0::2.0
```

Die vollständige Liste der unterstützten Dokumententypen steht ebenfalls als XML-Datei und als Genericode-Codeliste unter der URL https://joinup.ec.europa.eu/svn/peppol/dev/public/codelists/infrastructure/identifier/latest (abgerufen am 17.9.2012) zur Verfügung.

#### 4.4.2.3 Prozess Identifikation

Das Schema für die Identifikation von Prozessen in PEPPOL ist immer `cenbii-procid-ubl`[2]. „cenbii“ steht dabei für „CEN/ISSS Business Interoperability Interfaces for Public procurement in Europe”, welche die in PEPPOL verwendeten Prozesse allgemein und syntaxneutral definiert haben. PEPPOL hat diese Prozesse als Basis verwendet und mit spezifischen Regeln und einem Mapping auf UBL erweitert (vgl. mit Kapitel 4.3.1). PEPPOL verwendet die CENBII „Profile IDs“ als Werte für die Prozess-Identifikation.

Beispiel: Für den Prozess „Invoice only“ (nur die Rechnung) wird folgender ID-Wert in PEPPOL verwendet:

[1] „docid“ steht für „document type ID“ und „qns“ steht für „qualified namespace“.
[2] „procid“ steht für „process ID“ und „ubl“ definiert das Syntax-Mapping auf UBL.

```
urn:www.cenbii.eu:profile:bii04:ver1.0
```

Kombiniert mit dem PEPPOL Schema für Prozesse entsteht folgende vollständige PEPPOL Prozess-ID:

```
cenbii-procid-ubl::urn:www.cenbii.eu:profile:bii04:ver1.0
```

Die vollständige Liste der unterstützten Prozesse steht als XML-Datei und als Genericode-Codeliste unter https://joinup.ec.europa.eu/svn/peppol/dev/public/codelists/infrastructure/identifier/latest (abgerufen am 17.9.2012) zur Verfügung. Darin enthalten ist neben der CENBII „Profile ID“ auch noch die ID des verwendeten PEPPOL BIS, sowie die Liste aller in diesem Prozess verwendbaren PEPPOL Dokumententypen.

### 4.4.3 Public Key Infrastructure (PKI)

In PEPPOL werden Zertifikate an allen wesentlichen Stellen eingesetzt. Sowohl beim Schreiben von DNS Einträgen, beim Erzeugen von SMP Antworten als auch beim Dokumentenaustausch werden digitale Zertifikate verwendet. Daher wurde eine eigene PEPPOL PKI (Public Key Infrastructure) definiert, auf der alle ausgegeben Zertifikate für die aktiven PEPPOL-Teilnehmer beruhen. Die folgende Abbildung zeigt die PEPPOL Zertifikatshierarchie:

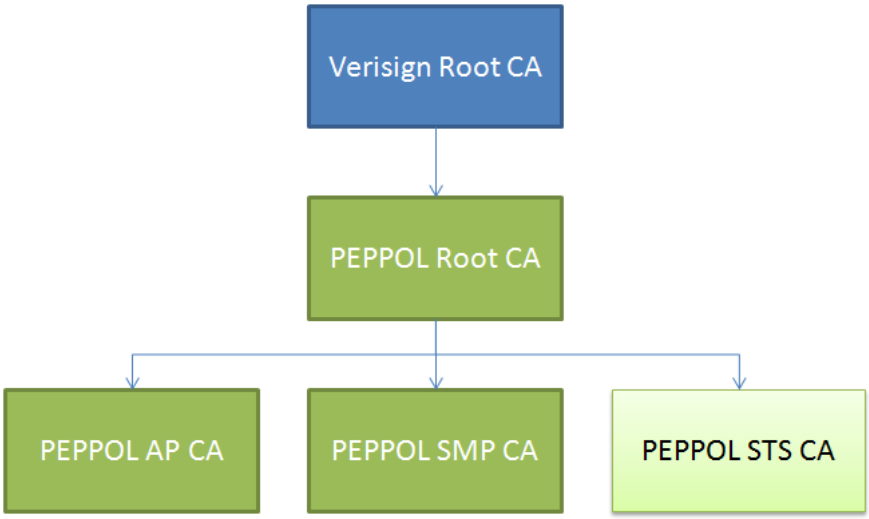


Abbildung 17: Struktur der PEPPOL PKI

Das PEPPOL Root-Zertifikat basiert auf einem Verisign Zertifikat, also einem anerkannten, und in den meisten Betriebssystemen vorinstallierten, Root-Zertifikat. Es gibt drei untergeordnete Zertifikate: Für AccessPoints („PEPPOL AP CA“), für SMPs („PEPPOL SMP CA“) und eines für den Secure Token Service („PEPPOL STS CA“). Verwendet werden aber nur das „PEPPOL AP CA“ und das „PEPPOL SMP CA“ Zertifikat. Die Verwendung des STS Zertifikats war ursprünglich geplant, wurde aber nie umgesetzt.

Jeder PEPPOL Teilnehmer der einen SMP betreiben möchte, bekommt ein Zertifikat, das auf dem „PEPPOL SMP CA“ Zertifikat basiert, und Teilnehmer, die einen AccessPoint betreiben wollen, bekommen ein auf dem „PEPPOL AP CA“ Zertifikat basierendes Zertifikat. Die öffentlichen Schlüssel (Public keys) der PEPPOL Root-Zertifikate stehen als globaler Truststore öffentlich zur

Verfügung. Details über die Vergabe von Zertifikaten und weitere Details finden sich in (Gundel und Pedersen 2011).

### 4.4.4 Service Metadata Locator (SML)

Der Service Metadata Locator (SML) ist eine zentrale Komponente in der PEPPOL Infrastruktur. Die Rolle des SML ist es, DNS Einträge zu erstellen, die später beim eigentlichen Dokumentenaustausch verwendet werden. Der SML wird also nur benötigt, wenn ein Teilnehmer oder ein SMP dem PEPPOL-Netzwerk beitritt bzw. dieses verlässt. Beim Datenaustausch ist der SML nicht involviert. Der SML bietet zwei SOAP basierte Web Services an, mit denen die Daten verändert werden können: `ManageServiceMetadata` zur Verwaltung von SMPs und `ManageParticipantIdentifier` zur Verwaltung von einzelnen Teilnehmern. Beide Web Services sind nur über HTTPS mit einem passenden PEPPOL Client-Zertifikat (basierend auf dem „PEPPOL SMP CA“) erreichbar (Sylvest et al. 2010b).

PEPPOL hat derzeit zwei SMLs im Einsatz: einen produktiven unter https://sml.peppolcentral.org und einen zum Testen unter https://smk.peppolcentral.org. Beide werden vom österreichischen Bundesrechenzentrum (BRZ) betrieben. Um auf die Webseiten per Browser oder Applikation zugreifen zu können, muss ein entsprechendes Client-Zertifikat vorliegen, sonst wird der Zugriff verweigert. Für beide Instanzen gibt es auch unterschiedliche Domain-Namen: das produktive System erzeugt Einträge in der Domain „sml.peppolcentral.org“, während das Testsystem Einträge in der Domain „smk.peppolcentral.org“ erzeugt.

#### *4.4.4.1 DNS Einträge für SMPs*

Wenn ein neuer SMP am PEPPOL Netzwerk teilnehmen möchte, so muss sich dieser einmalig beim SML registrieren. Für diese Registrierung sind 3 Elemente notwendig: die „SMP-ID“, die physische Adresse und die logische Adresse des SMP Servers. Im Zuge dieser Registrierung wird neben einem Datenbank-Eintrag auch ein DNS Eintrag – der sog. „Publisher-Eintrag“ – erstellt. Dieser Eintrag wird als Referenz für die Teilnehmer-DNS-Einträge verwendet, um ein einfaches Aktualisieren zu erlauben. Der Publisher-Eintrag ist entweder ein DNS A-Eintrag, d.h. ihm ist eine IPv4-Adresse zugewiesen, oder ein CNAME- Eintrag, der auf einen Namen in einer anderen Domain verweist.

Die Struktur des Namens eines „Publisher-Eintrags“ sieht wie folgt aus:

```
<SMP-ID>.publisher.<SML-Domain>
```

Daraus ergibt sich auch, dass die SMP-ID nur aus Zeichen bestehen darf, die innerhalb eines DNS-Namens gültig sind. Die SML-Domain ist der Name der Top-Level Domain, in der die Einträge gespeichert werden. Für die produktiven DNS-Einträge ist die Top-Level Domain „sml.peppolcentral.org“ und für Test-DNS-Einträge „smk.peppolcentral.org“.

Beispiel: Für die SMP-ID „TESTSMP“ auf dem produktiven SML wird folgender DNS-Name erzeugt:

```
TESTSMP.publisher.sml.peppolcentral.org.
```

Der Punkt am Ende des Namens ist im Rahmen der DNS Spezifikationen (RFC 1034 und 1035) gültig.

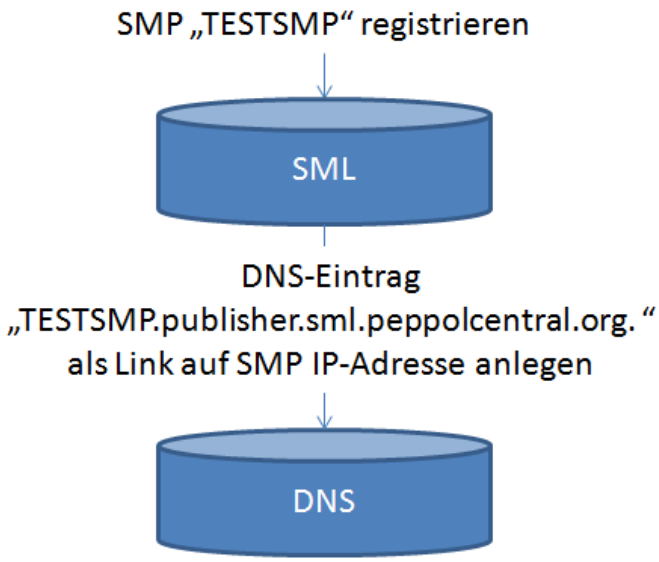


Abbildung 18: Ablauf der SMP-Registrierung

#### 4.4.4.2 DNS Einträge für Teilnehmer

Aus der eindeutigen PEPPOL Teilnehmer- ID wird ein eindeutiger DNS Name generiert, der folgendes Schema hat (ohne Protokoll):

```
<Hashwert der Empfänger-ID>.<Schema-ID>.<SML-Domain>
```

Der Hashwert der Empfänger-ID ist ein MD5-Hash über die PEPPOL Teilnehmer-ID in Kleinschreibung mit dem fixen Präfix „B-“. Die Kleinschreibung ist wichtig, damit immer derselbe Hash-Wert berechnet werden kann. Außerdem ist in PEPPOL explizit vorgegeben, dass die Kleinschreibung verwendet werden muss, selbst wenn das dahinterliegende ID Schema zwischen Klein- und Großschreibung unterscheidet. Die Schema-ID ist der Wert des verwendeten Teilnehmer-ID-Schemas und in PEPPOL immer „iso6523-actorid-upis“ (siehe Kapitel 4.4.2.1). Da das Teilnehmer-ID-Schema ein direkter Bestandteil des DNS-Namens ist, gibt es in BUSDOX spezifische Regeln, in Form eines regulären Ausdrucks, wie ein Teilnehmer-ID-Schema aussehen darf. Die SML-Domain ist der Name der Top-Level Domain, in der die Einträge gespeichert werden.

Beispiel: für die fiktive Teilnehmer-ID „0088:123abc“ (das entspricht der GLN Nummer „123abc“) auf dem produktiven SML lautet der entsprechende DNS-Name:

```
B-f5e78500450d37de5aabe6648ac3bb70.iso6523-actorid-
upis.sml.peppolcentral.org.
```

Hinter dieser kryptischen URL wartet der SMP des entsprechenden Teilnehmers auf Anfragen. D.h. dass nur die PEPPOL-Teilnehmer-ID des Empfängers bekannt sein muss, um einen Datenaustausch durchführen zu können. Aus dieser kann wie oben beschrieben der DNS-Name des Empfängers berechnet werden. Dieser führt zur SMP, die entsprechend der SMP-Spezifikation standardisiert abgefragt werden kann.

Der SML geht genau nach diesen Regeln vor, wenn ein neuer Teilnehmer im PEPPOL Netzwerk registriert werden soll. Er berechnet den teilnehmerspezifischen DNS Namen, und legt damit

einen DNS CNAME-Eintrag an, der auf den entsprechenden DNS-Publisher-Eintrag der angegebenen SMP lautet (das kann somit als Alias des Publisher-Eintrags betrachtet werden).

Beispiel: für die fiktive Teilnehmer-ID „0088:123abc“ die der SMP-ID "TESTSMP" zugeordnet ist, würde also der DNS-Name

```
B-f5e78500450d37de5aabe6648ac3bb70.iso6523-actorid-
upis.sml.peppolcentral.org.
```

ein Alias auf den folgenden SMP-Publisher-Eintrag sein:

```
TESTSMP.publisher.sml.peppolcentral.org.
```

Daher liefert die DNS-Auflösung der PEPPOL Teilnehmer-ID "0088:123abc" die IP-Adresse des zugeordneten SMPs zurück.

Die folgende Abbildung 19 zeigt ein Sequenzdiagramm mit den üblichen Aktionen Anlegen, Ändern und Löschen eines PEPPOL-Teilnehmers als Zusammenspiel von SMP, SML und DNS.

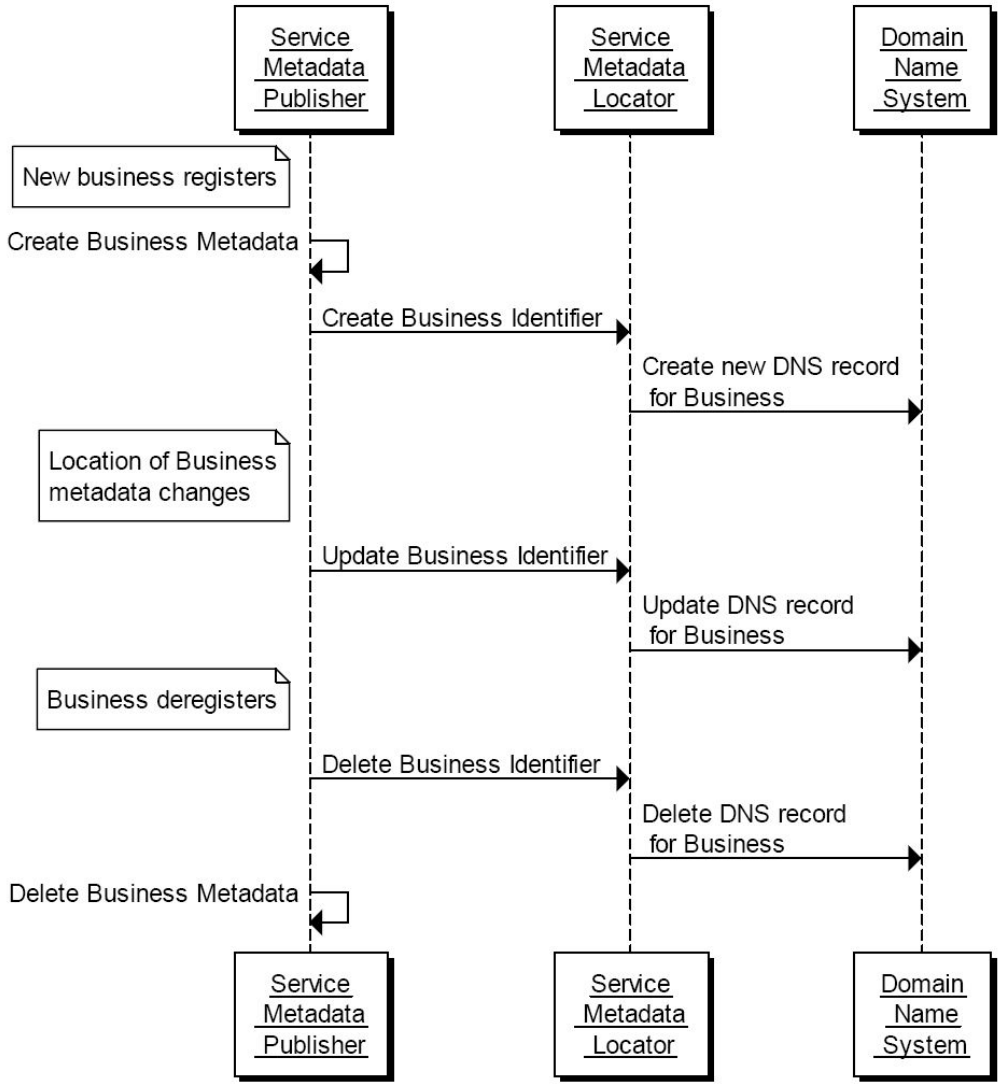


**Abbildung 19: Sequenzdiagramm für einen SMP der Teilnehmer Einträge im SML anlegt, ändert oder löscht (Sylvest et al. 2010b)**

Die folgende Abbildung 20 zeigt den vereinfachten Ablauf einer Registrierung des Teilnehmers "0088:123abc" im PEPPOL-Netzwerk bis hin zum DNS:

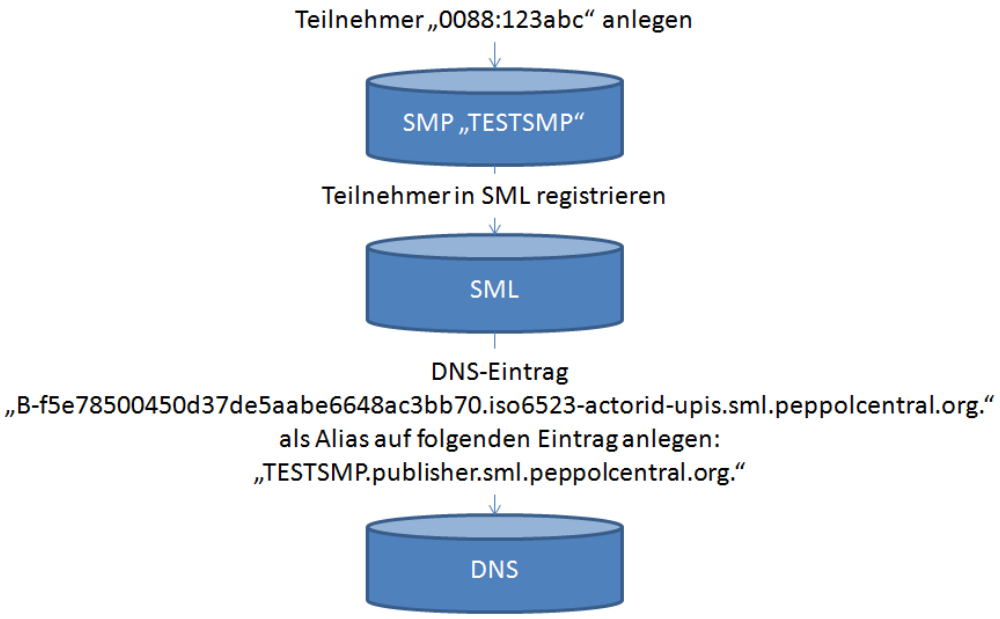


Abbildung 20: Ablauf der Teilnehmerregistrierung

### 4.4.5 Service Metadata Publisher (SMP)

Der PEPPOL Service Metadata Publisher (SMP) ist eine Registry mit Informationen, die für den eigentlichen Datenaustausch benötigt werden. Diese Informationen enthalten u.a. die Endpunkt-URL des Access Points. Bei der SMP handelt es sich um eine dezentrale Registry, d.h. jeder Anbieter eines Access Points kann auch einen SMP anbieten. In der Praxis ist es jedoch durchaus üblich, dass ein SMP gemeinsam von mehreren Service Providern verwendet wird. Die SMP-Spezifikation ist nicht PEPPOL spezifisch, sondern kann auch außerhalb der E-Procurement-Domäne verwendet werden.

Ein SMP, und zwar der des Empfängers, ist in jeden PEPPOL Dokumentenaustausch involviert. Bevor ein Dokument mittels START Profil ausgetauscht wird, muss der Sender anhand der PEPPOL-Teilnehmer-ID des Empfängers ermitteln, unter welcher URL („wo“) der Access Point erreichbar ist. Diese und andere Informationen befinden sich in der SMP und können über eine öffentliche Schnittstelle abgefragt werden. Eine solche Abfrage kann nicht nur im Zuge eines Dokumentenaustauschs durchgeführt werden, sondern bereits vorher zu Informationszwecken.

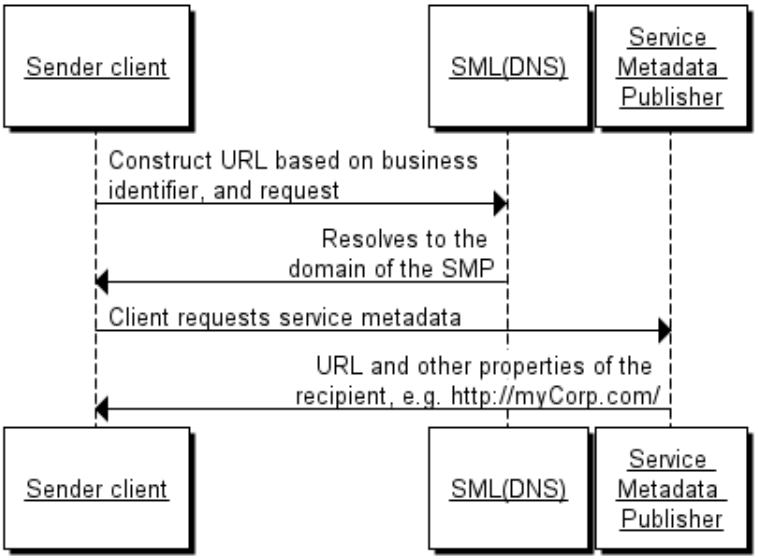


Abbildung 21: Endpunkt Auflösung über den SMP (Sylvest et al. 2010e)

Technisch ist ein SMP ein Service im Internet (kein Web Service!), das auf Maschine-zu-Maschine Kommunikation ausgelegt ist. D.h. dass es standardmäßig keine Benutzeroberfläche gibt, sondern nur Service-Schnittstellen. Diese Schnittstellen basieren auf der REST Technologie und wurden ausschließlich für den Lesezugriff spezifiziert. Die Spezifikation für Schreibzugriffe im SMP muss nicht einheitlich sein, da es stets in der Hoheit des betreibenden Service Providers liegt, ob, wie und wann Daten geändert werden können (Sylvest et al. 2010e).

#### 4.4.5.1 Initialisierung

Um einen SMP in PEPPOL zu registrieren muss einmalig eine Registrierung beim SML durchgeführt werden (siehe Kapitel 4.4.4.1). Anschließend können neue Teilnehmer direkt bei der SMP eingetragen werden, und der SMP muss dafür Sorge tragen, dass die Teilnehmer auch im SML registriert werden. Beim Löschen eines Teilnehmers aus einer SMP muss der entsprechende Teilnehmer auch aus der SML, und damit auch dem DNS, gelöscht werden. Es ist darauf zu achten, dass ein Teilnehmer ausschließlich in einer SMP enthalten sein kann, da sonst die korrekte DNS-Auflösung über den generierten Namen nicht mehr funktionieren kann. Falls ein PEPPOL-Teilnehmer den Service Provider wechselt, so kann es auch vorkommen, dass er einer anderen SMP zugeordnet werden muss. Da die entsprechenden Änderungen im DNS-System einige Zeit in Anspruch nehmen können, kann für die Übergangsperiode im alten SMP eine Weiterleitung auf den neuen SMP eingerichtet werden, sodass unabhängig ob die DNS-Auflösung den alten oder den neuen SMP ergibt, stets die URL des neuen Access Points zurückgeliefert wird. Jeder SMP benötigt ein Zertifikat basierend auf dem "PEPPOL SMP CA" Zertifikat um im SML Teilnehmer anlegen und löschen zu können. Dasselbe Zertifikat wird auch verwendet um die vom SMP erzeugten Antworten elektronisch zu signieren.

#### 4.4.5.2 Datenmodell

Das Datenmodell der SMP unterscheidet grundsätzlich zwischen `ServiceGroup` und `ServiceMetadata` / `SignedServiceMetadata`. Die `ServiceGroup`-Struktur repräsentiert einen Satz von Services eines spezifischen Teilnehmers in einer SMP, und enthält Referenzen (URLs) auf die entsprechenden `SignedServiceMetadata` Objekte. Die `ServiceMetadata`-

Struktur enthält alle Metadaten über ein spezifisches elektronisches Service, um einen (PEPPOL-) Teilnehmer in die Lage zu versetzen einen spezifischen Dokumententyp (z.B. eine elektronische Rechnung) über ein bestimmtes Transport-Protokoll zu erhalten – im Falle von PEPPOL ist das das START Profil. Außerdem identifiziert diese Struktur die Geschäftsprozesse, in denen ein Dokumententyp ausgetauscht werden kann, sowie diverse Betriebsdaten (Aktivierungsdatum, Ablaufdatum, Kontakt-Adresse etc.). Die Informationen aus einer `ServiceMetadata`-Struktur sind ausreichend um einen PEPPOL-Dokumentenaustausch zu initiieren. Im Gegensatz zur `ServiceMetadata`-Struktur ist die `SignedServiceMetadata`–Struktur mit einem „PEPPOL SMP CA“ basierenden Zertifikat digital signiert – sonst sind sie identisch (Sylvest et al. 2010e).

#### *4.4.5.3 REST-API*

Der SMP bietet eine standardisierte REST-API für den Lesezugriff an. Als Vorbedingungen für den Einsatz einer SMP ist es notwendig, dass die Software auf dem Standard-HTTP-Port 80 im Root-Verzeichnis („/“) erreichbar ist. Ein Einsatz von HTTPS ist nicht möglich.

Um die `ServiceGroup`, also alle registrierten Services eines Teilnehmers, abzufragen muss ein HTTP GET Request in der Form `/{identifier scheme}::{id}` durchgeführt werden. Für die fiktive PEPPOL Teilnehmer-ID „0088:123abc“ auf der fiktiven SMP „http://smp.example.org“ lautet die URL also

```
http://smp.example.org/iso6523-actorid-upis%3A%3A0088%3A123abc
```

Zu beachten ist, dass das Zeichen „:“ ein Spezialzeichen in URLs ist, und daher entsprechend dem URL-Encoding als „%3A“ angegeben werden muss.

Um die Metadaten eines spezifischen Services als `SignedServiceMetadata` zu erhalten muss ein HTTP GET Request in der Form `/{identifier scheme}::{id}/services/{docType}` durchgeführt werden. Für die obigen Beispieldaten und den fiktiven Dokumententyp „urn:example:documenttype“ ergibt sich die folgende URL (in einer langen Zeile):

```
http://smp.example.org/iso6523-actorid-upis%3A%3A0088%3A123abc/services/urn%3Aexample%3Adocumenttype
```

Für beide Aufrufe sind die Standard-HTTP-Antwortcodes 200 (Erfolg), 404 (Teilnehmer oder Dokumententyp für Teilnehmer nicht gefunden) und 500 (interner Server-Fehler) vorgesehen.

Die schreibende API ist nicht spezifiziert, da diese nicht von außerhalb des SMP-Betreibers verwendet werden muss. So ist es möglich, die schreibenden Operationen dem lokalen Sicherheitskonzept des SMP-Betreibers anzupassen.

### 4.4.6 START Profil

Secure Trusted Asynchronous Reliable Transport (kurz “START”) ist ein SOAP-basiertes Web Service Profil, über das Access Points miteinander kommunizieren. Es ist sicher und verlässlich, basierend auf folgenden offenen Standards (Sylvest et al. 2010d):

- SOAP 1.1 (http://www.w3.org/TR/2000/NOTE-SOAP-20000508/, Stand: 19.10.2012)
- WS-Addressing 1.0 (http://www.w3.org/TR/ws-addr-soap/, Stand: 19.10.2012)
- WS-Security 1.1 (https://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf, Stand: 19.10.2012) und WS-Security SAML

Token Profile 1.1 (http://docs.oasis-open.org/wss/v1.1/wss-v1.1-errata-os-SAMLTokenProfile.pdf, Stand: 19.10.2012)
- WS-Transfer (http://www.w3.org/TR/2009/WD-ws-transfer-20090924/, Stand: 19.10.2012)
- WS-ReliableMessaging 1.1 (http://docs.oasis-open.org/ws-rx/wsrm/200702/wsrm-1.1-spec-os-01-e1.html, Stand: 19.10.2012)
- SAML 2.0 (http://docs.oasis-open.org/security/saml/v2.0/, Stand: 19.10.2012)

START ist Teil der BUSDOX-Spezifikationen und wurde im Rahmen von PEPPOL entwickelt, ist aber nicht PEPPOL-spezifisch. Vergleichbare Protokolle sind z.B. AS2 (Applicability Statement 2, RFC 4130) und AS3 (Applicability Statement 3, RFC 4823), allerdings gab es zum Zeitpunkt der Entstehung von START keine qualitativ ansprechenden Open Source Implementierungen (AS2) bzw. war die Spezifikation noch nicht fertig (AS3).

Als Ziele der START Spezifikation werden u.a. genannt (Sylvest et al. 2010d):

- Es muss nur dieses eine Profil implementiert werden, um Zugang zu BUSDOX zu bekommen
- Ein einfaches, interoperables Muster zur Kommunikation von Access Points zu definieren
- Sicherstellen, dass Dokumente verlässlich ausgetauscht werden
- Die Vertraulichkeit von übertragenen Dokumenten durch Verschlüsselung sicherstellen
- Integrität und Authentizität von erhaltenen Dokumenten durch Signaturprüfung sicherstellen
- Den Austausch von verschlüsselten Dokumenten ermöglichen
- Metadaten für den Dokumentenaustausch zu definieren
- SAML 2.0 Assertions und Tokens in standardisierter Art und Weise in BUSDOX darstellen

Im Gegensatz dazu werden als Nicht-Ziele genannt:

- Die Überprüfung von Zertifikaten, Teilnehmer-IDs und anderen Details die für eine vollständige Instanz von BUSDOX notwendig sind
- Die Kommunikation mit SML und SMP zu spezifizieren
- Die Verwendung von SAML 2.0 Tokens für andere Zwecke zu beschreiben
- Die Authentifizierung von Sendern durchzuführen

START verwendet nur die WS-Transfer „Create“ Funktion – alle anderen WS-Transfer Funktionen sind nicht spezifiziert. Das Ziel von START ist es einzig und alleine Dokumente sicher und verlässlich auszutauschen, sonst gibt es keine weitere Funktionalität.

#### *4.4.6.1 Metadaten*

Um auch verschlüsselte Dokumente transportieren zu können, ist es unabdingbar, dass neben dem eigentlichen Dokument noch zusätzliche Metadaten übertragen werden, die für das Routing relevante Informationen enthalten. Da es sich um ein SOAP-basiertes Profil handelt wurden diese Metadaten in den SOAP-Header integriert, während die ausgetauschte Nachricht im SOAP-Body enthalten ist. BUSDOX definiert sechs Header-Einträge die von allen Access Points verstanden werden müssen (Sylvest et al. 2010a; Sylvest et al. 2010d):

- `SenderIdentifier`: die ID des Senders (muss nicht zwangsläufig eine PEPPOL Teilnehmer-ID sein, falls es sich um einen unidirektionalen Datenaustausch handelt).
- `RecipientIdentifier`: die PEPPOL Teilnehmer-ID des Empfängers. Damit kann der empfangende Access Point die Nachricht an den endgültigen Empfänger weiterleiten.
- `DocumentIdentifier`: die ID des verwendeten PEPPOL-Dokumententyps.
- `ProcessIdentifier`: die ID des verwendeten PEPPOL-Geschäftsprozesses (BIS).
- `MessageIdentifier`: eine eindeutige ID für diese Dokumentenübermittlung (z.B. eine UUID). Diese dient hauptsächlich der einfachen Nachverfolgbarkeit von Nachrichten.
- `ChannelIdentifier`: eine ID die zur Unterscheidung von verschiedenen Kommunikationskanälen innerhalb eines Access Points dient (wird hauptsächlich im Zusammenhang mit LIME benötigt).

### 4.4.7 LIME Profil

Das Lightweight Message Exchange Profile (LIME) ist ein einfacheres Web Service Profil als START und wurde von PEPPOL speziell für den Einsatz in KMUs (kleine und mittlere Unternehmen) entwickelt. Das Ziel dieses Profils ist es, KMUs einen einfachen Zugang zur BUSDOX-Infrastruktur zu ermöglichen. Auch das LIME Profil basiert auf offenen Standards (Sylvest et al. 2010c):

- HTTPS mit Basic Authentication
- SOAP 1.1 (http://www.w3.org/TR/2000/NOTE-SOAP-20000508/, Stand: 19.10.2012)
- WS-Addressing 1.0 (http://www.w3.org/TR/ws-addr-soap/, Stand: 19.10.2012)
- WS-Transfer (http://www.w3.org/TR/2009/WD-ws-transfer-20090924/, Stand: 19.10.2012)

LIME ist für die direkte Anbindung von E-Procurement Applikationen an einen Access Point entwickelt worden. Um den Unterschied zwischen START und LIME zu erklären, kann die Architektur von E-Mail als Analogie herangezogen werden: E-Mail Server kommunizieren untereinander mittels SMTP, während E-Mail-Clients über POP3 oder IMAP mit dem Server kommunizieren. Die Protokolle sind sehr unterschiedlich, da auch die Anforderungen an Server und Client sehr unterschiedlich sind. Server müssen ununterbrochen erreichbar sein, während Clients nur zu bestimmten Zeiten mit dem Server verbunden sind. Genauso ist auch der Unterschied zwischen START und LIME zu verstehen: die Server, die nonstop online sein müssen, kommunizieren mit Hilfe von START, während LIME speziell für die Kommunikation mit Clients ausgelegt ist. Das impliziert auch, dass LIME weitere, über den Dokumentenaustausch hinausgehende, Funktionalitäten anbieten muss, wie zum Beispiel die Abfrage aller auf dem Server für einen Client verfügbaren Nachrichten (Sylvest et al. 2010c).

### 4.4.8 Access Point (AP)

Ein PEPPOL Access Point ist eine Software Komponente, die mindestens das START Profil implementiert. Ein AP als solcher ist nirgends in PEPPOL spezifiziert, wohl aber sind einige Vorgaben innerhalb der START und LIME Spezifikationen zu finden. So ist für START und LIME vorgegeben, dass HTTPS für den Transport eingesetzt werden muss. Außerdem müssen die Nachrichten bei START verschlüsselt sein, was den Einsatz eines auf der „PEPPOL AP CA“ basierenden Zertifikats notwendig macht.

Um die Umsetzung der doch recht komplizierten START Spezifikation zu erleichtern wurde von PEPPOL eine einheitliche WSDL-Datei (Web Services Description Language) zur Verfügung gestellt. Diese enthält alle notwendigen Einstellungen für Java-Implementierungen, und kann unter https://joinup.ec.europa.eu/svn/peppol/PEPPOL_EIA/1-ICT_Architecture/1-ICT-Transport_Infrastructure/14-ICT-Services-Components/ICT-Transport-START_WSDL_SW-200 (Stand 20.10.2012) abgerufen werden.

Für LIME gibt es leider keine „offiziell einheitliche“ WSDL-Datei, obwohl diese auch implementierungsunabhängig spezifiziert wurde. Der Grund dafür ist die mangelnde Akzeptanz dieser Schnittstelle, weshalb Sie projektintern nicht weiterentwickelt wurde. Die WSDL-Datei der Implementierung PEPPOL Silicone (siehe Kapitel 4.4.12.2) kann jedoch als Referenz herangezogen werden. Die aktuelle Version der WSDL-Datei kann unter https://peppol-silicone.googlecode.com/svn/trunk/java/transport-lime-client/src/main/resources/WEB-INF/wsdl (Stand 20.10.2012) abgerufen werden.

### 4.4.9 Dokumenten Validierung

Die Validierung (Überprüfung) von übermittelten PEPPOL-Dokumenten ist ein wichtiger Bestandteil des Dokumentenaustauschs. Dafür wurde in PEPPOL das von der CENBII entwickelte „Validation Framework“ verwendet. Dieses sieht eine mehrstufige Überprüfung von Dokumenten je nach verwendetem Kontext vor. Die Überprüfung von Dokumenten ist ein wichtiger Bestandteil der Interoperabilitätsbestrebungen von PEPPOL, da so Fehler und Inkompatibilitäten frühzeitig und automatisiert erkannt werden können.

#### *4.4.9.1 CENBII Validation Framework*

Wie auch bei den Dokumenten, sind die Validierungsregeln von CENBII nur abstrakt, also ohne Verbindung zu einer Syntax, verfasst. PEPPOL spricht in diesem Fall von Geschäftsregeln („business rules“). Erst durch das „Binding“ dieser abstrakten Regeln auf UBL sind Validierungsregeln („validation rules“) entstanden. Es werden zwei unterschiedliche Fehlerklassen definiert: „Warnung“ und „Fataler Fehler“. Falls im Zuge der Überprüfung nur eine oder mehrere Warnungen ermittelt werden, so ist das Gesamtdokument trotzdem als gültig zu betrachten, während wenn mindestens ein fataler Fehler gefunden wird, das Dokument als ungültig angesehen werden muss (Bausà Peris 2012).

So lautet z.B. die abstrakte E-Bestellungs-Regel „BIICORE-T01-R435“[3]:

„Element 'PartyIdentification' may occur at maximum 1 times.”

Diese Regel muss im Kontext „Customer“ ausgeführt werden und ist eine Warnung. Im UBL-Kontext wird diese Regel in folgenden XPath-Ausdruck überführt:

```
count(/ubl:Order/cac:BuyerCustomerParty/cac:Party/cac:PartyIdentification)<=1
```

[3] Aus der Datei „biicore-T01-BusinessRules-v01.ods“, enthalten in https://joinup.ec.europa.eu/svn/peppol/PEPPOL_EIA/1-ICT_Architecture/1-ICT-PostAward_eProcurement/14-ICT-Services-Components/ICT-PostAward-TRDM_010a_SCH-201.zip (Stand 21.10.2012)

Wenn ein anderes Syntax-Binding als UBL verwendet wird, so müssen auch die Validierungsregeln dementsprechend angepasst werden. Die Geschäftsregeln sind davon nicht betroffen, und dürfen auf keinen Fall geändert werden.

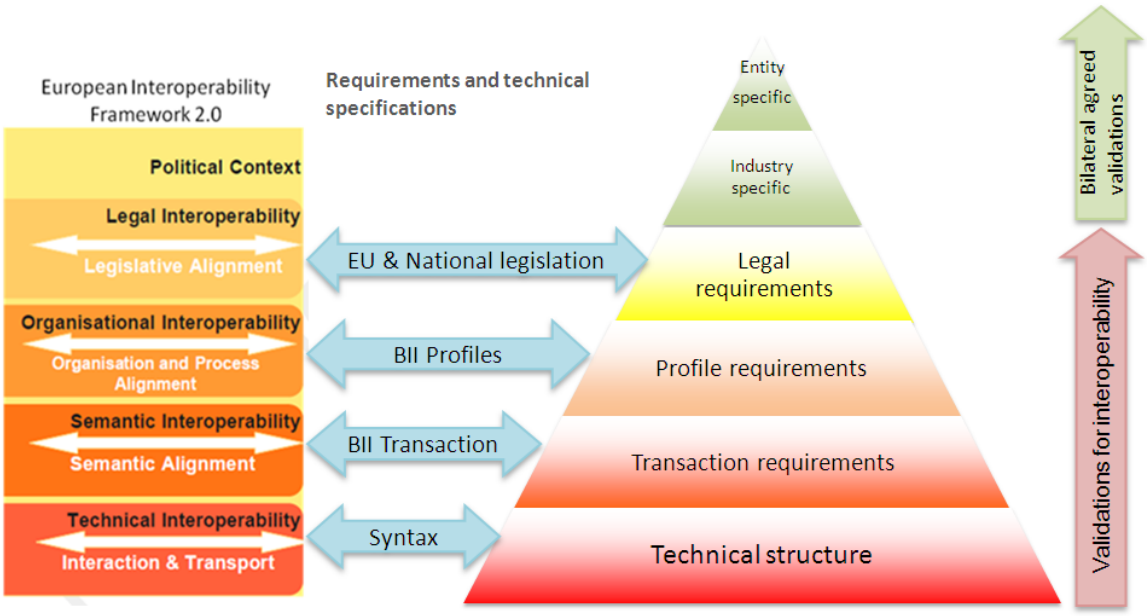


Abbildung 22: CEN/BII Validation Framework (Pedersen und Skulason 2012)

Abbildung 22 zeigt den Aufbau des CENBII Validation Frameworks und setzt ihn in Verbindung zum EIF 2.0. Es wird hier auch von der „pyramid of validation" gesprochen. Bei der Überprüfung eines Dokuments muss die Pyramide von unten nach oben durchgegangen werden, wobei mindestens die untersten drei Ebenen überprüft werden müssen, da diese essentiell für die Einhaltung der Interoperabilität sind. Sobald der erste fatale Fehler auftritt, egal auf welche Ebene, gilt die Überprüfung als fehlgeschlagen. Die folgende Liste beschreibt die unterschiedlichen Ebenen für die Validierung eines Dokuments (Bausà Peris 2012):

- **Technical structure**: Dabei handelt es sich um die Überprüfung des Dokuments gegen ein XML-Schema (XSD), oder eine vergleichbare Struktur-Definition. Das XML-Schema stellt die Übermenge aller erlaubten XML-Elemente dar, und setzt Beschränkungen bzgl. der Reihenfolge und der Anzahl von Elementen. Jedes PEPPOL-UBL-Dokument muss dem passenden UBL-XSD entsprechen (eine E-Rechnung muss der UBL-Invoice-XSD entsprechen, ein E-Katalog der UBL-Catalogue-XSD etc.).
- **Transaction requirements**: Auf dieser Ebene geht es um die Geschäftsregeln von CENBII, für die in einer bestimmten Transaktion ausgetauschten Dokumente. Wie im Kapitel 4.3.1 erläutert, kann eine Transaktion in verschiedenen Profilen wiederverwendet werden. Genauso werden alle Geschäftsregeln, die für eine Transaktion gelten, in allen Profilen gleich angewendet, in denen diese Transaktion enthalten ist.
- **Profile requirements**: Auf der Profil-Ebene geht es um zusätzliche Einschränkungen die sich durch das verwendete Profil ergeben. Als Beispiel kann die E-Rechnung genannt werden: wenn Sie im Rahmen des BIS 6a (Procurement) verwendet wird, so ist die Referenz auf die Bestellung verpflichtend, da vorher eine elektronische Bestellung

durchgeführt werden muss. Während wenn die E-Rechnung im Kontext von BIS 4a (Basic invoice only) ausgetauscht wird, diese Regel nicht existiert.

- **Legal requirements**: Bei den rechtlichen Anforderungen geht es um spezifische Anforderungen die ein Land, bzw. ein Empfängerkreis, dem Sender auferlegt. So existiert z.B. in Österreich die Anforderung, dass die UID-Nummer des Leistungsempfängers auf einer Rechnung aufscheinen muss, wenn der Rechnungsbetrag größer als € 10.000,-- ist. Da diese Anforderungen von Land zu Land unterschiedlich sind, wurde diese Ebene hinzugefügt. Im Rahmen von PEPPOL konnten für Österreich, Dänemark, Italien und Norwegen Spezialregeln identifiziert und umgesetzt werden.
- **Industry specific**: Diese Ebene ist optional, und gibt Regeln vor, die für einen größeren Empfängerkreis (z.B. einen Industriezweig), gelten. In PEPPOL wird diese Ebene für spezifische Anforderungen von Bundes-Empfängern genutzt. So muss z.B. bei einer Rechnung an den österreichischen Bund genau eine Kontoverbindung angegeben werden.
- **Entity specific**: Die letzte Ebene ist ebenfalls optional, und dient als Auffangbecken für alle Regeln, die durch die anderen Ebenen nicht abgedeckt wurden. Sie erlaubt die Verwendung von Regeln, die nur zwischen zwei Partnern gelten, und daher sehr spezifisch sind. In PEPPOL wird diese Ebene nicht verwendet.

#### *4.4.9.2 Technische Umsetzung*

Die Regeln der „Technical structure“ Ebene werden in PEPPOL mit Hilfe von XML-Schemas (XSDs) abgebildet. Auf allen anderen Ebenen kommt Schematron (siehe Kapitel 2.3) zum Einsatz. Zusätzlich zu XSD und Schematron kommen noch Codelisten zum Einsatz, die in PEPPOL direkt als Schematron-Regeln abgebildet werden. PEPPOL Silicone bietet eine Umsetzung sämtlicher in PEPPOL verfügbarer Validierungsregeln auf Basis von UBL-XSDs und Schematron Regeln.

### 4.4.10 Dokumenten Visualisierung

Da es sich bei den in PEPPOL übertragenen Dokumenten um technische Dokumente handelt, die für Menschen schwer lesbar sind, wurden im Rahmen von PEPPOL auch einige XSLT Stylesheets entwickelt, um diese Dokumente menschenlesbar zu machen. Für die Dokumententypen Bestellung, Rechnung und Gutschrift liegen solche Stylesheet vor.

### 4.4.11 Dokumentenaustausch

In diesem Kapitel wird ein beispielhafter Dokumentenaustausch zwischen zwei Geschäftspartnern über die PEPPOL Transport Infrastruktur gezeigt. Es wird das „4-corner model“ angewendet. Die Voraussetzung für einen Dokumentenaustausch ist, dass der Empfänger in einer SMP registriert ist. Dabei soll, wenn möglich, eine bereits existierende Identifikation wie z.B. eine GLN (Global Location Number) oder eine DUNS-Nummer (Dun & Bradstreet) verwendet werden. Diese Identifikation muss extern zwischen den Geschäftspartnern ausgetauscht werden, bevor ein Dokumentenaustausch im PEPPOL Netzwerk stattfinden kann. Der Sender muss in keiner SMP registriert sein, außer die gewählte PEPPOL BIS erfordert eine Rückantwort auf den ersten Dokumentenaustausch. Dadurch entsteht eine implizite Notwendigkeit, da beide Parteien sowohl als Sender als auch als Empfänger auftreten.

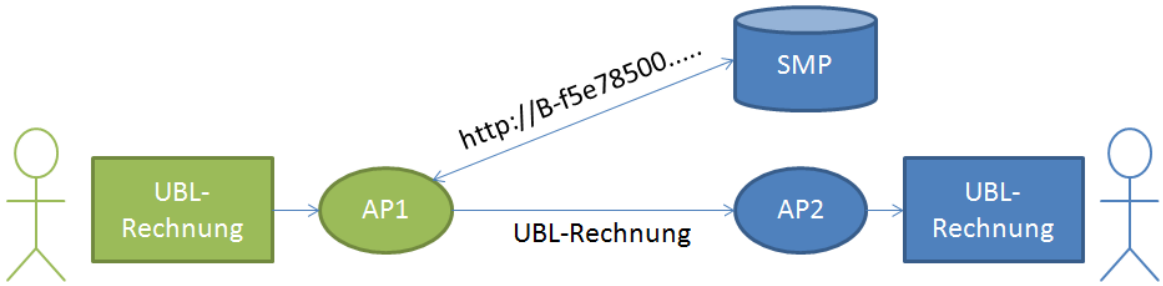


Abbildung 23: Schematische Darstellung des Dokumentenaustauschs einer E-Rechnung

Abbildung 23 zeigt die schematische Darstellung einer Rechnungsübermittlung über die PEPPOL Transport Infrastruktur. Auf der linken Seite, in Grün gehalten, befindet sich der Sender. Auf der rechten Seite, in Blau gehalten, befindet sich der Empfänger. Der Sender muss eine Rechnung im Format UBL erzeugen; wie, ist nicht Teil der PEPPOL-Spezifikationen. Diese UBL-Rechnung wird dann gemeinsam mit der PEPPOL-Teilnehmer-ID des Empfängers an den „AP1“, den Sender-Access Point, übertragen. Auch hierzu macht PEPPOL keine Angaben, da, wie bereits erwähnt, davon ausgegangen wird, dass es bereits existierende Geschäftsbeziehungen zwischen dem Sender und seinem Service Provider, der den „AP1“ betreibt, gibt. Der AP1 bildet dann aus der Teilnehmer-ID des Empfängers den entsprechenden DNS Namen („B-f5e78500….“) und fügt die entsprechenden Parameter für den SMP-Aufruf hinzu, um eine SMP `SignedServiceMetadata`-Struktur zu erhalten (siehe Kapitel 4.4.5.3). Mit dieser URL kann nun der Empfänger-SMP aufgerufen werden, der als Ergebnis u.a. die Endpunkt-URL von „AP2“, dem Empfänger-Access Point für Rechnungen, zurückliefert. Nun beginnt AP1 mit der eigentlichen Übertragung der Rechnung zu AP2 unter Zuhilfenahme des START Profils. Für AP1 ist die PEPPOL-Übertragung nun beendet, und es liegt in der Verantwortung von AP2 die Rechnung an den eigentlichen Empfänger weiterzuleiten.

Wie aus dieser Beschreibung ersichtlich ist, ist der SML nicht in den Datenaustausch involviert. Die Auflösung des Namens „B-f5378500…“ erfolgt ausschließlich über das DNS-System.

### 4.4.12 Implementierungen

Für alle aufgezählten Komponenten gibt es Implementierungen im Rahmen des PEPPOL Projekts. Die Historie ist kompliziert, da es einige Wechsel der Hostingprovider gab. Ursprünglich waren die Projekte auf www.osor.eu gehostet, eine von der Europäischen Kommission (EC) geförderte Plattform, die mit Sourceforge (www.sf.net) vergleichbar war. Diese Plattform wurde dann mit der ebenfalls von der EC geförderten Plattform www.semic.eu zu www.joinup.eu zusammengelegt, auf der noch immer Teile der Referenzimplementierungen zu finden sind.

Nachdem im dortigen Subversion (SVN) Repository jedoch im Laufe der Zeit Chaos entstand, wurde Anfang 2012 entschieden, den Entwicklungsbereich auszulagern. Dieser befindet sich nun auf Google Code als Projekt mit dem Namen „PEPPOL Silicone“ (siehe Kapitel 4.4.12.2). Im Repository auf Joinup werden nur noch die Releases veröffentlicht – im Quellformat und als binäre Pakete. Allerdings wird auf Joinup auch weiterhin noch eine AccessPoint Implementierung für Java und .NET gepflegt, allerdings mit wenigen Änderungen. Diese Implementierung wird im Detail im Kapitel 4.4.12.1 beschrieben. Parallel dazu entstand ein Spin-Off der Java AccessPoint Implementierung mit dem Namen „Oxalis“, deren besonderer Fokus auf einer einfachen Installation liegt (siehe Kapitel 4.4.12.3).

Alle Implementierungen sind als Open Source Software (OSS) dual lizensiert, sowohl unter der European Union Public License 1.1 (EUPL), als auch unter der Mozilla Public License 1.1 (MPL). Durch diese duale Lizensierung kann der Anwender selbst entscheiden, welchen Lizenzbedingungen er Folge leisten möchte. Der Lizenztext der EUPL kann der Webseite http://joinup.ec.europa.eu/software/page/eupl (Stand 27.9.2012) entnommen werden, während der Lizenztext der MPL 1.1 unter http://www.mozilla.org/MPL/1.1/ (Stand 27.9.2012) nachgelesen werden kann.

#### 4.4.12.1 Referenzimplementierung

Ursprünglich wurden alle Artefakte auf www.osor.eu bzw. www.joinup.eu entwickelt. Da es aber unterschiedliche Auffassungen im Entwicklungsteam gab, wurde ein Großteil der Entwicklung nach Google Code verschoben, und nur alte Teile, bzw. eine reine START-Access Point Implementierung verblieben auf Joinup. Die alten Artefakte der Referenzimplementierung sind nicht mehr empfehlenswert, da sie durch neuere Versionen in PEPPOL Silicone abgelöst wurden. Nur die Standalone-START-Access Point-Implementierung in Java und .NET wurde dort noch weitergepflegt. Das ist die einzige offizielle Komponente, die auch in .NET vorliegt, während alle anderen Softwarekomponenten nur in Java vorhanden sind. Allerdings gibt es noch immer Inkompatibilitäten zwischen der Java und der .NET Version, was zu einer beschränkten Einsetzbarkeit führt. Die Ursache für die Probleme liegt jedoch nicht in der Implementierung selbst, sondern in den verwendeten Web Service Bibliotheken.

Die Referenzimplementierung gibt die WSDL (Web Service Definition Language) Datei vor, die von allen AccessPoint Implementierungen zu verwenden sind. Diese befindet sich im Web unter https://joinup.ec.europa.eu/svn/peppol/dev/infrastructure/start_wsdl (Stand 27.09.2012).

#### 4.4.12.2 PEPPOL Silicone

PEPPOL Silicone ist ein "Spin-off" der PEPPOL Referenzimplementierung, welche sämtliche PEPPOL Komponenten als Java Implementierung enthält: SML, SMP, AccessPoint, START, LIME, Validierung und Visualisierung. Außerdem ist ein Web GUI in einer rudimentären Version enthalten, welches jedoch noch keinen einsetzbaren Status hat.

Die Entwicklung wird auf der freien Hosting Plattform Google Code als OSS durchgeführt. Die Projekt Webseite ist http://code.google.com/p/peppol-silicone/ (Stand 27.9.2012). Als Lizenz kommen sowohl die EUPL 1.1 als auch die MPL 1.1 zum Einsatz. Alle Komponenten stehen vollständig als Java 1.6 Quellcode zur Verfügung. Als Build-Umgebung wird Apache Maven 3 mit einer aktiven Internetverbindung vorausgesetzt. Die Qualität der Komponenten ist gut, aber gerade im Bereich der Dokumentation bestehen einige Lücken. Trotzdem ist es die empfohlene PEPPOL Implementierung, da sowohl SML, SMP, AccessPoint als auch Dokumentenvalidierung enthalten sind.

#### 4.4.12.3 Oxalis

Oxalis wurde vom norwegischen PEPPOL Mitglied DIFI (Agency for Public Management and eGovernment) in Auftrag gegeben, als es PEPPOL Silicone noch nicht gab. Das erklärte Ziel war es, eine einfach zu verwendende AccessPoint Implementierung zu erstellen, die mit der Referenzimplementierung kompatibel ist. Die Projektseite befindet sich unter https://github.com/difi/oxalis (Stand 27.9.2012).

# 5 ERPOL

Das Ziel von ERPOL ist es den bidirektionale Datenaustausch zwischen ERPEL und PEPPOL zu definieren und umzusetzen. Der Name setzt sich aus den Anfangsbuchstaben von „ERPEL“ und dem Ende von „PEPPOL“ zusammen. Im Rahmen dieser Arbeit werden Schnittstellen zwischen der ERPEL-Registry und dem PEPPOL-SMP sowie zwischen dem ERPEL-Messaging Hub und dem PEPPOL-Access Point spezifiziert und umgesetzt.

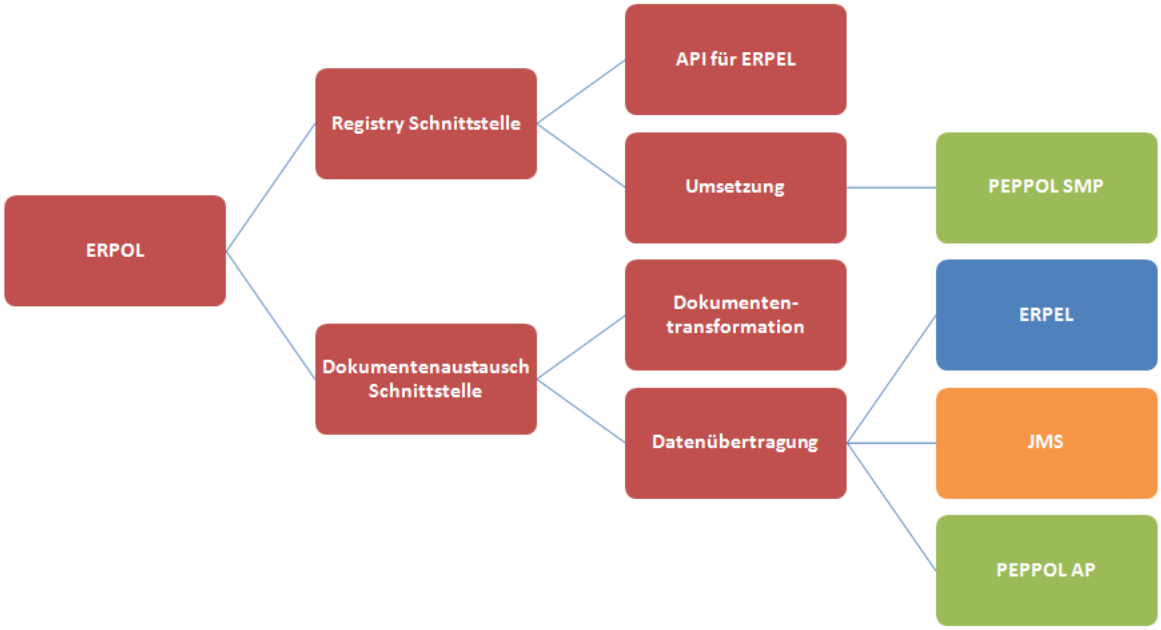


Abbildung 24: Übersicht über die Komponenten von ERPOL

Abbildung 24 zeigt die Übersicht über die Komponenten von ERPOL und wie diese mit ERPEL und PEPPOL in Verbindung stehen.

Für diese Arbeit wurde von PEPPOL eine Academic License und damit je ein Zertifikat für PEPPOL SMP und Access Point zur Verfügung gestellt. Mit diesen Zertifikaten ist es möglich einen echten Datenaustausch über die PEPPOL-Infrastruktur vorzunehmen, jedoch mit der Einschränkung, dass das Zertifikat nicht kommerziell verwendet werden darf.

## 5.1 Schnittstellenspezifikation

Zum Datenaustausch zwischen ERPEL und PEPPOL werden verschiedene Schnittstellen benötigt. Einerseits wird eine unidirektionale Schnittstelle von der ERPEL-Registry zum PEPPOL-SMP benötigt, andererseits wird eine bidirektionale Schnittstelle zwischen dem ERPEL-Messaging Hub und dem PEPPOL-AP für den Dokumentenaustausch benötigt.

Für alle Schnittstellen ist ein entsprechendes Logging zu implementieren, damit die einzelnen Schritte auch im Nachhinein noch nachvollziehbar sind. Dabei ist darauf zu achten, dass keine Datenschutz-relevanten Informationen protokolliert werden, sondern nur technische Informationen.

Für die Umsetzung aller Schnittstellen soll ausschließlich freie Open Source Software verwendet werden, deren Lizenzen auch die Verwendung in kommerziellen Produkten ermöglichen (z.B. Apache 2 Lizenz nicht jedoch die GPL-Lizenz).

### 5.1.1 Dokumententypen

ERPEL hat genau einen Dokumententyp mit sechs Subdokumenttypen und PEPPOL unterstützt verschiedene Dokumenttypen. Die folgende Tabelle enthält das Mapping vom ERPEL-Subdokumenttyp auf den PEPPOL-Dokumenttyp.

| ERPEL-Subdokumenttyp | PEPPOL-Dokumenttyp |
|---|---|
| Request for quote (Ausschreibung) | --- |
| Offer (Angebot) | Teilweise via Catalogue |
| **Order (Bestellung)** | **Order** |
| **Order confirmation (Bestellbestätigung)** | **Order confirmation** |
| Dispatch advice (Lieferschein) | --- |
| **Invoice (Rechnung)** | **Invoice und CreditNote** |

Tabelle 1: Mapping von ERPEL-Subdokumenttyp auf PEPPOL-Dokumenttyp

Ein teilweises Mapping vom ERPEL-Angebot auf den PEPPOL-Katalog macht im Rahmen von ERPOL keinen Sinn, da nur vollständige Dokumente ausgetauscht werden sollen. Daher bleiben drei Dokumententypen übrig, die in der oberen Tabelle fett dargestellt sind.

Für den ProcessIdentifier muss es ebenfalls ein Mapping geben. ERPEL unterstützt genau einen Prozess mit verschiedenen optionalen Aktivitäten. PEPPOL definiert hingegen unterschiedliche Prozesse, die ausschließlich verpflichtende Aktivitäten enthalten. Um ein bestmögliches Mapping ohne Informationsverlust zu gewährleisten, werden nur die einfachen PEPPOL-Prozesse (BIS 3a und BIS 4a - siehe Kapitel 4.3.1) registriert und die Prozessverwaltung muss in ERPEL durchgeführt werden. Weder für die Bestellbestätigung noch für die Gutschrift gibt es in PEPPOL einen Prozesstyp, der ausschließlich diesen Dokumententyp überträgt. Bestellbestätigungen sind nur im Prozesstyp BIS 6a (siehe Kapitel 4.3.1) enthalten, während Gutschriften nur in BIS 5a und BIS 6a (siehe Kapitel 4.3.1) enthalten sind - beide BIS spezifizieren einen Prozess, der den Austausch von mehreren verschiedenen Dokumententypen enthält. Daraus ergibt sich folgendes Mapping für die verbleibenden zwei ERPEL-Subdokumententypen:

| ERPEL-Subdokumenttyp | PEPPOL-Dokumenttyp ID | PEPPOL-Prozesstyp ID |
|---|---|---|
| Order (Bestellung) | urn:oasis:names:specification:ubl:schema:xsd:Order-2::Order##urn:www.cenbii.eu:transaction:biicoretrdm001:ver1.0:#urn:www.peppol.eu:bis:peppol3a:ver1.0::2.0 | urn:www.cenbii.eu:profile:bii03:ver1.0 |
| Invoice (Rechnung) | urn:oasis:names:specification:ubl:schema:xsd:Invoice-2::Invoice##urn:www.cenbii.eu:transaction:biicoretrdm010:ver1.0:#urn:www.peppol.eu:bis:peppol4a:ver1.0::2.0 | urn:www.cenbii.eu:profile:bii04:ver1.0 |

Tabelle 2: ERPEL-Subdokumenttyp auf PEPPOL Identifier Mapping

Aus diesen Anforderungen ergeben sich die beiden Dokumenttypen *Order* und *Invoice*, für die ein vollständiges und durchgängiges Mapping zwischen ERPEL und PEPPOL umsetzbar ist.

## 5.2 Registry-Schnittstelle

Die Registry-Schnittstelle dient zum Austausch der Daten zwischen der ERPEL-Registry und dem PEPPOL-SMP. Da nicht jeder ERPEL-Teilnehmer im PEPPOL-Netzwerk vertreten sein muss, gibt es in der ERPEL-Registry eine spezifische Kennung für alle PEPPOL-Teilnehmer (einen Alias). Diese Kennung muss der PEPPOL-Identifier des Teilnehmers sein, der pro Teilnehmer im PEPPOL-Netzwerk eindeutig sein muss (z.B. *9914:ATU12345678*). Die ERPEL-Registry ist das führende System, weshalb diese Schnittstelle unidirektional in Richtung des PEPPOL-SMP ist. Wenn sich PEPPOL-relevante Daten eines ERPEL-Teilnehmers ändern, müssen diese auch im SMP geändert werden.

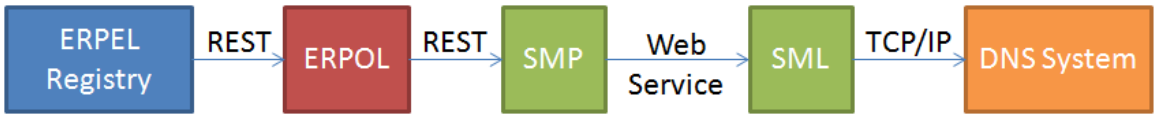


Abbildung 25: Ablauf der Registry-Schnittstelle

Abbildung 25 zeigt den gesamten Ablauf der Registry-Schnittstelle von ERPEL bis zum DNS Server. Theoretisch könnte die ERPEL-Registry auch direkt mit dem SMP kommunizieren, um die technischen Details der Schnittstelle zu verbergen, wurde aber entschieden, dass ERPOL als Abstraktionsebene eingezogen wird. Das hat den Vorteil, dass im Falle einer Änderung im SMP oder in der Schnittstelle des SMPs nur ERPOL adaptiert werden muss. Außerdem können über die vereinfachte Schnittstelle von ERPOL theoretisch auch andere Applikationen Einträge im SMP erzeugen.

### 5.2.1 Inhaltliche Beschreibung

Der SMP benötigt folgende Daten um einen gültigen Eintrag erstellen zu können:

- PEPPOL-Teilnehmer ID (ParticipantIdentifier)
- PEPPOL-Dokumententyp ID (DocumentIdentifier)
- PEPPOL-Prozesstyp ID (ProcessIdentifier)
- URL des PEPPOL Access Points (EndpointReference)
- Ein Flag, ob das Geschäftsdokument eine elektronische Signatur haben muss oder nicht(RequireBusinessLevelSignature) – wird derzeit nicht aktiv verwendet
- Optional die minimale Autorisierungsebene (MinimumAuthenticationLevel) – wird derzeit nicht aktiv verwendet
- Optional ein Datum, ab wann der Eintrag gültig ist (ServiceActivationDate)
- Optional ein Datum, bis wann der Eintrag gültig ist (ServiceExpirationDate)
- Den Base64-kodierten, DER-verschlüsselten Public Key des Access Point Zertifikats (Certificate)
- Eine optionale Beschreibung (ServiceDescription)
- Eine E-Mail Adresse für den technischen Support (TechnicalContactUrl)
- Eine optionale URL für technische Informationen zu diesem Eintrag (TechnicalInformationUrl)
- Den Namen des zu verwendenden Protokolls (transportProfile)

Einige der Informationen können automatisiert vergeben werden (da sie implementierungs-spezifisch sind), andere müssen explizit angegeben werden. In der folgenden Tabelle wird die Herkunft der einzelnen Datenfelder zusammengefasst:

| Datenfeld | Herkunft |
|---|---|
| ParticipantIdentifier | Muss von der ERPEL-Registry mitgegeben werden |
| DocumentIdentifier | Muss von der ERPEL-Registry mitgegeben werden |
| ProcessIdentifier | Muss von der ERPEL-Registry mitgegeben werden |
| EndpointReference | Konstant aus ERPOL |
| RequireBusinessLevelSignature | Konstant „false“ (da ERPEL keine Signaturen verlangt) |
| MinimumAuthenticationLevel | Konstant „1“ |
| ServiceActivationDate | Kann optional von der ERPEL-Registry mitgegeben werden |
| ServiceExpirationDate | Kann optional von der ERPEL-Registry mitgegeben werden |
| Certificate | Konstant aus ERPOL |
| ServiceDescription | Kann optional von der ERPEL-Registry mitgegeben werden |
| TechnicalContactUrl | Konstant aus ERPOL |
| TechnicalInformationUrl | Konstant aus ERPOL |
| transportProfile | Konstant „busdox-transport-start“ |

Tabelle 3: SMP Datenfelder für einen einzelnen Eintrag

Die meisten Punkte in der oberen Tabelle sind entweder konstant oder optional. Nur ParticipantIdentifier, DocumentIdentifier und ProcessIdentifier sind Pflichtfelder, die von der ERPEL-Registry an die Schnittstelle übergeben werden müssen. Der ParticipantIdentifier muss wie bereits beschrieben, in der ERPEL-Registry gepflegt werden. Ein Problem sind DocumentIdentifier und ProcessIdentifier, da für diese ein Mapping zwischen ERPEL und PEPPOL verwendet werden muss (siehe Kapitel 5.1.1).

### 5.2.2 Technische Beschreibung

Nachdem die inhaltlichen Anforderungen an die Registry-Schnittstelle definiert sind, kann die technische Schnittstelle beschrieben werden. Die folgenden Use Cases sollen durch die Schnittstelle abgebildet werden:

- Erzeugen eines neuen PEPPOL-SMP-Eintrags
- Ändern eines existierenden PEPPOL-SMP-Eintrags
- Löschen eines existierenden PEPPOL-SMP-Eintrags
- Abfrage der Daten eines existierenden PEPPOL-SMP-Eintrags

Als Schnittstellentechnologie soll REST eingesetzt werden, da es im Vergleich zu Web Services einen geringeren Overhead während der Entwicklung und der Ausführung hat. Sowohl Request als auch Response sollen JSON-kodiert sein, da es ein flexibles und schlankes Datenformat ist. Zur Absicherung der Netzwerkverbindung wird HTTPS eingesetzt.

Zum Testen der Schnittstelle werden JUnit-Tests verwendet, die die entsprechenden Aufrufe tätigt. Eine Beschreibung befindet sich in Kapitel 5.4.2.

#### *5.2.2.1 Erzeugen und Ändern eines SMP-Eintrags*

Die aufzurufende URL lautet: `/registryupdate`

Die zu verwendende HTTP-Methode ist: `PUT` (da PUT idempotent ist kann es zum Erzeugen und zu Ändern verwendet werden)

Der zu übergebende Parameter ist ein JSON-Objekt mit folgendem Layout:

```
{
  participantid: string
  documenttypeid: string
  processtypeid: string
  ( validfrom: string )?
  ( validto: string )?
  ( description: string )?
}
```

- `participantid` ist die ID des PEPPOL-Teilnehmers und ist verpflichtend.
- `documenttypeid` ist die ID des PEPPOL-Dokumenttyps und ist verpflichtend.
- `processtypeid` ist die ID des PEPPOL-Prozesstyps und ist verpflichtend.
- `validfrom` ist ein optionales Datum ab wann der Eintrag gültig ist. Der String muss als XML-Schema-Datum ohne Zeitzone kodiert sein (JJJJ-MM-TT) und ist optional.
- `validto` ist ein optionales Datum bis wann der Eintrag gültig ist. Der String muss als XML-Schema-Datum ohne Zeitzone kodiert sein (JJJJ-MM-TT) und ist optional.
- `description` ist ein optionaler Beschreibungstext für diesen Eintrag.

Folgende Fehlercodes können von der Schnittstelle als Antwort geliefert werden:

| Fehlercode | HTTP-Status | Beschreibung |
|---|---|---|
| | 200 | Der Eintrag wurde erfolgreich erstellt/aktualisiert |
| ParticipantIDMissing | 400 | Der Parameter participantid fehlt |
| ParticipantIDInvalid | 400 | Der Parameter participantid ist ungültig |
| DocumentTypeIDMissing | 400 | Der Parameter documenttypeid fehlt |
| DocumentTypeIDInvalid | 400 | Der Parameter documenttypeid ist ungültig |
| ProcessTypeIDMissing | 400 | Der Parameter processtypeid fehlt |
| ProcessTypeIDInvalid | 400 | Der Parameter processtypeid ist ungültig |
| ValidFromInvalid | 400 | Der Parameter validfrom hat ein ungültiges Format |
| ValidToInvalid | 400 | Der Parameter validto hat ein ungültiges Format |
| InternalError | 500 | Bei der Verarbeitung ist ein interner Fehler aufgetreten |
| SMPError | 500 | Beim Aufruf des PEPPOL-SMPs ist ein Fehler aufgetreten |

Tabelle 4: Fehlercodes für Schnittstelle

Die Antwort auf jede Anfrage liefert neben einem HTTP-Statuscode folgende JSON-Struktur zurück:

```
{
  success: boolean
  ( create: boolean )?
  ( errorcode: string )?
  ( errordetails: string )?
}
```

- `success` gibt an, ob die Verarbeitung erfolgreich war oder nicht.
- `create` gibt an, ob der Eintrag neu angelegt wurde (`true`) oder ob ein existierender Eintrag geändert wurde (`false`). Dieses Feld ist nur im Erfolgsfall vorhanden.
- `errorcode` ist ein optionaler String der im Fehlerfall den abstrakten Fehlercode enthält. Dieses Feld ist nur im Fehlerfall vorhanden.
- `errordetails` ist ein optionaler String der im Fehlerfall Details zum Fehler enthalten kann. Dieses Feld kann nur im Fehlerfall vorhanden sein.

#### 5.2.2.2 Löschen eines SMP-Eintrags

Die aufzurufende URL lautet: `/registrydelete`

Die zu verwendende HTTP-Methode ist: `PUT` (`DELETE` ist nicht möglich, da es keine Rückgabewerte zulässt)

Der zu übergebende Parameter ist ein JSON-Objekt mit folgendem Layout:

```
{
  participantid: string
  ( documenttypeid: string )?
}
```

- `participantid` ist die ID des PEPPOL-Teilnehmers und ist verpflichtend.
- `documenttypeid` ist die ID des PEPPOL-Dokumenttyps und ist optional.

Wenn nur der Parameter `participantid` vorhanden ist, so werden alle Einträge des Teilnehmers gelöscht. Wenn der Parameter `documenttypeid` vorhanden ist, so werden nur die passenden Einträge gelöscht.

Folgende Fehlercodes können von der Schnittstelle als Antwort geliefert werden:

| Fehlercode | HTTP-Status | Beschreibung |
|---|---|---|
| | 200 | Der Eintrag wurde erfolgreich gelöscht |
| ParticipantIDMissing | 400 | Der Parameter participantid fehlt |
| ParticipantIDInvalid | 400 | Der Parameter participantid ist ungültig |
| DocumentTypeIDInvalid | 400 | Der Parameter documenttypeid ist ungültig |
| InternalError | 500 | Bei der Verarbeitung ist ein interner Fehler aufgetreten |
| SMPError | 500 | Beim Aufruf des PEPPOL-SMPs ist ein Fehler aufgetreten |

Tabelle 5: Fehlercodes für Schnittstelle

Die Antwort auf jede Anfrage liefert neben einem HTTP-Statuscode folgende JSON-Struktur zurück:

```
{
  success: boolean
  ( delete: boolean )?
  ( errorcode: string )?
  ( errordetails: string )?
}
```

- `success` gibt an, ob die Verarbeitung erfolgreich war oder nicht.

- `delete` gibt an, ob ein Eintrag gelöscht wurde (`true`) oder nicht (`false`). Dieses Feld ist nur im Erfolgsfall vorhanden.
- `errorcode` ist ein optionaler String der im Fehlerfall den abstrakten Fehlercode enthält. Dieses Feld ist nur im Fehlerfall vorhanden.
- `errordetails` ist ein optionaler String der im Fehlerfall Details zum Fehler enthält. Dieses Feld kann nur im Fehlerfall vorhanden sein.

#### *5.2.2.3 Abfrage eines SMP-Eintrags*

Die aufzurufende URL lautet:
`/registryget/{participantid}?documenttypeid={documenttypeid}&processtypeid={processtypeid}`

Die zu verwendende HTTP-Methode ist: `GET`

Die in der URL zu übergebenden Parameter sind:

- `participantid` ist die ID des PEPPOL-Teilnehmers und ist verpflichtend.
- `documenttypeid` ist die ID des PEPPOL-Dokumenttyps und ist optional.
- `processtypeid` ist die ID des PEPPOL-Prozesstyps und ist optional.

Wenn nur der Parameter `participantid` vorhanden ist, so werden alle Einträge des Teilnehmers zurückgeliefert. Wenn der Parameter `documenttypeid` und/oder der Parameter `processtypeid` vorhanden sind, so werden die nur passenden Einträge zurückgeliefert.

Folgende Fehlercodes können von der Schnittstelle als Antwort geliefert werden:

| Fehlercode | HTTP-Status | Beschreibung |
|---|---|---|
| | 200 | Der Eintrag wurde erfolgreich ausgelesen |
| ParticipantIDMissing | 400 | Der Parameter participantid fehlt |
| ParticipantIDInvalid | 400 | Der Parameter participantid ist ungültig |
| DocumentTypeIDInvalid | 400 | Der Parameter documenttypeid ist ungültig |
| ProcessTypeIDInvalid | 400 | Der Parameter processtypeid ist ungültig |
| InternalError | 500 | Bei der Verarbeitung ist ein interner Fehler aufgetreten |
| SMPError | 500 | Beim Aufruf des PEPPOL-SMPs ist ein Fehler aufgetreten |

Tabelle 6: Fehlercodes für Schnittstelle

Die Antwort auf jede Anfrage liefert neben einem HTTP-Statuscode folgende JSON-Struktur zurück:

```
{
  success: boolean
  ( entries: {
      participantid: string
      documenttypeid: string
      processtypeid: string
      ( validfrom: string )?
      ( validto: string )?
      ( description: string )?
    }*
```

```
  )?
  ( errorcode: string )?
  ( errordetails: string )?
}
```

- `success` gibt an, ob die Verarbeitung erfolgreich war oder nicht.
- `entries` enthält die Liste der Einträge die den Abfragekriterien entsprechen. Die Liste kann 0-n Einträge enthalten und ist nur im Erfolgsfall vorhanden. Die einzelnen Felder jedes Objekts haben folgende Bedeutung:
  - `participantid`: die PEPPOL-Teilnehmer ID (wie in der Anfrage angegeben)
  - `documenttypeid`: die PEPPOL-Dokumenttyp ID
  - `processtypeid`: die PEPPOL-Prozesstyp ID
  - `validfrom`: das optionale Datum ab wann der Eintrag gültig ist. Wenn es vorhanden ist, ist es als XML-Schema-Datum ohne Zeitzone formatiert (JJJJ-MM-TT).
  - `validto`: das optionale Datum bis wann der Eintrag gültig ist. Wenn es vorhanden ist, ist es als XML-Schema-Datum ohne Zeitzone formatiert (JJJJ-MM-TT).
  - `description`: die optionale Beschreibung die beim Anlegen angegeben wurde.
- `errorcode` ist ein optionaler String der im Fehlerfall den abstrakten Fehlercode enthält. Dieses Feld ist nur im Fehlerfall vorhanden.
- `errordetails` ist ein optionaler String der im Fehlerfall Details zum Fehler enthält. Dieses Feld kann nur im Fehlerfall vorhanden sein.

#### *5.2.2.4 PEPPOL SMP Installation*

Um einen PEPPOL SMP-Server aufzusetzen müssen folgende Schritte durchgeführt werden. Eine detaillierte Beschreibung in englischer Sprache befindet sich in (Helger 2012b).

1. Systemseitig müssen mindestens Java ≥ 1.6, Apache Tomcat ≥ 6.x und MySQL ≥ 5.1 installiert sein.
2. Als erstes muss die Datenbank konfiguriert werden. Dazu muss die Datei https://joinup.ec.europa.eu/svn/cipaedelivery/tags/1.1.3/cipa-smp-server-library/src/etc/database_backups/smp_20110222.sql (Stand 25.9.2013) aus CIPA e-Delivery heruntergeladen werden. In Zeile 112 befinden sich der Name und das Passwort des SMP-Standardbenutzers („peppol_user“ und „Test1234“) – das ist nicht der Datenbankbenutzer. Die mit diesem Script erzeugte Datenbank hat den Namen „smp“ und verwendet UTF-8 zur Zeichenkodierung. Anschließend muss sichergestellt werden, dass ein Datenbankbenutzer Lese- und Schreibrechte auf die neu erstellte Datenbank gegeben werden. Dazu kann entweder ein neuer Benutzer angelegt werden, oder einem existierenden Benutzer diese Rechte zugewiesen werden.
3. Im nächsten Schritt muss die Datei „cipa-smp-full-webapp-1.1.3.war“ aus CIPA e-Delivery heruntergeladen und entpackt werden (https://joinup.ec.europa.eu/nexus/content/repositories/releases/eu/europa/ec/cipa/cipa-smp-full-webapp/1.1.3/cipa-smp-full-webapp-1.1.3.war - Stand 25.9.2013). Der Inhalt des entpackten Ordners entspricht der zu veröffentlichenden Web-Applikation.
4. Laut Spezifikation muss der SMP im Wurzelverzeichnis und auf Port 80 des Webservers auf Anfragen warten. Daher muss der Inhalt des „webapp“-Ordners als „ROOT“ Web-

Applikation im Tomcat zur Verfügung gestellt werden. Bevor der Server gestartet werden kann, muss noch das Zertifikat eingespielt und die SMP-Konfigurationsdatei bearbeitet werden. Damit der SMP auf dem Standard-http-Port 80 laufen kann, muss entweder die „server.xml“ von Tomcat modifiziert werden, oder es muss ein anderer Webserver als Reverse Proxy vor den Tomcat geschaltet werden, der die Anfragen an die SMP-Applikation weiterleitet.

5. Die Zertifikatsdatei sollte am besten unterhalb des Ordners „WEB-INF/classes/“ eingespielt werden. Wenn dort ein Ordner „keystore“ angelegt wird, so ist in der SMP-Konfigurationsdatei am wenigsten zu ändern.
6. In der zentralen Konfigurationsdatei „WEB-INF/classes/config.properties“, kann u.a. die Anbindung zum SML, der Pfad zu den Zertifikaten und die Datenquelle konfiguriert werden. Eine genaue Beschreibung der einzelnen Punkte befindet sich in (Helger 2012b).
7. Abschließend kann der Tomcat gestartet werden. Nach Eingabe der URL http://localhost im Browser erscheint folgende Seite (durch eine interne automatische Weiterleitung auf http://localhost/web/index.html):

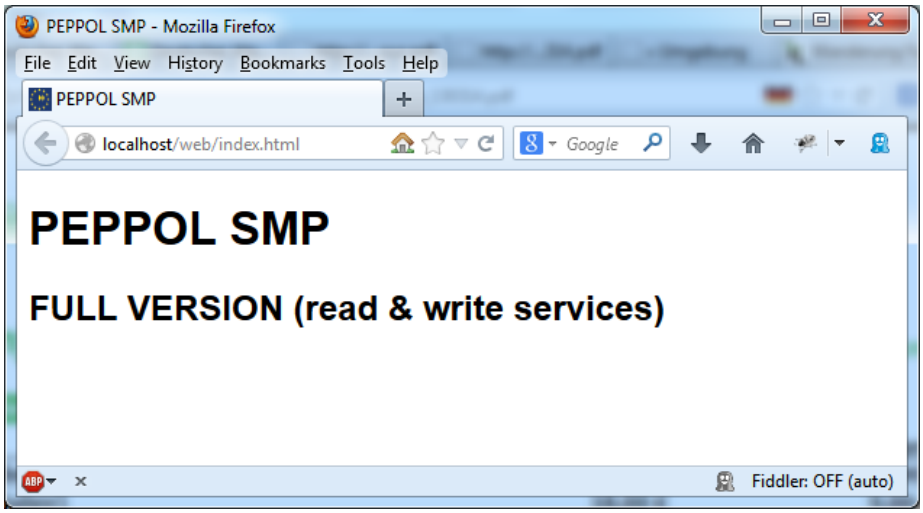


Abbildung 26: Screenshot der SMP Startseite

#### 5.2.2.5 Weiterleitung der Daten an PEPPOL

Die von der Schnittstelle empfangenen Daten werden direkt an den PEPPOL-SMP weitergeleitet. In der SMP-Spezifikation sind die nur-lesenden Operationen ausspezifiziert, es gibt aber keine Aussagen über das Schnittstellen-Layout der schreibenden Operationen, da diese nicht öffentlich zugänglich sein müssen (siehe Kapitel 4.4.5.3). Trotzdem sind in der CIPA e-Delivery Implementierung (siehe Kapitel 5.4.1) schreibende Operationen umgesetzt, die von der ERPOL-Schnittstelle aufgerufen werden. Die schreibenden SMP-Funktionen sind - wie die lesenden Funktionen - als REST-Funktionen umgesetzt. Zur Authentifizierung müssen Benutzername und Passwort in der HTTP-Basic-Auth Kodierung (Benutzername und Passwort Base64-kodiert) per HTTP-Header übergeben werden (Helger 2012b).

#### 5.2.2.6 Technische Details

Die Registry-Schnittstelle bietet ein REST-Interface an. Als REST-Bibliothek kommt Jersey 2.3.1 (http://jersey.java.net/ - Stand 30.10.2013) serverseitig und clientseitig (zum Testen) zum Einsatz.

Während der Entwicklung kann eine lokale Installation des SMPs verwendet werden. Dazu muss die Anbindung an den PEPPOL-SML deaktiviert werden, um keine DNS-Einträge zu erstellen. Als Voraussetzung muss eine MySQL-Datenbank eingerichtet werden, in der die Einträge gespeichert werden. Eine Abfrage der erstellten Einträge per DNS ist daher nicht möglich, während der Entwicklung aber auch nicht notwendig (Helger 2012b).

## 5.3 Dokumentenaustausch-Schnittstelle

Zum Austausch von Dokumenten zwischen ERPEL und PEPPOL ist eine bidirektionale Schnittstelle notwendig, um sowohl Dokumente von ERPEL über PEPPOL zu versenden, als auch um Dokumente, die über PEPPOL einlangen, nach ERPEL zu transferieren.

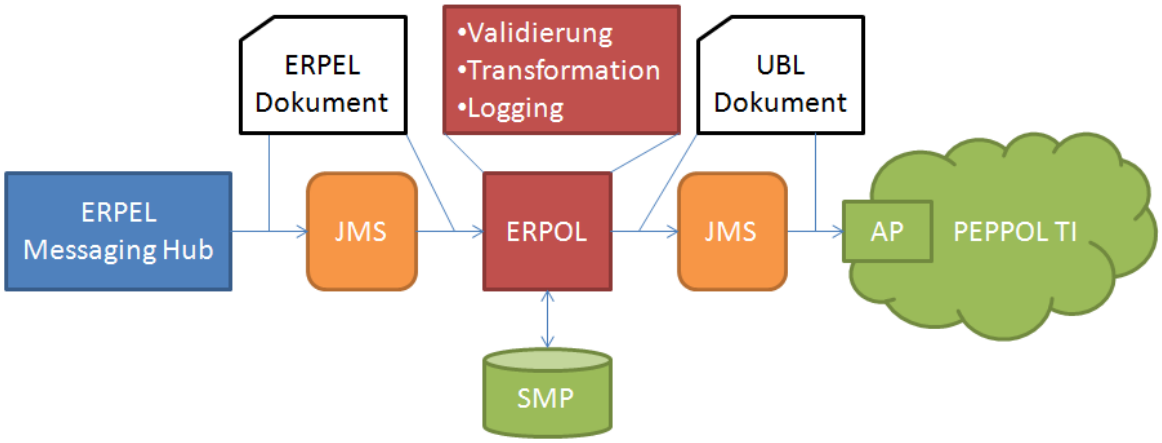


Abbildung 27: Dokumentenaustausch von ERPEL nach PEPPOL

Abbildung 27 zeigt den schematischen Ablauf einer Dokumentenübertragung von ERPEL nach PEPPOL. Dazu stellt der ERPEL-Messaging Hub das zu übermittelnde Dokument in eine Java Message Service (JMS)-Queue. ERPOL holt sich das Dokument aus der Queue, transformiert es nach UBL, sucht die Endpunktadresse des Empfängers in der SMP, stellt es samt der Metadaten in eine andere JMS-Queue und von dort wird es dann über die PEPPOL-Transport Infrastruktur gesendet.

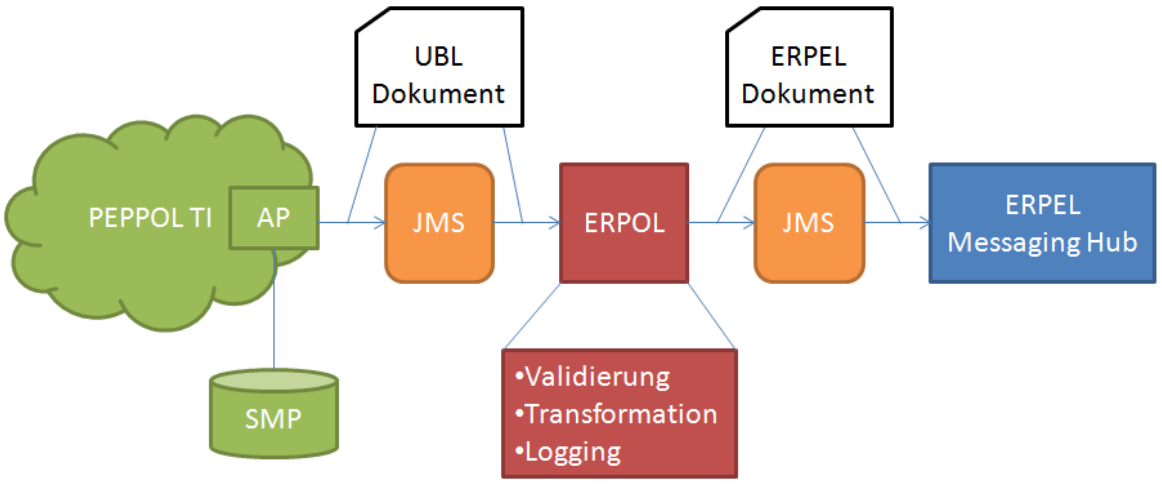


Abbildung 28: Dokumentenaustausch von PEPPOL nach ERPEL

Abbildung 28 zeigt den umgekehrten Weg von PEPPOL nach ERPEL. Als impliziten Schritt beim Empfang eines Dokuments wird im Access Point der eigene SMP gefragt, ob der Empfang des eingehenden Dokuments überhaupt erlaubt ist, da es sein kann, dass der PEPPOL Sender veraltete Daten aus seinem Cache genommen hat. Eingehende UBL-Dokumente werden vom Access Point in eine JMS-Queue gestellt, von ERPOL abgeholt, in das ERPEL-Dokumentenformat transformiert und in eine andere JMS-Queue für ERPEL zur Abholung bereitgestellt. Der ERPEL-Messaging Hub holt sich die Dokumente aus der Queue und liefert sie an den passenden Empfänger.

### 5.3.1 Middleware

Das Java Message Service (JMS) ist ein API zum Nachrichtenaustausch zwischen Applikationen. Zu den Features zählen u.a. synchroner und asynchroner Datenaustausch, once-and-only-once Datenaustausch, die Möglichkeit zur Datenpersistenz und vieles mehr. Diese Eigenschaften machen JMS sehr beliebt zum sicheren und effizienten Datenaustausch zwischen Applikationen (Snyder et al. 2011).

Im Rahmen von ERPOL wird ein JMS-Provider als Middleware zwischen den ERPEL Messaging Hub, ERPOL und den PEPPOL Access Point geschaltet. Das Ziel ist einerseits die möglichst lose Kopplung der Komponenten und andererseits dient die Middleware auch zur Vermeidung von Datenverlust. Wenn die Empfängerseite einer JMS-Queue nicht verfügbar ist – z.B. auf Grund eines Wartungsfensters – geht keine Nachricht verloren, sondern sie kann in einer Datenbank beim JMS Provider gespeichert werden (falls die Nachricht als persistent markiert ist).

Aus technischen Gründen müssen auch aus PEPPOL eingehende Dokumente in einer separaten JMS-Queue gespeichert werden, da der PEPPOL Access Point in der Standard-Ausführung bereits eine vollständige Web-Applikation ist. Das Speichern in der JMS-Queue wurde als eigener "START Receiver" (als Access Point Plugin) umgesetzt. ERPOL nimmt die Dokumente aus der Queue und startet die Verarbeitung. Für die Kommunikation von ERPEL nach PEPPOL ist diese JMS-Queue technisch nicht notwendig, da der Access Point Client als Bibliothek zur Verfügung steht, und somit einfach in andere Applikationen eingebunden werden kann, aber für eine bessere Testbarkeit wird auch hier eine JMS-Queue verwendet.


Abbildung 29: JMS Provider innerhalb der ERPOL-Kommunikation

Abbildung 29 zeigt die Verwendung des JMS-Providers innerhalb der Schnittstelle. Dabei handelt es sich in allen Fällen um ein und dieselbe Instanz des JMS-Providers, es werden aber unterschiedliche Queues verwendet. Queues sind Punkt-zu-Punkt-Kommunikationskanäle, die von einem JMS-Provider zur Verfügung gestellt werden. Sie werden über Namen angelegt, befüllt und abgefragt. Für ERPOL sind folgende Queues notwendig:

- ERPEL zu PEPPOL

a. **FROM_ERPEL_INBOX** – für Nachrichten die von ERPEL nach PEPPOL versendet werden. In diese Queue muss der ERPEL Messaging Hub zu übertragende Nachrichten stellen.
b. **FROM_ERPEL_ERROR** – für Nachrichten die von ERPOL abgelehnt werden (z.B. auf Grund von Validierungs- oder Transformationsfehlern). Der ERPEL Messaging Hub muss diese Queue auslesen, um den Übertragungsstatus eines Dokuments zu ermitteln.
c. **FROM_ERPEL_SUCCESS** – für Nachrichten die erfolgreich über die PEPPOL Transport Infrastruktur versendet wurden. Der ERPEL Messaging Hub muss diese Queue auslesen, um den Übertragungsstatus eines Dokuments zu ermitteln.
d. **TO_PEPPOL_INBOX** – für Nachrichten die von ERPOL erfolgreich bearbeitet wurden und über die PEPPOL Transport Infrastruktur versendet werden sollen.
e. **TO_PEPPOL_RESPONSE** – für Nachrichten, die das Übertragungsergebnis der PEPPOL Transport Infrastruktur enthalten. Diese muss von ERPOL ausgelesen und das Ergebnis entweder in die Queue *FROM_ERPEL_ERROR* oder in die Queue *FROM_ERPEL_SUCCESS* gestellt werden.

- PEPPOL zu ERPEL
  a. **FROM_PEPPOL_INBOX** - für eingehende UBL-Dokumente, die über die PEPPOL Transport-Infrastruktur übertragen werden und nach ERPEL übermittelt werden sollen. In diese Queue muss der PEPPOL Access Point eingehende Nachrichten stellen.
  b. **FROM_PEPPOL_RESPONSE** – für die Ergebnisse der Verarbeitung von eingehenden PEPPOL-Dokumenten in ERPOL. Der PEPPOL Access Point muss diese Queue auslesen um den Übertragungsstatus eines Dokuments zu ermitteln.
  c. **TO_ERPEL_INBOX** – für Nachrichten, die von ERPOL nach ERPEL übermittelt werden. Der ERPEL Messaging Hub muss diese Queue auslesen um Dokumente an den endgültigen Empfänger weiterzuleiten.

In alle Queues werden XML-Nachrichten eingebracht. Alle Queues sind persistent, d.h. Nachrichten werden erst in einer Datenbank beim JMS Provider abgelegt, bevor sie von der Gegenstelle abgeholt werden können. Da JMS XML-Objekte nicht nativ unterstützt, müssen diese vorher in die textuelle Form überführt werden, und können dann als JMS-TextMessage oder als JMS-BytesMessage übertragen werden. Aus Performancegründen wird die Übertragung als UTF-8-kodierter String in einer BytesMessage durchgeführt, da diese den geringsten Overhead beim Serialisieren und Deserialisieren hat.

### 5.3.2 Dokumententransformation

Einer der anspruchsvollsten Aspekte von ERPOL ist die Dokumententransformation vom ERPEL-Dokumenttyp nach UBL und zurück, da PEPPOL nur das Dokumentenformat UBL unterstützt. Das Standardwerkzeug für die Transformation zwischen verschiedenen XML-Dokumenttypen ist die Extensible Stylesheet Language Transformation (XSLT). Dabei handelt es sich um einen XML-Dialekt, mit dessen Hilfe XML-Dokumente in XML-, HTML- oder Textdokumente überführt werden können. Der größte Vorteil von XSLT-Skripten ist, dass sie plattformunabhängig einsetzbar sind und dadurch nicht an eine spezifische Programmiersprache gebunden sind. Die Nachteile von XSLT sind u.a. die Laufzeitperformance, die schlechte Wartbarkeit bei komplexen

Mappings und die hohe Komplexität, wenn Elemente nicht linear transformiert werden können. Auf Grund der genannten Nachteile wurde die Transformation als nativer Java-Code umgesetzt. Dadurch kann hohe Performance und Flexibilität genauso gewährleistet werden, wie eine hohe Plattformunabhängigkeit.

Zum Lesen und Schreiben der XML-Dokumente wird die „Java Architecture for XML Binding“ (JAXB) in der Version 2.1[4] eingesetzt. Dadurch können XML-Dokumente XML-Schema-konform gelesen und geschrieben werden. Die Befüllung der XML-Dokumente erfolgt über reguläre Java-Objekte (Plain Old Java Objects – POJOs), die basierend auf dem XML-Schema mit Hilfe des Maven-Plugins „org.jvnet.jaxb2.maven2::maven-jaxb2-plugin“ automatisch generiert wurden.

Folgende Transformationen werden in ERPOL umgesetzt:

- ERPEL-Invoice → PEPPOL-Invoice
- PEPPOL-Invoice → ERPEL-Invoice
- ERPEL-Order → PEPPOL-Order
- PEPPOL-Order → ERPEL-Order

Da sich viele Konstrukte innerhalb der ERPEL-Dokumenttypen und der PEPPOL-Dokumenttypen wiederholen, wurden die Transformationen der gemeinsamen Elemente (sowohl von ERPEL nach PEPPOL als auch vice versa) zentralisiert.

Bevor eine Transformation beginnt, wird das Quelldokument gegen das Quell-XML-Schema und die entsprechenden Quell-Schematron-Regeln geprüft, sodass nur wohlgeformte und valide XML-Dokumente überhaupt transformiert werden können. Nach der eigentlichen Transformation wird das Zieldokument ebenfalls gegen das Ziel-XML-Schema geprüft, sodass potentiell ungültige Dokumente möglichst frühzeitig erkannt werden. Zusätzlich zur XML-Schema-Prüfung des erzeugten Dokuments wird auch noch eine Schematron-Prüfung für ERPEL- und PEPPOL-Dokumente angewendet, damit sichergestellt ist, dass die transformierten Dokumente wirklich valide sind. Falls ein Dokument nicht transformierbar ist, so wird eine entsprechende Fehlermeldung ausgegeben, und der Dokumentenaustausch findet nicht statt. Folgende Fehler können während der Transformation ERPEL nach UBL auftreten:

| Fehler-ID | Beschreibung |
|---|---|
| INVALID_COUNTRY_CODE | Es konnte kein gütiger Ländercode gefunden werden. |
| INVALID_CURRENCY | Die Währung konnte nicht in das Zielformat überführt werden. |
| REDUCTION_SURCHARGE_NOT_FOUND | Es wurde kein Auf-/Abschlag gefunden, der dem aktuellen `BaseAmount` entspricht. |
| DUPLICATE_PAYMENT_MEANS | Bei einer Bestellung darf nur eine Zahlungsart angegeben werden. |

Tabelle 7: Fehlercodes bei der Transformation ERPEL nach UBL

Außerdem können auch noch Fehler bei der PEPPOL-Schematron-Validierung auftreten, die aber hier nicht taxativ aufgezählt werden, weil sie je nach Kontext unterschiedlich sein können.

[4] Version 2.1 ist standardmäßig in allen JRE Versionen ab 1.6 enthalten, während für die neuere Version 2.2 eine manuelle Adaption jeder lokalen JRE notwendig wäre.

Bei der umgekehrten Transformation von UBL nach ERPEL können folgende Fehler auftreten:

| Fehler-ID | Beschreibung |
|---|---|
| NO_ADDRESS_NAME_PROVIDED | Einer Adresse fehlt der Name. |
| NO_ADDRESS_TOWN_PROVIDED | Einer Adresse fehlt die Stadt. |
| NO_ADDRESS_ZIP_PROVIDED | Einer Adresse fehlt die PLZ. |
| NO_ADDRESS_COUNTRY_PROVIDED | Einer Adresse fehlt der Ländercode. |
| NO_CONTACT_NAME_PROVIDED | Einer Kontaktperson fehlt der Name der Person. |
| INVALID_COUNTRY_CODE | Es konnte kein gütiger Ländercode gefunden werden. |
| NO_CUSTOMER_VATIN_PROVIDED | Es fehlt die UID-Nummer des Kunden. |
| INVALID_ID | Ein notwendiger Wert ist zu lang. |
| NO_SUPPLIER_VATIN_PROVIDED | Es fehlt die UID-Nummer des Lieferanten. |
| NO_ITEM_DESCRIPTION | Einer Zeile fehlt der Positionstext. |
| NO_ARTICLE_NUMBER_PROVIDED | Einer Zeile fehlt die Artikelnummer |

Tabelle 8: Fehlercodes bei der Transformation UBL nach ERPEL

Zusätzlich zu den Fehlercodes wird je nach Kontext auch eine entsprechende Fehlermeldung der darunterliegenden Software-Komponente mitgeliefert. Bei dieser Transformation können aufgrund des strikteren ERPEL-XSDs weit mehr Fehler auftreten als in die andere Richtung.

### 5.3.3 ERPEL zu PEPPOL

Wenn der ERPEL Messaging Hub eine Nachricht verarbeitet, die für das PEPPOL-Netzwerk gedacht ist, so muss das zu übertragende Dokument samt einiger Metadaten in die JMS-Queue *FROM_ERPEL_INBOX* gestellt werden. Ein PEPPOL Access Point benötigt folgende Metadaten:

- Die Webservice Endpunkt Adresse des Empfänger Access Points
- Message ID – die eindeutige, interne Nachrichten ID zur Nachverfolgung. Das kann z.B. eine UUID sein.
- Sender ID – der PEPPOL-Identifier des Senders
- Recipient ID – der PEPPOL-Identifier des Empfängers
- Document Type ID – die ID des PEPPOL Dokumenttyps
- Process ID – die ID des PEPPOL Prozesstyps

Zusätzlich gibt es in PEPPOL noch das Metadatenfeld Channel ID, das den verwendeten Kanal angibt, welches für dieses Szenario aber irrelevant ist.

ERPEL benötigt beim Senden als Metadaten zur Identifikation des übertragenen Dokuments nur eine Message ID. Da es dafür Sorge trägt, dass die Message ID eindeutig ist, kann die ERPEL-Message ID direkt als PEPPOL-Message ID wiederverwendet werden, wodurch auf ein internes Mapping der Message-IDs zwischen ERPEL und PEPPOL verzichtet werden kann.

In Kapitel 5.2 wurde bereits erklärt, dass derzeit nur Bestellungen und Rechnungen übertragen werden können. Im Prinzip wäre es ausreichend, wenn außer dem ERPEL-Dokument nur noch die PEPPOL-Identifier von Sender und Empfänger als Metadaten mitgegeben würden, da sich sowohl der Dokumenttyp als auch der Prozesstyp aus dem ERPEL-Dokument ableiten lassen. Um aber flexibel auf zukünftige Erweiterungen reagieren zu können, müssen beide Felder vom ERPEL-Messaging Hub explizit als Metadaten übergeben werden.

#### 5.3.3.1 Dokumentenaustausch

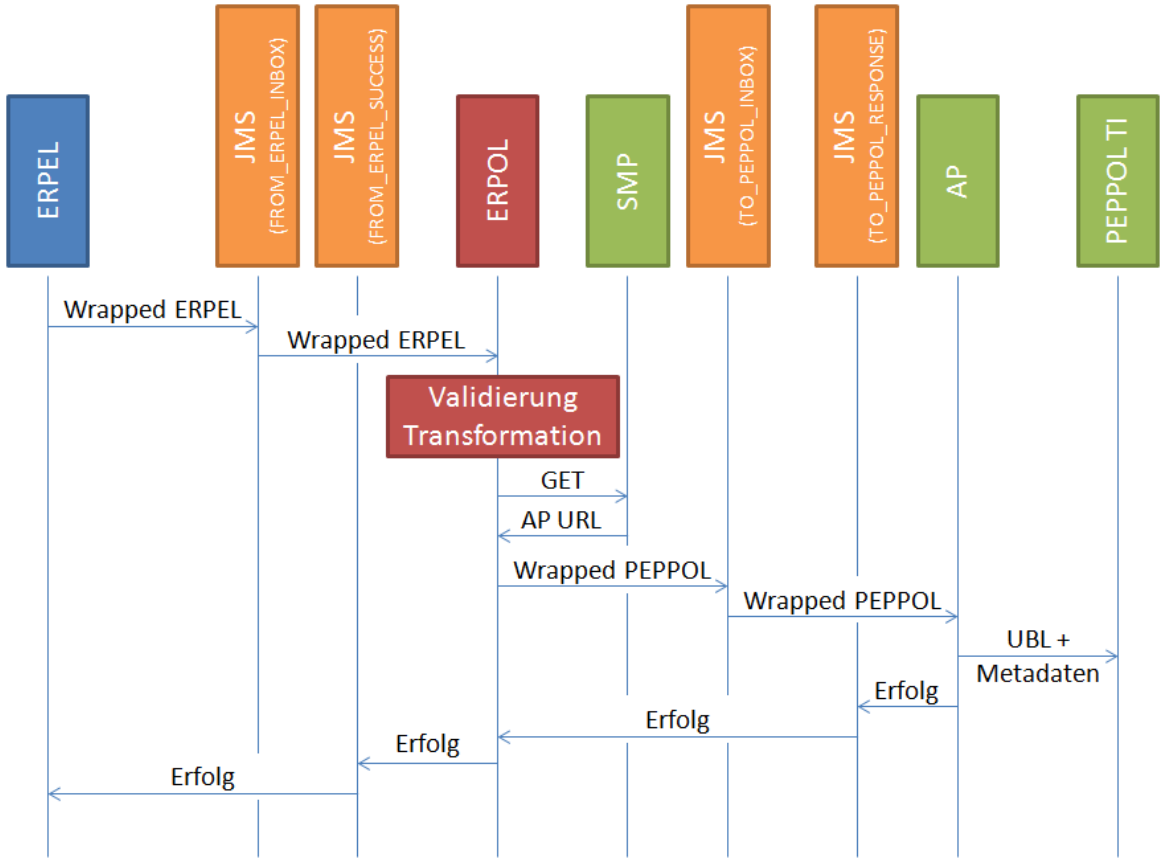


Abbildung 30: Sequenzdiagramm ERPEL → PEPPOL im Erfolgsfall

Abbildung 30 zeigt den detaillierten Ablauf einer Nachrichtenübermittlung von ERPEL nach PEPPOL im Erfolgsfall. ERPEL stellt das zu übermittelnde WrappedErpel Dokument (siehe Kapitel 5.3.3.2.1) in die JMS-Queue *FROM_ERPEL_INBOX*, ERPOL holt es von dort ab, validiert es und transformiert das ERPEL-Dokument in das UBL-Format (die XML-Schema- sowie die Schematron-Prüfungen sind Teil der Transformation - siehe Kapitel 5.3.2). Anschließend wird die Webservice Endpunkt Adresse des Empfängers durch einen SMP-Lookup ermittelt und zusammen mit den anderen Metadaten in die JMS-Queue *TO_PEPPOL_INBOX* zur Übermittlung via PEPPOL gestellt. Der Sender Access Point holt sich das Dokument aus der JMS-Queue und sendet es über die PEPPOL Transport Infrastruktur und stellt die Übertragungsbestätigung in die JMS-Queue *TO_PEPPOL_RESPONSE*. Diese Bestätigung wird von ERPOL übernommen und im passenden Format in die JMS-Queue *FROM_ERPEL_SUCCESS* gestellt, von wo sich der ERPEL Messaging Hub die Antwort abholen kann.

Im Vergleich zur Dokumentenaustauschschnittstelle in die andere Richtung (PEPPOL zu ERPEL) kann der ERPEL Messaging Hub asynchron auf das Ergebnis warten und die Antwort somit zu einem späteren Zeitpunkt verarbeiten.

Folgende Fehlerklassen und Fehler können während dieses Prozesses auftreten:

| Fehlerklasse | Fehler-ID | Beschreibung |
|---|---|---|
| WRAPPED_ERPEL | SOURCE_INVALID_XML | Im WrappedErpel-Dokument ist kein Erpel-Dokument enthalten. |

| | | |
|---|---|---|
| WRAPPED_ERPEL | METADATA_ERROR | Mindestens eines der Metadatenfelder ist syntaktisch inkorrekt. |
| TRANSFORMATION | Siehe Kapitel 5.3.2 | |
| TRANSFORMATION | TRANSFORMATION_FAILED | Ein unspezifizierter Fehler ist aufgetreten. Das erzeugte Dokument entspricht nicht dem XML-Schema. |
| ERPOL | WRAPPED_PEPPOL_INVALID | Das erzeugte WrappedPeppol-Dokument entspricht nicht dem XML-Schema. |
| ERPOL | DELIVERY_ERROR_TO_PEPPOL | Beim Einstellen in die JMS-Queue *TO_PEPPOL_INBOX* ist ein Fehler aufgetreten. |
| ERPOL | E2P_ERROR_INVALID | Das erzeugte Erpel2PeppolError-Dokument entspricht nicht dem XML-Schema. |
| ERPOL | DELIVERY_ERROR_FROM_ERPEL_ERROR | Beim Einstellen in die JMS-Queue *FROM_ERPEL_ERROR* ist ein Fehler aufgetreten. |
| ERPOL | EXCEPTION | Ein unbekannter Fehler ist aufgetreten. |

Tabelle 9: Fehler bei der Dokumentenverarbeitung von ERPEL nach PEPPOL

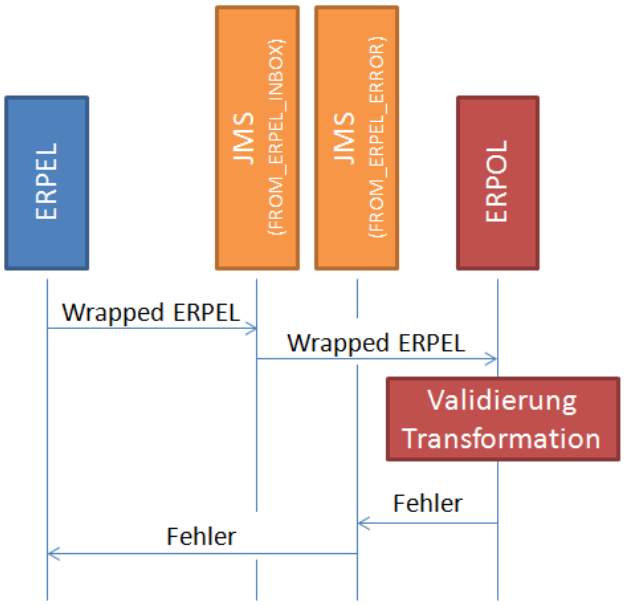


Abbildung 31: Sequenzdiagramm ERPEL → PEPPOL im Fehlerfall

Abbildung 31 zeigt den Ablauf wenn bei der Transformation in ERPOL ein Fehler auftritt. Im Vergleich zum Erfolgsfall sieht der PEPPOL Access Point das Dokument nie. Außerdem landet die Fehlernachricht von ERPOL in einer anderen JMS-Queue (*FROM_ERPEL_ERROR*) als im Erfolgsfall. Für andere Fehlerfälle muss der Ablauf entsprechend angepasst werden.

#### *5.3.3.2 Dokumententypen*

An dieser Stelle werden alle Dokumententypen deklariert die im Zuge des Dokumentenaustauschs von ERPEL nach PEPPOL benötigt werden.

##### 5.3.3.2.1 WrappedErpel

Um die Metadaten und das ERPEL-Dokument gemeinsam an ERPOL übergeben zu können, müssen diese in eine einzige XML-Datei zusammengefügt werden. Dazu wurde das folgende XML-Schema definiert:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns="http://www.helger.com/ns/wrapped-erpel/1.0"
           targetNamespace="http://www.helger.com/ns/wrapped-erpel/1.0"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified"
           xmlns:erpel="http://erpel.at/schemas/1p0/documents">
  <xs:import namespace="http://erpel.at/schemas/1p0/documents"
             schemaLocation="http://erpel.at/schemas/1p0/documents" />
  <xs:complexType name="KeyValuePairType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="key" type="xs:string" use="required" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="WrappedErpelType">
    <xs:sequence>
      <xs:element name="KeyValuePair" type="KeyValuePairType" minOccurs="0"
maxOccurs="unbounded" />
      <xs:element name="MessageID" type="xs:string" />
      <xs:element name="SenderID" type="xs:string" />
      <xs:element name="RecipientID" type="xs:string" />
      <xs:element name="DocumentTypeID" type="xs:string" />
      <xs:element name="ProcessID" type="xs:string" />
      <xs:element name="Document" type="erpel:DocumentType" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="WrappedErpel" type="WrappedErpelType" />
</xs:schema>
```

Die hier gezeigte Version des XML-Schemas ist eine um Kommentare und Anmerkungen gekürzte Version, um den Blick auf die wesentlichen Dinge zu richten. Für alle benötigten Metadaten gibt es ein eigenes Element und das ERPEL-Dokument folgt direkt nach den Metadaten als typisiertes Element. Das Wurzelelement des Schemas hat den Tag-Namen „WrappedErpel“, weshalb Dokumente dieses Typs im Folgenden auch als „WrappedErpel-Dokumente“ bezeichnet werden.

##### 5.3.3.2.2 WrappedPeppol

Um ein UBL-Dokument und alle PEPPOL Metadaten inklusive der Webservice Endpunkt Adresse über die PEPPOL Transport Infrastruktur zu versenden wurde folgendes XML-Schema definiert:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns="http://www.helger.com/ns/wrapped-peppol/1.0"
           targetNamespace="http://www.helger.com/ns/wrapped-peppol/1.0"
           elementFormDefault="qualified"
```

```
           attributeFormDefault="unqualified">
  <xs:complexType name="KeyValuePairType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="key" type="xs:string" use="required" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="WrappedPeppolType">
    <xs:sequence>
      <xs:element name="KeyValuePair" type="KeyValuePairType" minOccurs="0"
maxOccurs="unbounded" />
      <xs:element name="EndpointURL" type="xs:anyURI" minOccurs="0" />
      <xs:element name="MessageID" type="xs:string" />
      <xs:element name="ChannelID" type="xs:string" minOccurs="0" />
      <xs:element name="SenderID" type="xs:string" />
      <xs:element name="RecipientID" type="xs:string" />
      <xs:element name="DocumentTypeID" type="xs:string" />
      <xs:element name="ProcessID" type="xs:string" />
      <xs:any processContents="skip" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="WrappedPeppol" type="WrappedPeppolType" />
</xs:schema>
```

Dieses XML-Schema ist dem WrappedErpel XML-Schema sehr ähnlich, hat aber zusätzlich noch ein `EndpointURL`-Element und das UBL-Dokument ist in diesem Fall als `xs:any` definiert, da UBL-Invoice und UBL-Order unterschiedliche XML-Namespaces verwenden, aber im gleichen Dokumentenformat übermittelt werden sollen. Außerdem soll das Format offen für zukünftige Dokumententypen sein, und nicht auf die ERPOL benötigten Typen beschränkt sein.

Das `EndpointURL`-Element ist optional, da es für den in Kapitel 5.3.3.1 beschriebenen Dokumentenaustausch von ERPEL nach PEPPOL verpflichtend ist, während es für den in Kapitel 5.3.4.2 beschriebenen Austausch von PEPPOL nach ERPEL nicht benötigt wird.

#### 5.3.3.2.3 PeppolSenderResponse

Das Ergebnis des Versenden eines Dokuments durch den PEPPOL AccessPoint Client wird in folgendem XML-Schema definiert:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns="http://www.helger.com/ns/peppol-sender-response/1.0"
           targetNamespace="http://www.helger.com/ns/peppol-sender-
response/1.0"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
  <xs:complexType name="PeppolSenderErrorType">
    <xs:sequence>
      <xs:element name="Text" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="PeppolSenderResponseType">
    <xs:sequence>
      <xs:element name="MessageID" type="xs:string" minOccurs="0" />
      <xs:element name="ErrorMessage" type="PeppolSenderErrorType"
minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
```

```
    <xs:attribute name="Success" type="xs:boolean" use="required" />
  </xs:complexType>
  <xs:element name="PeppolSenderResponse" type="PeppolSenderResponseType" />
</xs:schema>
```

Das `Success`-Attribut gibt an, ob das Versenden erfolgreich war oder nicht. Im Fehlerfall sollte mindestens ein `ErrorMessage`-Element enthalten sein.

Das `MessageID`-Element ist optional, da unter gewissen Bedingungen auch Fehler auftreten können, bei denen keine Message ID angegeben werden kann (z.B. wenn die Message ID fehlt).

#### 5.3.3.2.4 Erpel2PeppolError

Im Fehlerfall wird eine XML-Nachricht in die JMS-Queue *FROM_ERPEL_ERROR* gestellt, um den ERPEL Messaging Hub über den Fehler zu informieren, sodass dieser wiederum den ursprünglichen Sender darüber informieren kann. Das XML-Schema für den Fehlerfall wurde wie folgt definiert:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns="http://www.helger.com/ns/erpel2peppol-error/1.0"
           targetNamespace="http://www.helger.com/ns/erpel2peppol-error/1.0"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
  <xs:complexType name="Erpel2PeppolSingleErrorType">
    <xs:sequence>
      <xs:element name="ErrorClass" type="xs:string" />
      <xs:element name="ErrorID" type="xs:string" />
      <xs:element name="ErrorMessage" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Erpel2PeppolErrorType">
    <xs:sequence>
      <xs:element name="MessageID" type="xs:string" minOccurs="0" />
      <xs:element name="Error" type="Erpel2PeppolSingleErrorType"
maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Erpel2PeppolError" type="Erpel2PeppolErrorType" />
</xs:schema>
```

Die Message ID wird mit dem Wert aus dem WrappedErpel-Dokument befüllt, damit der EPREL-Messaging Hub sie zu der entsprechenden Quellnachricht zuordnen kann. Im Wurzelelement müssen außerdem 1-n `Error`-Elemente vorhanden sein, die die Details zu den einzelnen Fehlern enthalten. Diese werden von ERPOL wie folgt belegt:

- `ErrorClass` ist die von ERPOL erzeugte Fehlerklasse.
- `ErrorID` enthält die ID des aufgetretenen Fehlers.
- `ErrorMessage` ist eine Beschreibung des Fehlers als menschenlesbarer Text.

Das übertragene Dokument muss in der Antwortnachricht nicht enthalten sein.

#### 5.3.3.2.5 Erpel2PeppolSuccess

Im Erfolgsfall wird ebenfalls eine XML-Nachricht erzeugt, jedoch in die JMS-Queue *FROM_ERPEL_SUCCESS* gestellt. Dies darf zeitlich erst nach dem erfolgreichen Einstellen in die

JMS-Queue *TO_PEPPOL_INBOX* durchgeführt werden. Die Erfolgsnachricht muss folgendem, sehr einfachen, XML-Schema entsprechen:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns="http://www.helger.com/ns/erpel2peppol-success/1.0"
           targetNamespace="http://www.helger.com/ns/erpel2peppol-success/1.0"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
  <xs:complexType name="Erpel2PeppolSuccessType">
    <xs:sequence>
      <xs:element name="MessageID" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="Erpel2PeppolSuccess" type="Erpel2PeppolSuccessType" />
</xs:schema>
```

Das Element `MessageID` enthält wie im Fehlerfall die Message ID aus dem WrappedErpel-Dokument.

#### 5.3.3.3 cipa-start-jmssender

Der *cipa-start-jmssender* ist eine eigenständige Web-Applikation zum Versenden von PEPPOL-Nachrichten über eine PEPPOL AccessPoint Client aus einer JMS-Queue heraus.

Für den *cipa-start-jmssender* gibt es eine eigene Konfigurationsdatei, die im Classpath unter `/config-start-jms-sender.xml` zu finden sein muss. Dort müssen folgende Einstellungen getätigt werden:

- `connection-string` – Der Verbindungsstring mit dem sich *cipa-start-jmsreceiver* zum JMS Server verbindet (kein Standardwert). Zum Testen mit einer lokalen ActiveMQ-Server-Instanz mit der Standardkonfiguration kann `tcp://localhost:61616` verwendet werden.
- `persistent-messaging` – ein Flag das angibt, ob die JMS-Nachrichten im JMS-Provider zwischengespeichert werden sollen oder nicht (Standardwert: `true`)
- `to-peppol-inbox-queue-name` – Name der JMS-Queue für zu Versendende Nachrichten (Standardwert: TO_PEPPOL_INBOX)
- `to-peppol-response-queue-name` – Name der JMS-Queue für die Antwortnachrichten (Standardwert: TO_PEPPOL_RESPONSE)

Das Projekt *cipa-start-jmssender*, das im Rahmen dieser Arbeit entwickelt wurde, steht als Open Source Software unter der „Apache 2“ Lizenz unter https://github.com/phax/cipa-start-jmssender zur Verfügung.

### 5.3.4 PEPPOL zu ERPEL

Für den umgekehrten Weg, also eingehende Nachrichten aus dem PEPPOL-Netzwerk in Richtung von ERPEL, ist eine eigene Spezifikation notwendig. Ein simples Umdrehen des Ablaufs von ERPEL zu PEPPOL funktioniert leider nicht.

Ein Problem beim Empfang eines Dokuments via PEPPOL ist, dass zum Zeitpunkt der Verfügbarkeit der SOAP-Nachricht bereits die technisch korrekte Übertragung an den Sender quittiert wurde. D.h. es gibt keine technisch saubere Möglichkeit ein Dokument abzulehnen –

weil es z.B. kein gültiges UBL-Dokument ist. Die einzige Möglichkeit besteht darin, eine Exception zu werfen, und damit dem Sender zu verständigen, dass das Dokument übertragen, aber nicht angenommen werden konnte. Allerdings widerspricht das der PEPPOL START-Spezifikation, da es dort genau vorgesehene Fälle für Exceptions gibt, jedoch keinen generischen Grund. In manchen PEPPOL BIS gibt es die Möglichkeit einer „Application Level Response" – einer Antwort auf Prozessebene – jedoch nicht in den von ERPOL unterstützten BIS mit nur einer Aktivität (siehe Kapitel 5.1.1). Dieses Problem wurde von PEPPOL schon als solches erkannt, weshalb derzeit in OpenPEPPOL an einer generischen „Message Level Response" – einer Antwort auf technischer Ebene – gearbeitet wird.

Als eingehende Daten stehen neben dem UBL-Dokument auch folgende PEPPOL-Metadaten zur Verfügung:

- Message ID – die PEPPOL-interne Nachrichten ID zur Nachverfolgung. Das kann z.B. eine UUID sein.
- Sender ID – der PEPPOL-Identifier des Senders
- Recipient ID – der PEPPOL-Identifier des Empfängers
- Document Type ID – die ID des PEPPOL Dokumenttyps
- Process ID – die ID des PEPPOL Prozesstyps

#### *5.3.4.1 PEPPOL Access Point*

Der CIPA e-Delivery PEPPOL Access Point ist bereits als fertige Web-Applikation unter https://joinup.ec.europa.eu/nexus/content/repositories/releases/eu/europa/ec/cipa/cipa-start-server/1.1.2/cipa-start-server-1.1.2.war verfügbar. Diese muss entpackt werden, um die Zertifikate zu konfigurieren. Details welche Zertifikate in welchen Konfigurationsdateien eingetragen werden müssen ist in (Helger 2012c) nachzulesen.

Die PEPPOL-Access Point Java-Implementierung bietet ein „Service Provider Interface" (SPI) an, mit dem eingehende Dokumente benutzerdefiniert verarbeitet werden können. Bei SPI handelt es sich um eine Standard-Vorgehensweise in Java, um Komponenten lose zu koppeln („loose coupling"). Über eine bestimmte Datei wird die Verbindung zwischen einem Interface und den passenden Implementierungen geschaffen. Dabei muss der Dateiname dem vollqualifizierten Namen des Interfaces entsprechen, und als Inhalt der Datei kann pro Zeile der Name einer implementierenden Klasse angegeben werden. Diese Klassen werden dann intern über Java Reflection instanziiert und müssen einen öffentlichen („public") Konstruktor ohne Argumente besitzen. Diese SPI-Dateien müssen immer im Verzeichnis `/META-INF/services` liegen, damit sie automatisch ladbar sind.

Der Name des zu implementierenden Interfaces lautet beim CIPA e-Delivery Access Point `eu.europa.ec.cipa.transport.start.server.IAccessPointServiceReceiverSPI`. Standardmäßig wird der *cipa-start-filereceiver* als Beispiel-Implementierung ausgeliefert, der eingehende Dokumente in ein konfigurierbares Verzeichnis auf die Festplatte schreibt. Für ERPOL ist diese Lösung nicht ideal, da das Lesen und Schreiben von Dateien zu ineffizient ist. Stattdessen wird im Rahmen von ERPOL ein *cipa-start-jmsreceiver* implementiert, der eingehende PEPPOL-Nachrichten in eine JMS-Queue zur weiteren Verarbeitung stellt, und synchron auf eine Antwortnachricht über JMS wartet. Dadurch entsteht eine lose Kopplung

zwischen dem Access Point und ERPOL, gleichzeitig ist es aber möglich eine synchrone Verarbeitung im Access Point zu erzielen, sodass Validierungs- und Transformationsfehler direkt per Exception an den PEPPOL-Sender zurückgemeldet werden können. Details zum *cipa-start-jmsreceiver* finden sich in Kapitel 5.3.4.4.

### 5.3.4.2 Dokumentenaustausch

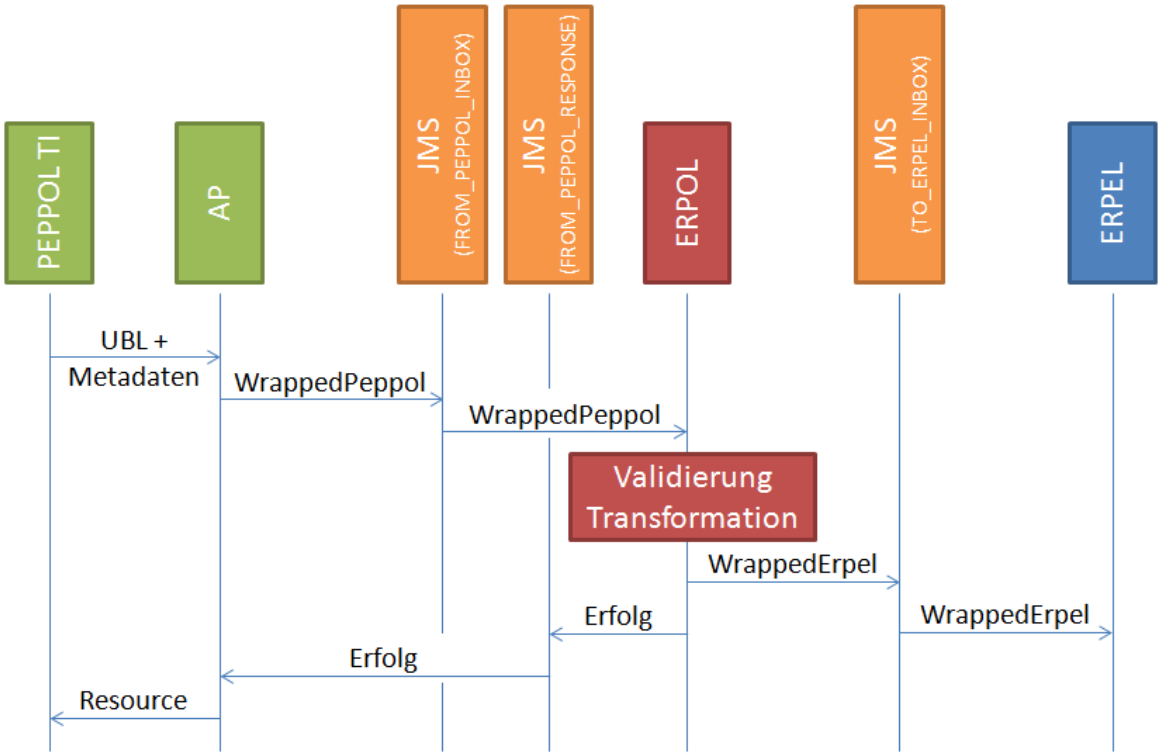


Abbildung 32: Sequenzdiagramm PEPPOL → ERPEL im Erfolgsfall

Abbildung 32 zeigt den detaillierten Ablauf einer erfolgreichen Datenübermittlung von PEPPOL nach ERPEL. Der Access Point stellt das eingehende UBL-Dokument und alle Metadaten in die JMS-Queue *FROM_PEPPOL_INBOX*. ERPOL holt es sich von dort und führt die Transformation des Dokuments vom PEPPOL-UBL-Format in das ERPEL-Format durch (siehe Kapitel 5.3.2). Die ursprünglichen Metadaten und das erzeugte ERPEL-Dokument werden in ein WrappedErpel-Dokument verpackt (siehe Kapitel 5.3.3.2.1) und in die JMS-Queue *TO_ERPEL_INBOX* gestellt. Die Verarbeitungsbestätigung wird von ERPOL in die JMS-Queue *FROM_PEPPOL_RESPONSE* gestellt, von wo aus der PEPPOL Access Point die Antwort entgegen nimmt und an den ursprünglichen Sender zurückmeldet. Wichtig ist, dass alle Schritte synchron ausgeführt werden, damit im Falle von Validierungs- oder Transformationsfehlern eine Exception an den PEPPOL-Sender zurückgesendet werden kann. Laut PEPPOL Transport Infrastructure Agreement (TIA) darf der Austausch einer Nachricht zwischen zwei PEPPOL APs maximal 300 Sekunden (5 Minuten) dauern.

In dieser Schnittstelle sind keine Antwort-JMS-Queues für ERPEL vorgesehen, da, wie bereits erwähnt, PEPPOL keine Möglichkeit bietet asynchron eine Antwort auf technischer Ebene zu senden, und der ERPEL-Messaging Hub die Nachrichten immer asynchron verarbeitet. Sämtliche Fehler, die danach im ERPEL-Messaging Hub auftauchen, werden in eine ERPEL-spezifische Dead

Letter Queue (DLQ) gestellt und müssen „out-of-band“, also bilateral zwischen ursprünglichem Sender und dem endgültigen Empfänger, geklärt werden.

Der ERPEL-Messaging Hub löst den ERPEL-Empfänger über das Metadatenfeld `RecipientID` auf. Es muss dazu einen ERPEL-Teilnehmer geben, der diese PEPPOL-ID als Alias in der ERPEL-Registry besitzt, sonst handelt es sich um eine interne Inkonsistenz und die Funktionsweise der Registry-Schnittstelle (siehe Kapitel 5.2) muss überprüft werden.

Folgende Fehlerarten können während der Abarbeitung eines aus PEPPOL kommenden Dokuments auftreten, und führen zu einer Exception im Access Point:

| Beschreibung |
|---|
| Im cipa-start-jmsreceiver ist ein Fehler aufgetreten. Siehe Kapitel 5.3.4.4 |
| Im WrappedPeppol-Dokument ist kein gültiges UBL-Dokument enthalten. |
| Mindestens eines der WrappedPeppol-Metadatenfelder ist inkorrekt oder fehlt. |
| Bei der Transformation von UBL nach ERPEL ist ein Fehler aufgetreten. Siehe Kapitel 5.3.2 |
| Beim Einstellen in die JMS-Queue *TO_ERPEL_INBOX* ist ein Fehler aufgetreten. |

Tabelle 10: Fehlerarten bei der Dokumentenverarbeitung von PEPPOL nach ERPEL

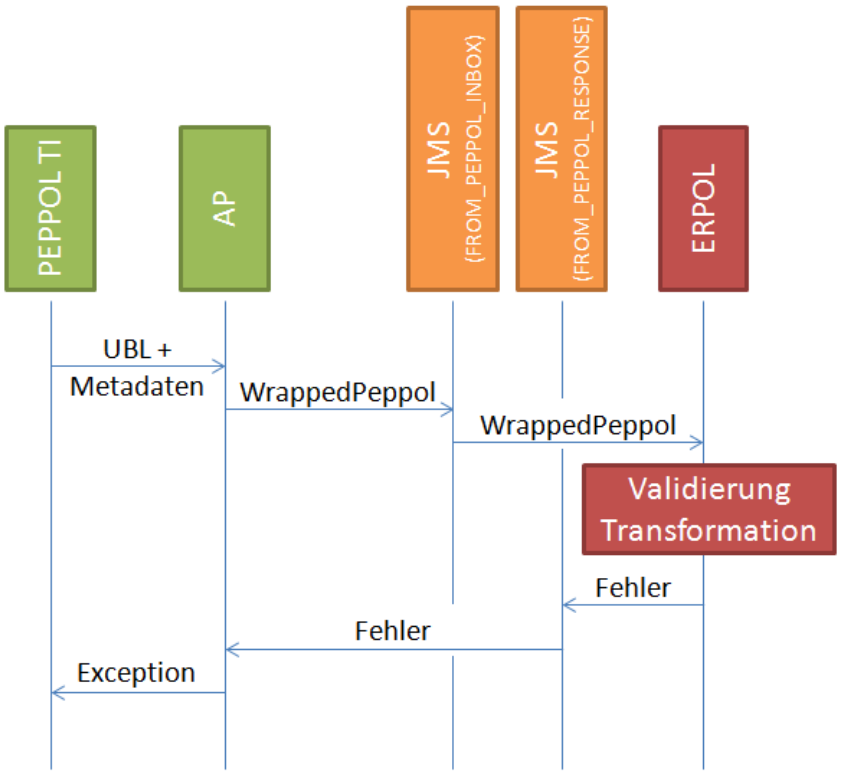


Abbildung 33: Sequenzdiagramm PEPPOL → ERPEL im Fehlerfall

Abbildung 33 zeigt das Sequenzdiagramm für die Übermittlung PEPPOL nach ERPEL im Fehlerfall. In dem gezeigten Fall ist in ERPOL während der Transformation ein Fehler aufgetreten. Der Ablauf ist entsprechend anzupassen, wenn der Fehler an einer anderen Stelle auftritt.

#### *5.3.4.3 Dokumententypen*

Im Access Point kommen die Metadaten als Teil des SOAP-Headers an, während die Nutzdaten im SOAP-Body enthalten sind. Zur Datenübertragung der Eingangsdaten in die JMS-Queue

*FROM_PEPPOL_INBOX* bietet sich das in Kapitel 5.3.3.2.2 definierte WrappedPeppol-Format an. Es besitzt alle notwendigen Felder, wobei das `EndpointURL`-Element für eingehende Dokumente aus PEPPOL nicht benötigt wird. Die Verarbeitungsergebnis von ERPOL im WrappedErpel-Format (siehe Kapitel 5.3.3.2.1) in die JMS-Queue *TO_ERPEL_INBOX* gestellt, sodass der ERPEL-Messaging Hub sowohl beim Einstellen in eine JMS-Queue (*FROM_ERPEL_INBOX*) als auch beim Abholen (*TO_ERPEL_INBOX*) immer Nachrichten im selben Format verarbeiten kann.

#### 5.3.4.3.1 PeppolReceiverResponse

Die Verarbeitungsergebnisse von ERPOL müssen in folgendem XML-Format in die Queue *FROM_PEPPOL_RESPONSE* gereicht werden:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns="http://www.helger.com/ns/peppol-receiver-response/1.0"
           targetNamespace="http://www.helger.com/ns/peppol-receiver-
response/1.0"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
  <xs:simpleType name="PeppolReceiverLevelType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="info" />
      <xs:enumeration value="warn" />
      <xs:enumeration value="error" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="PeppolReceiverMessageType">
    <xs:sequence>
      <xs:element name="Level" type="PeppolReceiverLevelType" />
      <xs:element name="Text" type="xs:string" />
      <xs:element name="Exception" type="xs:string" minOccurs="0" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="PeppolReceiverResponseType">
    <xs:sequence>
      <xs:element name="Message" type="PeppolReceiverMessageType"
minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="Success" type="xs:boolean" use="required" />
  </xs:complexType>
  <xs:element name="PeppolReceiverResponse" type="PeppolReceiverResponseType"
/>
</xs:schema>
```

Dieses Format entspricht dem Layout das der CIPA e-Delivery Access Point von den Empfänger-Plugins erwartet. Der Inhalt des `Level`-Elements entspricht dem zu verwendenden Log-Level, das `Text`-Element enthält den zentralen Text (z.B. die Fehlermeldung) und im `Exception`-Element kann eine aufgetretene Exception als Text eingefügt werden.

### *5.3.4.4 cipa-start-jmsreceiver*

Der *cipa-start-jmsreceiver* ist ein Plugin für den CIPA e-Delivery Access Point. Es muss anstelle des standardmäßig enthaltenen *cipa-start-filereceiver* verwendet werden, damit eingehende Nachrichten in einer JMS-Queue landen bzw. die Verarbeitungsergebnisse von dort ausgelesen

werden. Als Dokumentenformate werden sowohl das WrappedPeppol-Format (siehe Kapitel 5.3.3.2.2) als auch das PeppolReceiverResponse-Format (siehe Kapitel 5.3.4.3.1) verwendet.

Um eine effiziente synchrone Verarbeitung von JMS-Nachrichten zu erreichen muss der *cipa-start-jmsreceiver*, da er die *FROM_PEPPOL_INBOX* JMS-Queue befüllt, eine JMS-Antwort-Queue (JMSReplyTo) und eine zufällige JMS-Korrelations-ID (JMSCorrelationID) vor der Übertragung setzen. Aus diesem Grund ist es nicht möglich zwei verschiedene JMS-Queues für positive und negative Antworten zu definieren, da der *cipa-start-jmsreceiver* sonst nicht wüsste wo die synchrone Antwort auf die Nachricht zu finden wäre. Der JMS-Empfänger in ERPOL sollte über Events getriggert werden (in dem das JMS `MessageListener` Interface implementiert wird), und muss beim Senden der JMS-Antwort die in der Quellnachricht enthaltene Antwort-Queue sowie dieselbe Korrelations-ID verwenden.

Die Nachrichten in beide Richtungen können nicht persistent sein, um eine möglichst performante Verarbeitung zu gewährleisten. Der *cipa-start-jmsreceiver* muss außerdem ein Timeout zum Empfang der Antwort Nachricht setzen, damit die Verarbeitung auch vom Access Point synchron durchgeführt werden kann, ohne dass der ursprüngliche PEPPOL-Sender ein Timeout erhält.

Folgende Fehler können während der Ausführung auftreten und führen schlussendlich zu einer Exception im Access Point:

| Fehler-ID | Beschreibung |
|---|---|
| SOURCE_TOO_MANY | Die PEPPOL-Nachricht besteht nicht aus genau einer Payload (sondern aus 0 oder mehr als einer) |
| SOURCE_INVALID_XML | Das UBL-Dokument ist kein gültiges XML-Dokument |
| DELIVERY_ERROR | Beim Einstellen in die JMS-Queue *FROM_PEPPOL_INBOX* ist ein Fehler aufgetreten. |
| RESPONSE_TIMEOUT | Beim Warten auf die Antwort aus der JMS-Queue *FROM_PEPPOL_RESPONSE* ist ein Timeout aufgetreten. |
| DESTINATION_INVALID_XML | Das aus der JMS-Queue FROM_PEPPOL_RESPONSE gelesene XML-Dokument ist kein gültiges XML-Dokument oder es entspricht nicht dem XML-Schema. |

Für den *cipa-start-jmsreceiver* gibt es eine eigene Konfigurationsdatei, die im Classpath unter `/config-start-jmsreceiver.properties` zu finden sein muss. Dort müssen folgende Einstellungen getätigt werden:

- `connection-string` – Der Verbindungsstring mit dem sich *cipa-start-jmsreceiver* zum JMS Server verbindet (kein Standardwert). Zum Testen mit einer lokalen ActiveMQ-Server-Instanz mit der Standardkonfiguration kann `tcp://localhost:61616` verwendet werden.
- `persistent-messaging` – ein Flag das angibt, ob die JMS-Nachrichten im JMS-Provider zwischengespeichert werden sollen oder nicht (Standardwert: `false`)
- `inbox-queue-name` – Name der JMS-Queue für eingehende Nachrichten (Standardwert: FROM_PEPPOL_INBOX)
- `response-queue-name` – Name der JMS-Queue für die Antwortnachrichten (Standardwert: FROM_PEPPOL_RESPONSE)

- `response-timeout` – die maximale Zeit in Millisekunden, die auf die Antwortnachricht gewartet werden soll (Standardwert: 30000). Falls der Wert 0 oder kleiner ist, so soll unbeschränkt auf die Antwort gewartet werden.

Um den *cipa-start-jmsreceiver* in einer Access Point Installation zu nutzen, muss die Datei `WEB-INF/lib/cipa-start-filereceiver-X.Y.Z.jar` aus der Installation gelöscht werden und stattdessen die Datei `cipa-start-jmsreceiver-X.Y.Z.jar` in dasselbe Verzeichnis kopiert werden. Dieser registriert sich über SPI automatisch beim Access Point als Handler für eingehende Dokumente.

Das Projekt *cipa-start-jmsreceiver*, das im Rahmen dieser Arbeit entwickelt wurde, steht als Open Source Software unter der „Apache 2“-Lizenz unter https://github.com/phax/cipa-start-jmsreceiver zur Verfügung.

## 5.4 Umsetzung

Im Folgenden werden die technischen Details der ERPOL-Umsetzung im Detail beschrieben. Zuerst werden die Entwicklungsumgebung und die Testumgebung beschrieben. Anschließend werden die Laufzeitumgebung und der Systemaufbau beschrieben.

### 5.4.1 Entwicklungsumgebung

ERPOL wird als Java 1.6 Web-Applikation entwickelt. Als Entwicklungsumgebung kommt Eclipse 4.x zum Einsatz (http://www.eclipse.org/ - Stand 30.3.2013). Als Build System kommt Apache Maven 3.0.x zum Einsatz (http://maven.apache.org/ - Stand 30.3.2013), sowohl auf der Commandline sowie als Eclipse Plugin (Teil der "Eclipse IDE for Java EE Developers").

Als Sourcecode Verwaltungssystem wird Subversion eingesetzt (http://subversion.apache.org/ - Stand 30.3.2013), mit den entsprechenden Clients auf der Kommandozeile (SlikSVN - http://www.sliksvn.com/en/download/ - Stand 30.3.2013), in Eclipse (Subclipse - http://subclipse.tigris.org/ - Stand 30.3.2013) und als Windows Shell-Extension (TortoiseSVN - http://tortoisesvn.net/ - Stand 30.3.2013). Bei Subversion ist darauf zu achten, dass alle Komponenten dieselbe Major- und Minor-Version haben (z.B. alle 1.7.x oder alle 1.8.x), da sie sich sonst selbst in die Quere kommen.

Zur Messung der Unit Test Code Coverage (Code-Abdeckung) kommt das Eclipse Plugin von Emma names EclEmma (http://www.eclemma.org/ - Stand 30.3.2013) zum Einsatz. Zur Laufzeitmessung wird der YourKit Profiler for Java (http://www.yourkit.com/ - Stand 30.3.2013) verwendet. Die statische Code Analyse wird mit Hilfe des FindBugs Eclipse Plugins (http://findbugs.sourceforge.net/manual/eclipse.html - Stand 30.3.2013) durchgeführt.

Zum Testen von ERPOL innerhalb von Eclipse wird Jetty (http://www.eclipse.org/jetty/ - Stand 30.3.2013) 7.x als Application Server verwendet, da er sich besser einbetten lässt als zum Beispiel Apache Tomcat (http://tomcat.apache.org/ - Stand 30.3.2013). Die Unit Tests werden mit Hilfe von Junit 4.x-Annotations definiert (http://junit.org/ - Stand 30.3.2013).

Für den Zugriff auf PEPPOL kommt CIPA e-Delivery 1.1.2 zum Einsatz (https://joinup.ec.europa.eu/svn/cipaedelivery/tags/1.1.2 - Stand 9.11.2013). Diese Version enthält sowohl einen Access Point (*cipa-start-server*) als auch einen SMP mit modifizierenden

APIs (*cipa-smp-full-webapp*). Als Basis-Bibliotheken kommen die ph OSS-Bibliotheken zum Einsatz (z.B. phloc-commons - http://code.google.com/p/phloc-commons/ - Stand 30.3.2013 bzw. phloc-schematron - http://code.google.com/p/phloc-schematron/ - Stand 23.4.2013), die auch die Basis von CIPA e-Delivery bilden.

Als JMS-Provider kommt Apache ActiveMQ (http://activemq.apache.org/ - Stand 23.4.2013) mit der integrierten KahaDB zum Einsatz. Für den produktiven Betrieb ist anstelle der KahaDB die Verwendung einer relationalen Datenbank – wie etwa MySQL – zu empfehlen, da KahaDB hauptsächlich auf Performance und weniger auf Sicherheit ausgelegt ist.

### 5.4.2 Testumgebung

#### *5.4.2.1 Registry Schnittstelle*

Zum Testen der Registry Schnittstelle genügt es, Unit-Tests zu erstellen, die die entsprechenden REST-Service-Aufrufe in unterschiedlicher Reihenfolge tätigen. Dazu ist keine spezielle Testapplikation notwendig.

#### *5.4.2.2 Dokumentenaustauschschnittstelle*

Um die entwickelten Komponenten lokal testen zu können, sind einige Besonderheiten zu berücksichtigen. Da es sich sowohl bei ERPEL als auch bei PEPPOL um Netzwerke zur Übertragung von Dokumenten handelt, ist ein Integrations-Test recht umständlich. Es ist zwingend erforderlich, dass eine zweite Partei als Sender oder als Empfänger involviert ist, weshalb Extraaufwand in die Tests der jeweiligen Netzwerk-Knotenpunkte investiert werden muss.

Als Fremdsysteme während des Testens kommen ein PEPPOL-SMP mit deaktivierter Verbindung zum SML, sowie Apache ActiveMQ als JMS-Provider mit der integrierten KahaDB zur Speicherung persistenter Nachrichten zum Einsatz.

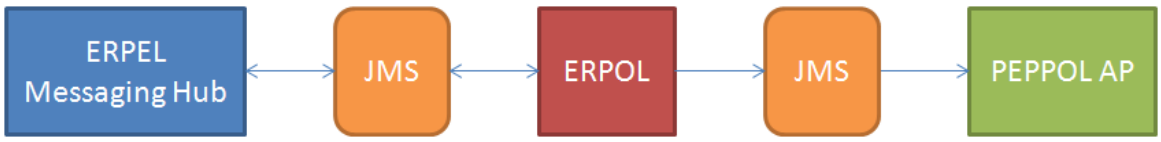


Abbildung 34: ERPEL zu PEPPOL Überblick

Abbildung 34 zeigt den vereinfachten Dokumentenfluss von ERPEL nach PEPPOL. Um diesen Ablauf zu testen bedarf es eines „Fake Messaging Hubs“ der ERPEL-Dokumente in die JMS-Queue stellt, sowie eines „Fake Access Points“, der die Dokumente entgegen nimmt, aber nicht versendet.


Abbildung 35: PEPPOL zu ERPEL Überblick

Abbildung 35 zeigt den Dokumentenfluss in die umgekehrte Richtung von PEPPOL nach ERPOL. In diesem Fall wird ein „Fake Access Point“ benötigt, der UBL-Dokumente in die entsprechende

JMS-Queue stellt, sowie ein „Fake Messaging Hub“, der die Dokumente aus der JMS-Queue entgegen nimmt, aber nicht weiterleitet.

Um die Tests in beide Richtungen abzudecken gibt es eine eigene Web-Applikation, mit der es möglich ist, händisch Dokumente auszuwählen und hochzuladen. Für die Richtung ERPEL zu PEPPOL muss das ausgewählte Dokument im Format WrappedErpel in die JMS-Queue *FROM_ERPEL_INBOX* gestellt werden, wo es von ERPOL abgeholt und verarbeitet wird. Nach Eintreffen einer Nachricht in der Queue *TO_PEPPOL_INBOX* wird – je nachdem ob es ein Gutfall oder ein Schlechtfall ist – die passende Antwortnachricht in die Queue *TO_PEPPOL_RESPONSE* geschrieben. Anschließend muss das endgültige Testergebnis aus den JMS-Queues *FROM_ERPEL_SUCCESS* (Gutfall) oder *FROM_ERPEL_ERROR* (Schlechtfall) gelesen und verifiziert werden. Der vollständige Ablauf des Dokumentenaustauschs ist in Abbildung 30 (Gutfall) bzw. Abbildung 31 (Schlechtfall) ersichtlich und in Kapitel 5.3.3.1 beschrieben.

Für die andere Richtung von PEPPOL nach ERPEL ist das Vorgehen ähnlich. Das ausgewählte Dokument muss in die JMS-Queue *FROM_PEPPOL_INBOX* gestellt werden, wo es von ERPOL abgeholt und verarbeitet wird. Nach dem Einlangen einer Nachricht in der JMS-Queue *TO_ERPEL_INBOX* muss diese verifiziert werden. Die Nachricht in der Queue *FROM_PEPPOL_RESPONSE* muss ebenfalls gelesen und verifiziert werden. Der vollständige Ablauf des Dokumentenaustauschs ist in Abbildung 32 (Gutfall) bzw. Abbildung 33 (Schlechtfall) ersichtlich und in Kapitel 5.3.4.2 beschrieben.

#### *5.4.2.3 erpol-web-tester*

Speziell zum Testen des ERPOL-Dokumentenaustauschs wurde der *erpol-web-tester* umgesetzt. Dabei handelt es sich um eine Web-Applikation mit der die Datenübertragung und Datentransformation in beide Richtungen getestet werden kann. Um diese Applikation zu verwenden muss mindestens der ActiveMQ JMS-Server sowie ERPOL (das Projekt *erpol-web-document*) laufen. Abhängig von den gewählten Einstellungen müssen ggf. noch andere Applikation laufen - siehe unten.

##### 5.4.2.3.1 ERPEL nach PEPPOL

Als Startseite wurde das Versenden von ERPEL nach PEPPOL gewählt. Die notwendigen Metadaten (Absender, Empfänger, Dokumenttyp und Prozess) und eine existierende ERPEL-Datei müssen angegeben werden. Sofern der PEPPOL-Versender ebenfalls getestet wird, kann auch noch gewählt werden, ob dieser eine Erfolgsnachricht oder eine Fehlernachricht zurücksenden soll. Durch Druck auf den Button "Start processing" wird das gewählte Dokument und alle Metadaten in ein *WrappedErpel*-Dokument (siehe Kapitel 5.3.3.2.1) verpackt und in die JMS-Queue *FROM_ERPEL_INBOX* gestellt.

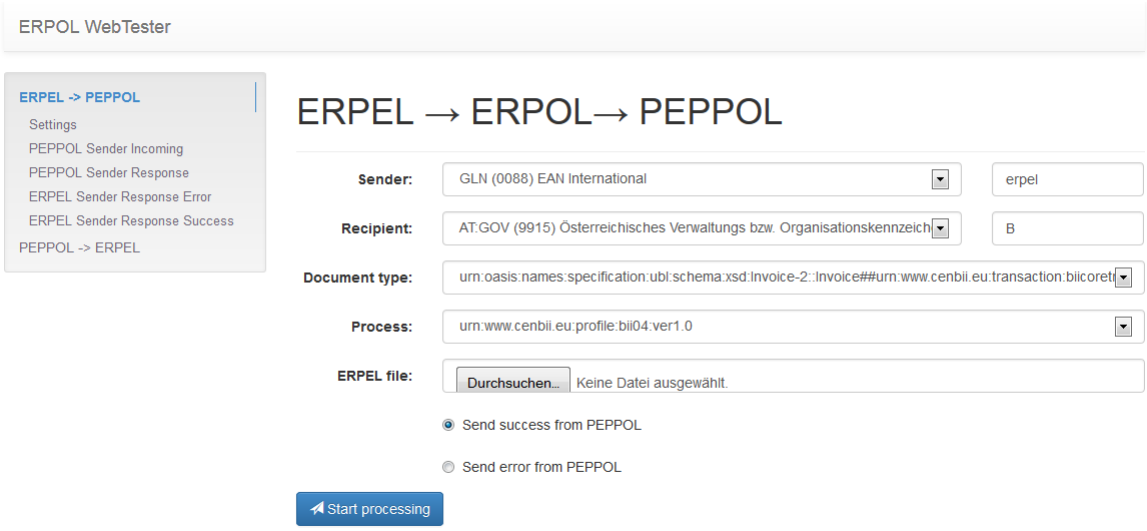


**Abbildung 36: Startseite ERPEL nach PEPPOL**

Dieses wird dann von ERPOL abgeholt, transformiert und in die Queue *TO_PEPPOL_INBOX* gestellt. Das Ergebnis des Versendens wird von ERPOL aus der Queue *TO_PEPPOL_RESPONSE* gelesen und entweder in ein *Erpel2PeppolSuccess*- (siehe Kapitel 5.3.3.2.5) oder ein *Erpel2PeppolError*-Dokument (siehe Kapitel 5.3.3.2.4) umgewandelt und in die passende Queue gestellt (*FROM_ERPEL_SUCCESS* oder *FROM_ERPEL_ERROR*).

Alle involvierten Dokumente können im *erpol-web-tester* detailliert angezeigt werden:


**Abbildung 37: Übersicht aller via PEPPOL zu versendenden Dokumente**

Abbildung 37 zeigt die Liste aller Dokumente, die vom PEPPOL-AccessPoint versendet werden sollen. Durch Klick auf eine Message-ID (blau hinterlegt) werden die Details der Nachricht angezeigt.

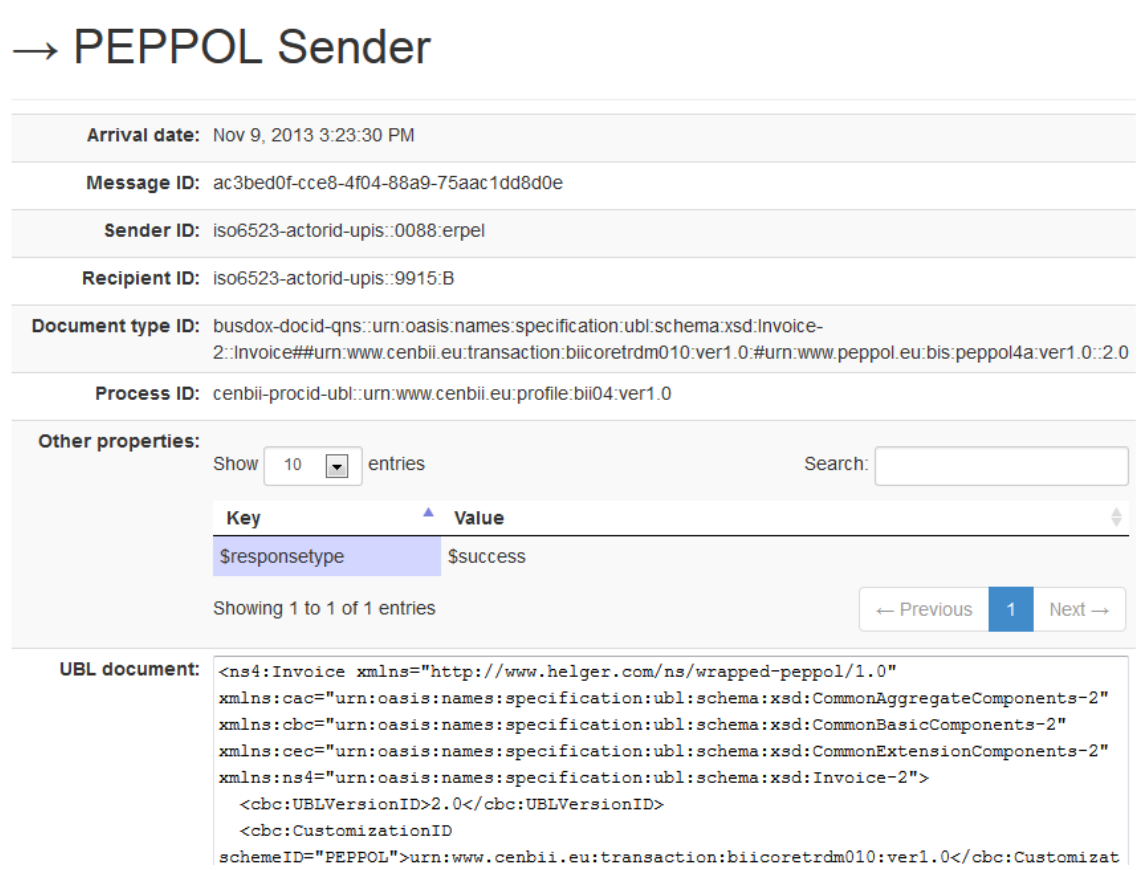


Abbildung 38: Details einer von PEPPOL zu versendenden Nachricht (Ausschnitt)

Neben den vorher angegebenen Metadaten wird auch noch das Ankunftsdatum sowie das gesamte durch ERPOL erzeugte UBL-Dokument angezeigt. Unter "Other properties" werden sonstige, implementierungsspezifische Eigenschaften dargestellt. Die in Abbildung 38 gezeigte Eigenschaft "$responsetype" mit dem Wert "$success" zeigt an, dass *erpol-web-tester* für diese Nachricht eine Erfolgsnachricht erzeugen soll.

Im Menüpunkt "PEPPOL Sender Response" werden die aus dieser Nachricht erzeugten Antworten dargestellt.


Abbildung 39: Übersicht der Ergebnisse des Versands via PEPPOL

Auch hier können durch Klick auf die blau hinterlegte Message-ID die Details der Nachricht angesehen werden:

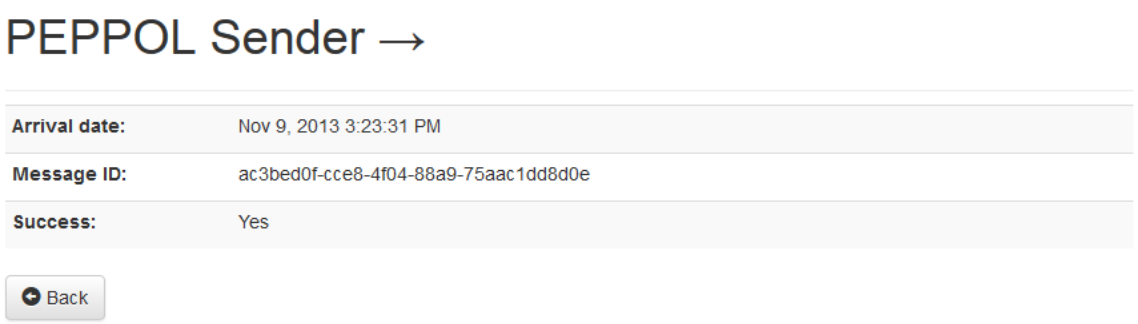


Abbildung 40: Detailergebnisse des Versands via PEPPOL im Erfolgsfall

Da für Abbildung 40 eine Gutmeldung erzeugt wurde, sind die verfügbaren Informationen relativ spartanisch.


Abbildung 41: Detailergebnisse des Versands via PEPPOL im Fehlerfall

Für Abbildung 41 wurde ein Fehlerfall erzeugt. In diesem Fall sieht man die detaillierten Fehlermeldungen - in diesem Fall nur "Fake error from ERPOL Web Tester".

Die an ERPEL zurückgesendeten Erfolgsmeldungen sind im Menüpunkt "ERPEL Sender Response Success" sichtbar. Dabei handelt es sich um die aus der *FROM_ERPEL_SUCCESS*-Queue entnommenen Nachrichten.


Abbildung 42: An ERPEL zurückgesendete Erfolgsnachrichten

Aus Mangel an weiteren Informationen werden in den Details im Erfolgsfall derzeit auch keine weiteren Informationen dargestellt als bereits in der Übersicht sichtbar sind:

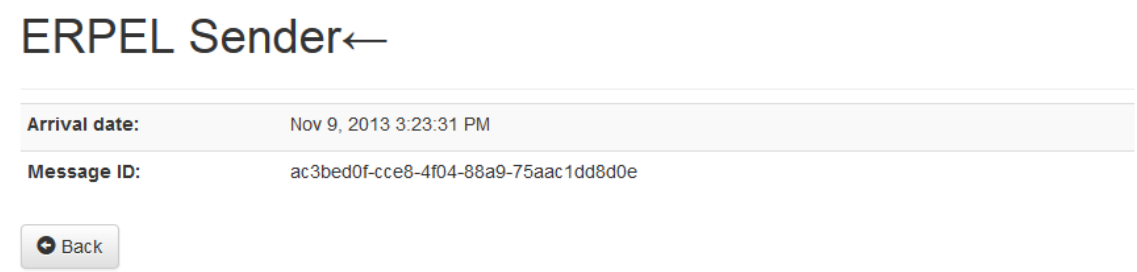


Abbildung 43: Details einer an ERPEL zurückgesendeten Erfolgsnachricht

Die an ERPEL gesendeten Fehlermeldungen sind im Menüpunkt "ERPEL Sender Response Error" ersichtlich. Dabei handelt es sich um die Nachrichten, die aus der *FROM_ERPEL_ERROR*-Queue geholt werden.


Abbildung 44: An ERPEL zurückgesendete Fehlernachrichten

Die Details der Fehlermeldung zeigen die Fehlermeldungen wie sie vom PEPPOL Access Point erzeugt wurden:

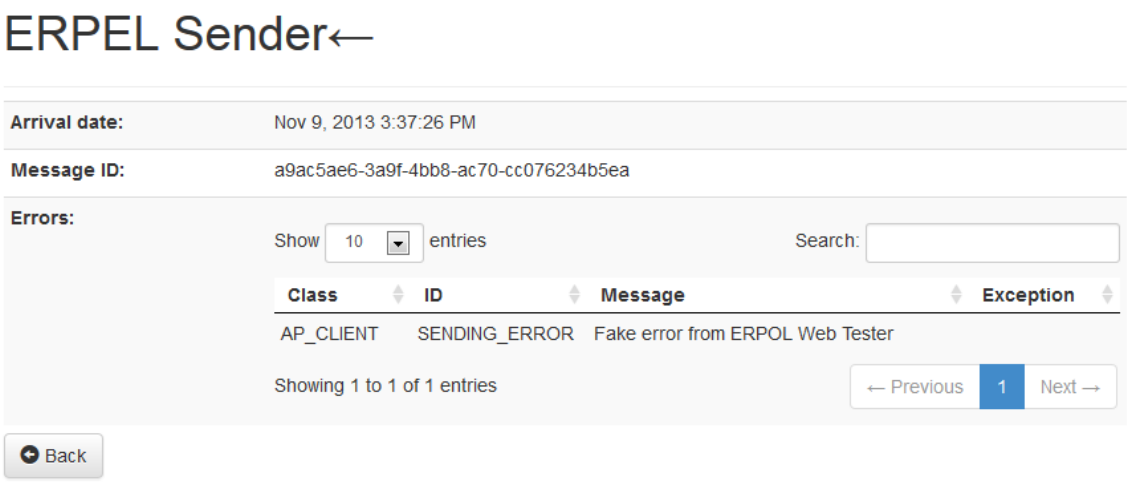


Abbildung 45: Details einer an ERPEL zurückgesendeten Fehlernachricht

#### 5.4.2.3.2 ERPEL nach PEPPOL Einstellungen

Über die Einstellungen (Menüpunkt "Settings") können sowohl der Mock-Versand über den PEPPOL-AccessPoint wie auch die Mock-Verarbeitung der für ERPEL bestimmten Nachrichten ein- bzw. ausgeschaltet werden:

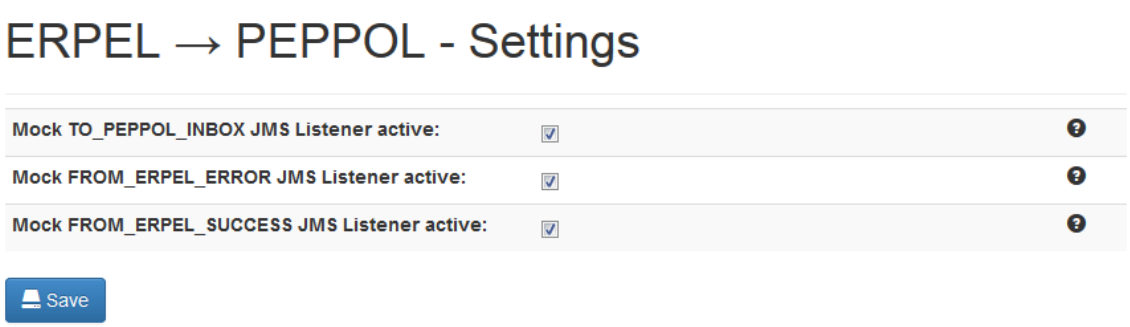


Abbildung 46: Einstellungen für den Austausch von ERPEL nach PEPPOL

Wenn der "TO_PEPPOL_INBOX Listener" deaktiviert wird, muss ein separater PEPPOL AccessPoint Client aktiv sein, der die Nachrichten über das PEPPOL-Netzwerk versendet (z.B. *cipa-start-jmssender*). Dadurch ändert sich auch die Darstellung der Seite zum Versenden der Nachrichten:

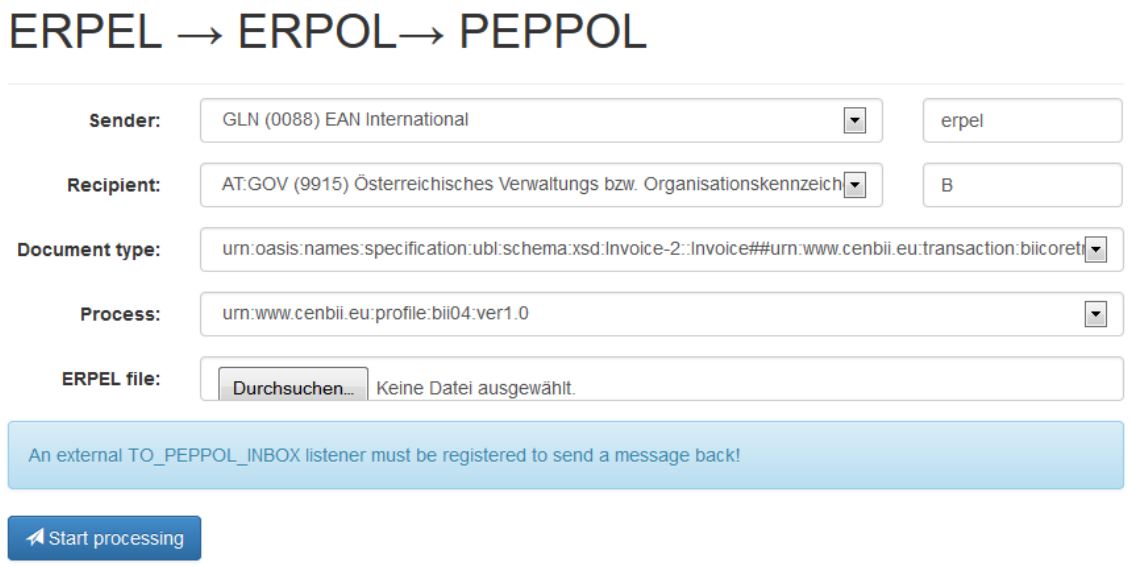


Abbildung 47: Startseite mit deaktiviertem TO_PEPPOL_INBOX Listener

Gleichzeitig werden auch in den Seiten "PEPPOL Sender Incoming" und "PEPPOL Sender Response" entsprechende Hinweise angezeigt, dass derzeit keine neuen Nachrichten gespeichert werden können:

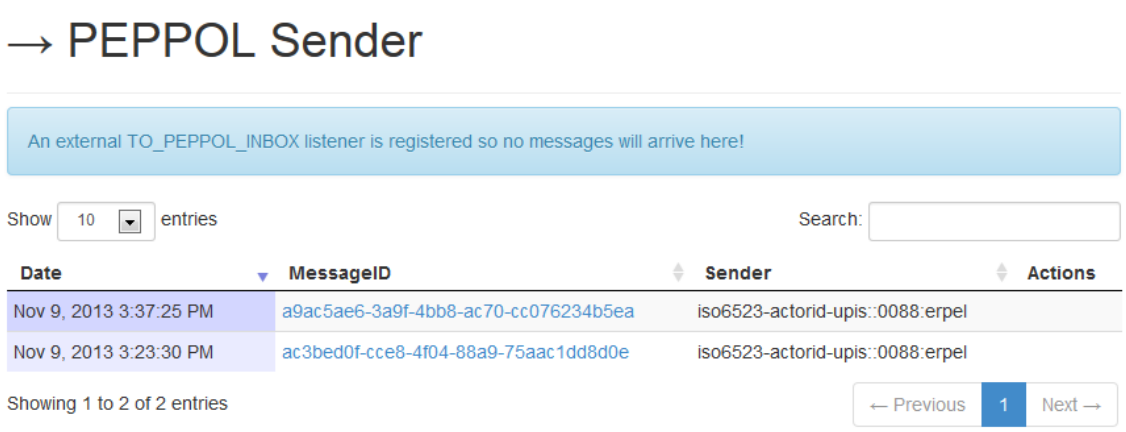


Abbildung 48: Darstellung der zu versendenden Nachrichten bei deaktiviertem Listener

Die Ergebnisse des Versands durch PEPPOL landen trotzdem im *erpol-web-tester* und die folgende Abbildung zeigt eine echte Fehlermeldung:

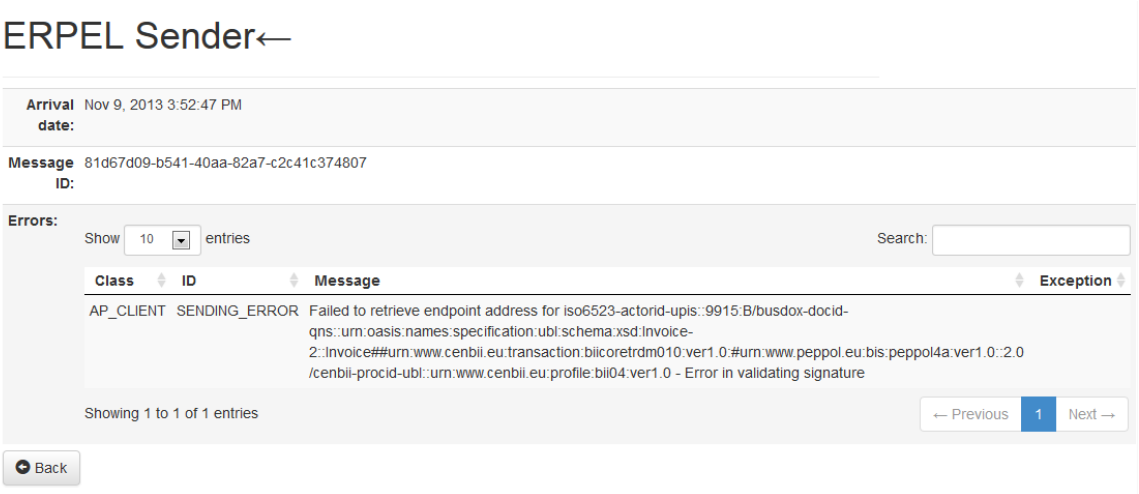


Abbildung 49: Echte Fehlermeldung beim Versenden über PEPPOL

Wenn in den Einstellungen zusätzlich noch die beiden ERPEL Listener deaktiviert werden, kann der Nachrichtenaustausch mit aktivem ERPEL getestet werden. In diesem Fall werden auch auf den entsprechenden Seiten Hinweise dargestellt:

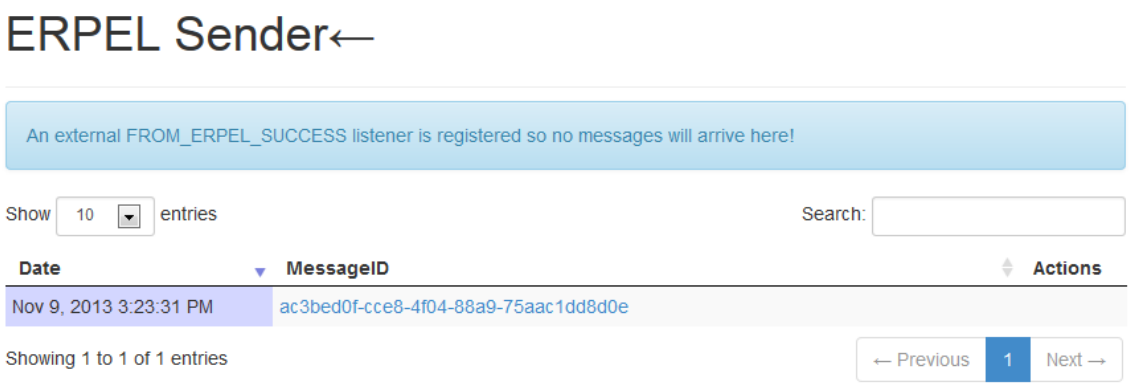


Abbildung 50: Darstellung von "ERPEL Sender Response Success" bei deaktiviertem Listener

#### 5.4.2.3.3 PEPPOL nach ERPEL

Das Versenden von PEPPOL nach ERPEL befindet sich im Menüpunkt "PEPPOL → ERPEL". Die notwendigen Metadaten (Absender, Empfänger, Dokumenttyp und Prozess) und eine existierende PEPPOL UBL-Datei müssen angegeben werden. Zusätzlich kann ausgewählt werden, ob das Dokument direkt über einen PEPPOL AccessPoint Client versendet werden soll, wodurch auch *cipa-start-jmsreceiver* getestet werden kann, oder ob das Dokument direkt in die JMS-Queue gestellt werden soll. Durch Druck auf den Button "Start processing" wird das gewählte Dokument entweder direkt per AccessPoint Client versendet oder alle Metadaten und das UBL-Dokument in ein *WrappedPeppol*-Dokument (siehe Kapitel 5.3.3.2.2) verpackt und in die JMS-Queue *FROM_PEPPOL_INBOX* gestellt.

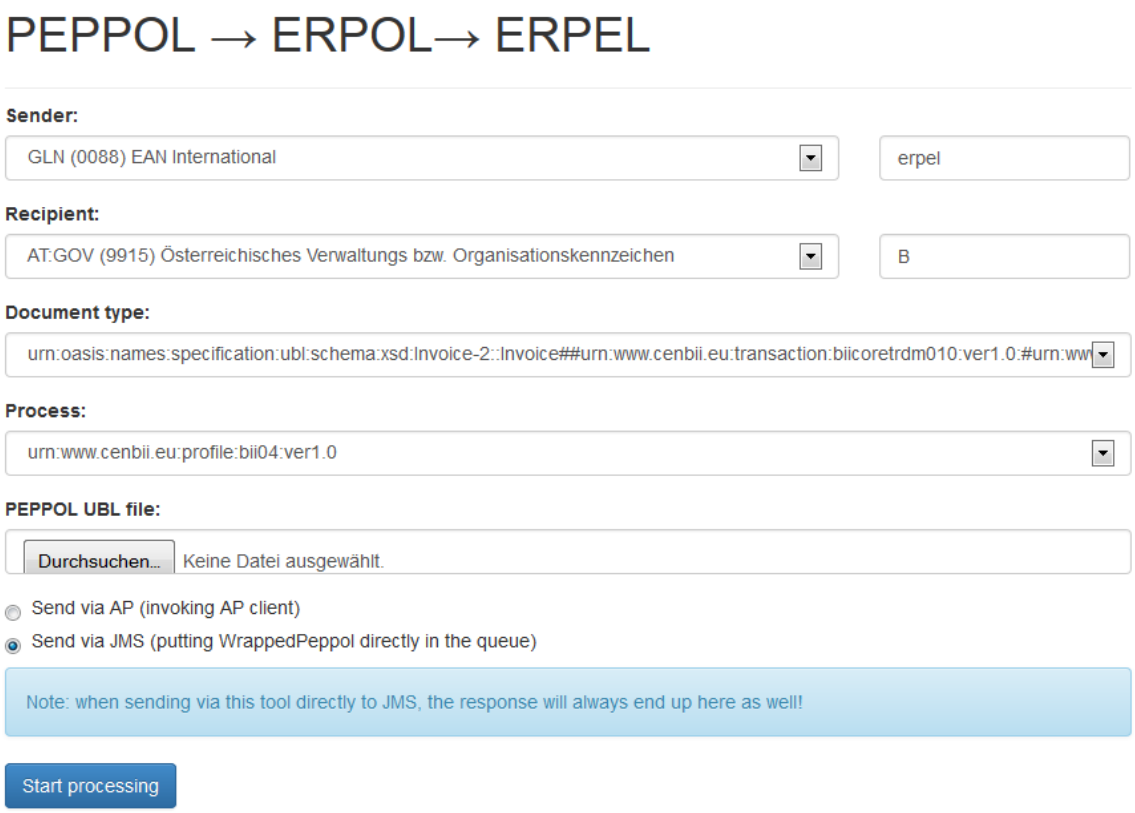


**Abbildung 51: Startseite PEPPOL nach ERPEL**

Wenn das Dokument via JMS gesendet wird, so landet das Ergebnis immer im *erpol-web-tester*, da mit einer zufälligen JMS-Korrelations-ID synchron auf die Antwort gewartet werden muss.

Auf der Seite "ERPEL Sender Incoming" wird die Liste aller an ERPEL versendeten Dokumente gezeigt:

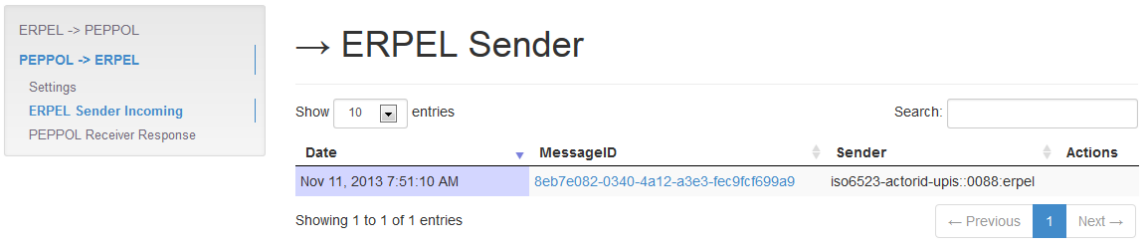


Abbildung 52: Übersicht über alle in ERPEL eingegangenen Dokumente

Durch Klick auf die Message-ID können auch hier die Details der durch ERPOL transformierten Nachricht angezeigt werden.

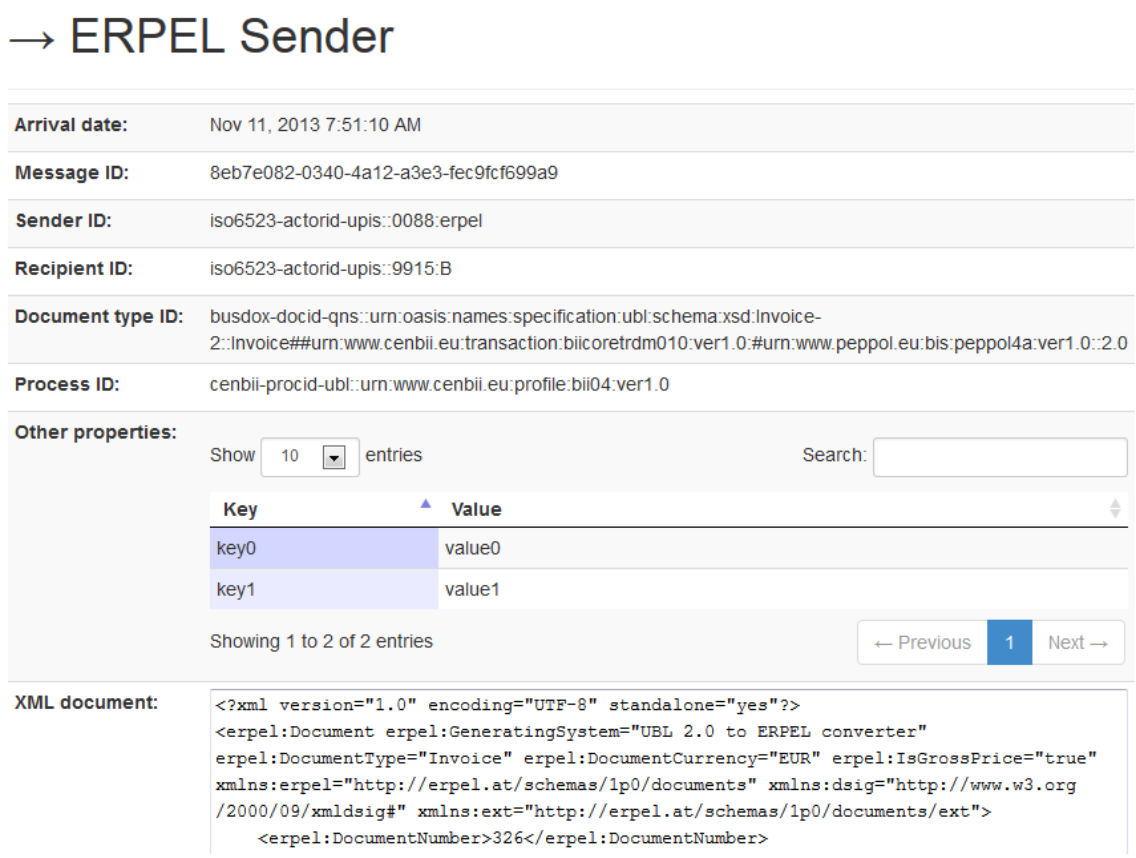


Abbildung 53: Details eines in ERPEL eingegangenen Dokuments

In Abbildung 53 sind neben der Metadaten auch zwei sonstige Eigenschaften sowie das von ERPOL transformierte ERPEL-Dokument abgebildet. Die "Other properties" sind nur zu Testzwecken übermittelt worden und haben in diesem Fall keine Bedeutung bei der Verarbeitung. Das dahinterliegende Datenformat ist das *WrappedErpel*-Format.

Im Menüpunkt "PEPPOL Receiver Response" werden alle von ERPEL an PEPPOL zurückgelieferten Nachrichten angezeigt.


Abbildung 54: Übersicht aller PEPPOL Antwortnachrichten

In Abbildung 54 ist im Vergleich zu den anderen Listen keine Message-ID enthalten. Das liegt daran, dass die hier gezeigten Verarbeitungsergebnisse normalerweise von einem START AccessPoint Server implizit zu einer bestimmten Message-ID in Bezug gesetzt werden. Durch Klick auf den jeweiligen Wert der Success-Spalte werden die Details einer Nachricht angezeigt, die intern im *PeppolReceiverResponse*-Format (siehe Kapitel 5.3.4.3.1) vorliegt.

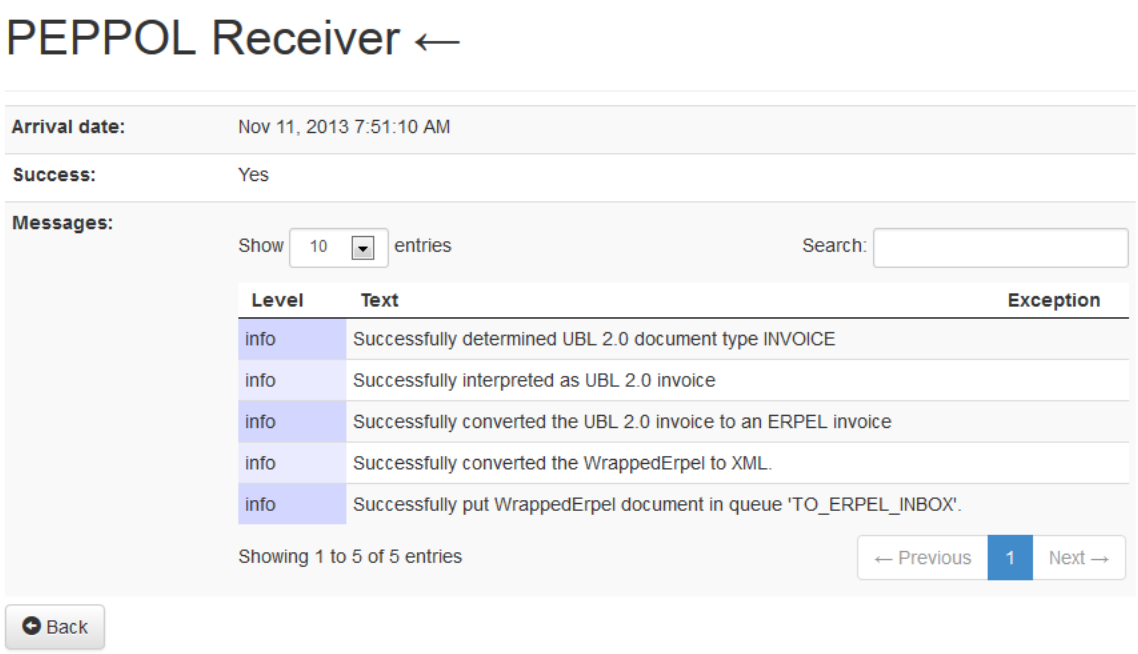


Abbildung 55: Details einer PEPPOL-Antwortnachricht

In Abbildung 55 ist ein Gutfall dargestellt und es wird eine Liste von ERPOL-Verarbeitungsschritten gezeigt. Diese Nachrichten werden vom CIPA e-Delivery AccessPoint in seiner Standardkonfiguration in die Logdatei geschrieben, und somit sind auch die genauen Verarbeitungsschritte von ERPOL zu einem späteren Zeitpunkt nachvollziehbar.

#### 5.4.2.3.4 PEPPOL nach ERPEL Einstellungen

Die Einstellungsmöglichkeiten bei der Dokumentenübertragung von PEPPOL nach ERPEL sind etwas geringer als in die andere Richtung:

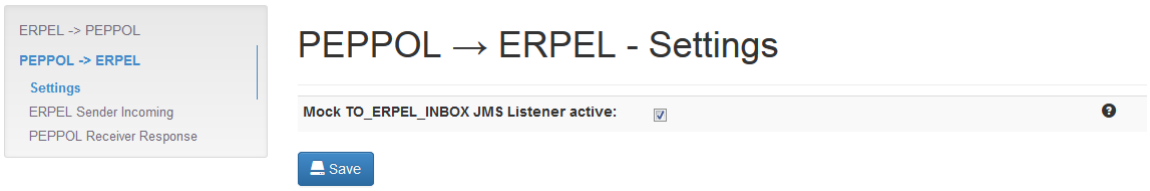


Abbildung 56: Einstellungen der Dokumentenübertragung PEPPOL nach ERPEL

Es kann nur eingestellt werden, ob die Verarbeitung der Nachrichten durch ERPEL getestet werden soll oder nicht. Eine vergleichbare Einstellung auf Seiten von PEPPOL gibt es nicht, da die Applikation, die die PEPPOL Dokumente weiterleitet auch immer für das Abholen der Ergebnisse verantwortlich ist. Um den Echt-Versand über einen PEPPOL AccessPoint Server zu testen, muss beim Versand des Dokuments die Option "Send via AP" gewählt werden (siehe Abbildung 51).

Wenn die Option "TO_ERPEL_INBOX Listener" deaktiviert ist, so kommen auf der Seite "ERPEL Sender Incoming" keine neuen Nachrichten dazu, worauf auch mit einer entsprechenden Meldung hingewiesen wird:

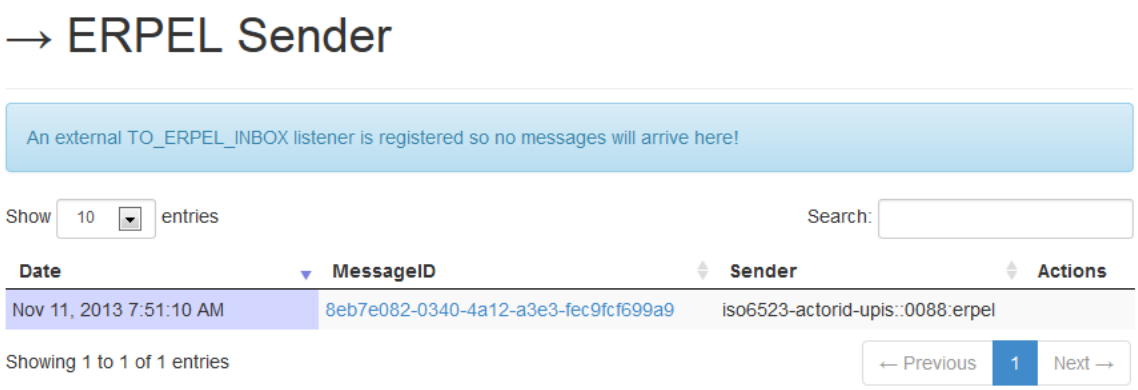


Abbildung 57: In ERPEL ankommende Nachrichten bei deaktiviertem Listener

In diesem Fall muss ERPEL aktiv sein und die Nachrichten "echt" zu verarbeiten.

### 5.4.3 Projektstruktur

Aus den Anforderungen an ERPOL sowie an die Testbarkeit ergibt sich folgende Projektstruktur für ERPOL:

- **erpol-web-registry**: die Web-Applikation, die die Registry-Schnittstelle implementiert. Sie enthält auch die Testfälle für die Registry-Schnittstelle. Siehe Kapitel 5.2.
- **erpol-web-document**: die Web-Applikation, die die bidirektionale Dokumentenaustauschschnittstelle implementiert. Das ist die ERPOL Web-Applikation.
    - **erpeldoc**: eine Bibliothek, die den ERPEL-Dokumententyp kapselt. Sie dient der einfachen Wiederverwendung des Dokumententyps in anderen Projekten.
    - **erpol-erpel-api**: eine Bibliothek, die die für den ERPOL-ERPEL-Nachrichtenaustausch notwendigen Datenformate kapselt: *WrappedErpel* (siehe Kapitel 5.3.3.2.1), *Erpel2PeppolError* (siehe Kapitel 5.3.3.2.4) und *Erpel2PeppolSuccess* (siehe Kapitel 5.3.3.2.5).

    - **erpol-transformation**: eine Bibliothek die ERPEL-Dokumente in UBL-Dokumente transformiert und vice versa (jedoch wie oben beschrieben nur Bestellung und Rechnung). Diese Bibliothek ist losgelöst von *erpol-web-document* um eine leichtere Wiederverwendung zu ermöglichen. Siehe Kapitel 5.3.2.
- **cipa-start-server**: der CIPA e-Delivery Access Point Server mit der ERPOL-spezifischen Konfiguration.
    - **cipa-start-jmsreceiver**: eine Bibliothek, die als Plugin für den CIPA e-Delivery Access Point fungiert und eingehende Dokumente in eine JMS-Queue stellt. Siehe Kapitel 5.3.4.4.
        - **cipa-start-jms-api**: eine Bibliothek, die die gemeinsam genutzten Datenformate von *cipa-start-jmsreceiver* und *cipa-start-jmssender* enthält: *WrappedPeppol* (siehe Kapitel 5.3.3.2.2), *PeppolSenderResponse* (siehe Kapitel 5.3.3.2.3) und *PeppolReceiverResponse* (siehe Kapitel 5.3.4.3.1).
- **cipa-smp-full-webapp**: die CIPA e-Delivery SMP Web-Applikation mit der ERPOL-spezifischen Konfiguration. Siehe Kapitel 5.2.2.4.
- **cipa-start-jmssender**: die Web-Applikation, die Nachrichten aus der Queue *TO_PEPPL_INBOX* entgegen nimmt und über die PEPPOL Transport Infrastruktur versendet. Siehe Kapitel 5.3.3.3.
- **erpol-web-tester**: die Testapplikation mit der der Dokumentenaustausch in beide Richtungen getestet werden kann. Siehe Kapitel 5.4.2.3.

Im Echtbetrieb wird das Projekt *erpol-web-tester* nicht benötigt, da dieses nur zum Testen dient. Im Testbetrieb sollte das Projekt *cipa-start-jmssender* nur aktiv sein, wenn produktive Nachrichten übertragen werden sollen.

*cipa-start-server* und *cipa-smp-full-webapp* sind die unveränderten Versionen von CIPA e-Delivery. Sie unterscheiden sich von den offiziell bereitgestellten Versionen nur durch die Konfiguration der PEPPOL-Zertifikate.

*cipa-start-jms-api*, *cipa-start-jmsreceiver* und *cipa-start-jmssender* wurden im Rahmen dieser Arbeit entwickelt und stehen als Open Source Projekte unter https://github.com/phax/cipa-start-jms-api, https://github.com/phax/cipa-start-jmsreceiver bzw. https://github.com/phax/cipa-start-jmssender zur Verfügung. Als Lizenz wurde die „Apache 2“-Lizenz gewählt, die auch die Nutzung in kommerziellen Produkten zulässt.

### 5.4.4 Anforderungen an die Laufzeitumgebung

ERPOL ist betriebssystemunabhängig und wurde sowohl auf Windows als auch Linux getestet. Als Voraussetzung ist übliche Hardware (32 oder 64 Bit) mit mindestens 1GB RAM – je mehr desto besser – sowie eine Internetverbindung notwendig. Softwareseitig muss mindestens Java 1.6 und ein Applikationsserver mit Unterstützung für die Servlet-Spezifikation 2.5 wie z.B. Apache Tomcat (mindestens Version 6.x) oder Jetty (in der Version 7.x) vorhanden sein. Auf der Firewall müssen neben den Standard HTTP und HTTPS Ports auch jene zum JMS-Provider und zum SMP-Server freigeschaltet werden, falls die Komponenten nicht auf derselben Maschine wie ERPOL laufen. Außerdem ist eine Firewall-Freischaltung von SMP zum SML auf Port 443 notwendig (https://smk.peppolcentral.org und https://sml.peppolcentral.org).

Für den Einsatz der CIPA e-Delivery Komponenten ist es notwendig die Java-Installation zu modifizieren, da eine spezielle Web Service-Bibliothek im "endorsed" Verzeichnis des Java Runtime Environments (JRE) enthalten sein muss. Die Anforderung ergibt sich aus dem Einsatz der Metro-Bibliotheken (http://metro.java.net/ - Stand 30.3.2013) für den Web Service Stack (Helger 2012a).

### 5.4.5 Systemaufbau

Für das im Rahmen dieser Arbeit entstehende Demo-Szenario ist es ausreichend, wenn alle benötigten Software-Artefakte – PEPPOL SMP, PEPPOL Access Point, JMS Provider und ERPOL – auf einer Maschine laufen. SMP, Access Point und ERPOL sind jeweils Java Web Applikationen, die zusammen in einem einzigen Tomcat laufen können, während ActiveMQ als eigenständige Anwendung läuft. Die Einschränkungen des SMP sind, dass er auf dem Standard-Web-Port 80 im Root-Context erreichbar sein muss, d.h. er muss als „ROOT“ Applikation innerhalb des Tomcats veröffentlicht werden. Der Access Point wiederum sollte auf Port 443, jedenfalls aber über HTTPS erreichbar sein, weshalb es sich anbietet, einen Apache httpd Webserver vor den Tomcat zu schalten, um die SSL-Verbindung korrekt zu terminieren. In diesem Fall fungiert der httpd-Server auch als Reverse Proxy zum Tomcat, damit dieser nicht von außen erreichbar sein muss.

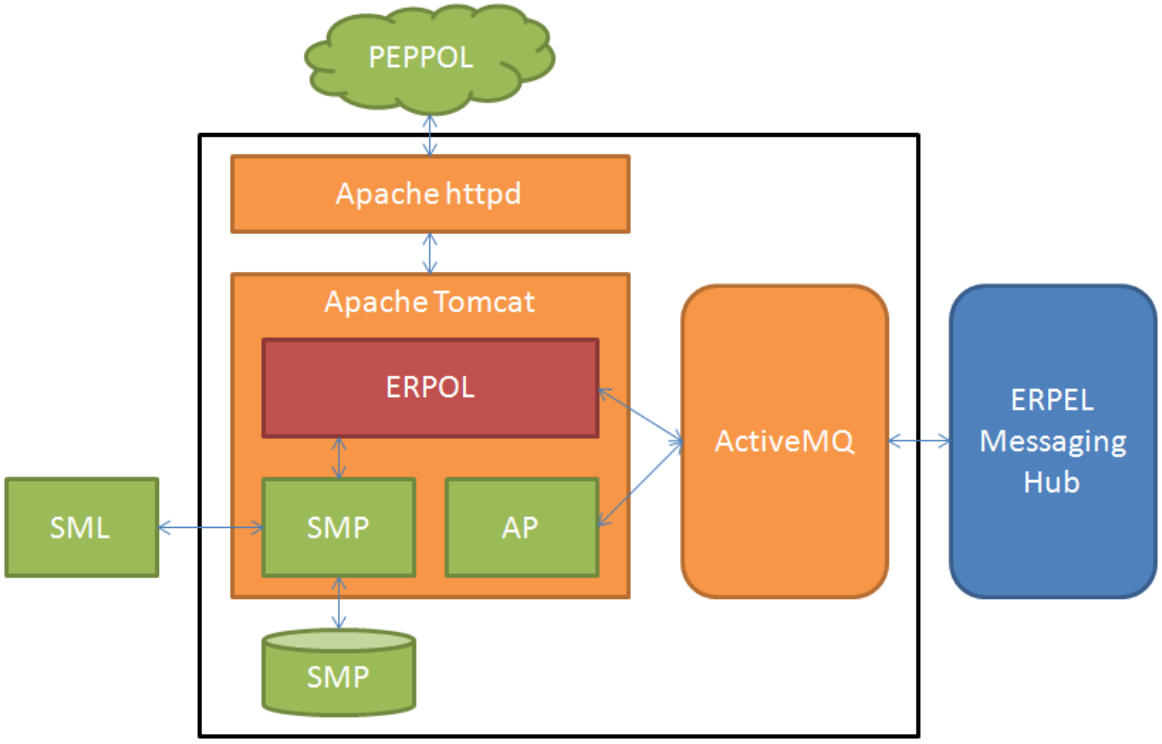


Abbildung 58: ERPOL Systemaufbau

Abbildung 58 zeigt den Aufbau und die Verbindungen der einzelnen Komponenten von ERPOL. Grüne Elemente (SML, SMP, AP, PEPPOL) zeigen PEPPOL Komponenten, blaue Elemente (ERPEL Messaging Hub) zeigen ERPEL-Komponenten und orangefarbene Elemente (httpd, Tomcat, ActiveMQ) stehen für externe Komponenten.

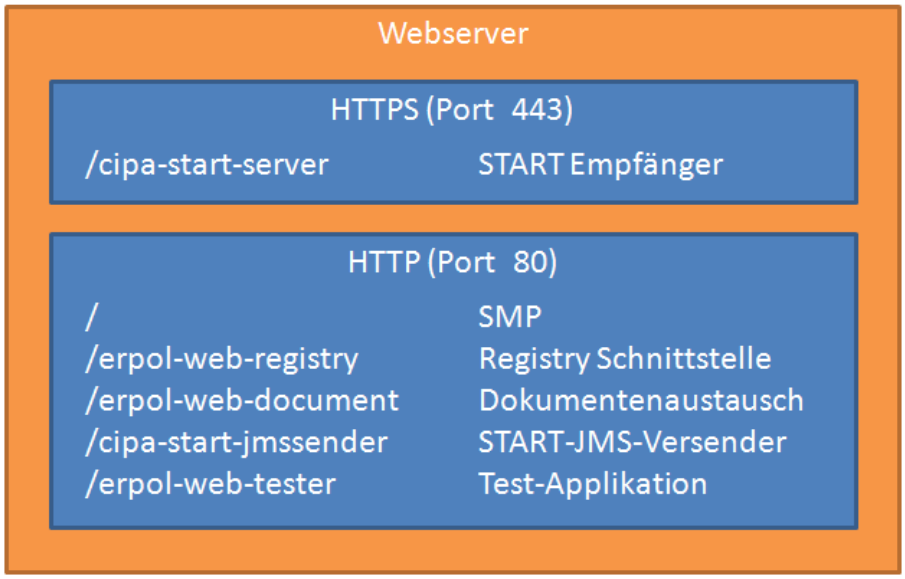


Abbildung 59: ERPOL Struktur auf dem Webserver

Abbildung 59 zeigt die Zusammenfassung aller für ERPOL notwendigen Web-Applikationen inklusive dem Port und dem Basisverzeichnis. Der JMS-Provider ist in dieser Abbildung nicht integriert, da er als eigenständiger Prozess außerhalb des Applikationsservers läuft.

# 6 Case Study

In diesem Kapitel soll die Arbeitsweise von ERPOL in einem produktiven Szenario überprüft werden. Dazu wurde die Übermittlung von PEPPOL-Rechnungen an den österreichischen Bund gewählt: Nachrichten werden von Lieferanten des Bundes an ERPEL gesendet, und mithilfe von ERPOL über die PEPPOL-Transportinfrastruktur an den österreichischen Bund gesendet.

Dieses Anwendungsbeispiel wurde ausgesucht, da der PEPPOL Access Point des Bundes zum Zeitpunkt der Erstellung dieser Arbeit der einzige öffentlich verfügbare PEPPOL Access Point in Österreich ist und gleichzeitig die E-Rechnung an den Bund einen real nutzbaren Use Case darstellt.

Zuerst werden die Details der Case Study in Kapitel 6.1 und die Software-Lösung für die Einlieferung von e-Rechnungen an den österreichischen Bund („E-Rechnung an den Bund“ – ER>B) in Kapitel 6.2 beschrieben. Abschließend werden die Umsetzungsdetails für diese spezifische Schnittstelle definiert (Kapitel 6.3) und Kapitel 6.4 fasst die Informationen zusammen und gibt einen Ausblick.

## 6.1 Szenario-Beschreibung

Die folgende Abbildung zeigt einen Überblick über den Ablauf der Einlieferung einer Rechnung von ERPEL über ERPOL und PEPPOL an den österreichischen Bund.

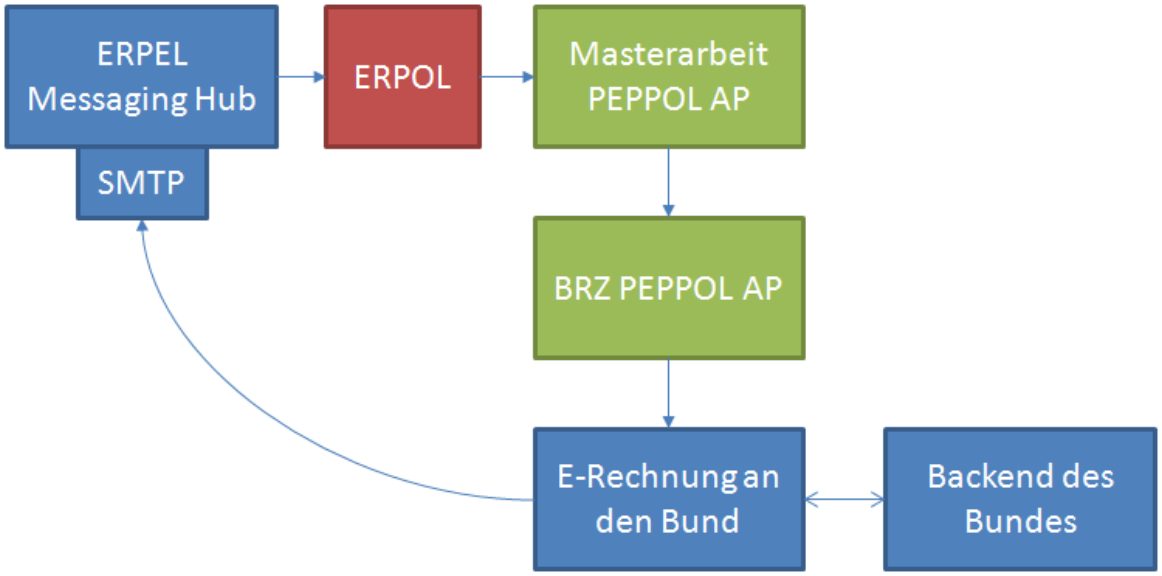


Abbildung 60: Ablauf der Dokumenteneinlieferung an den Bund

Rechnungen, die via ERPEL versendet werden, werden in ERPOL nach UBL transformiert, und über den PEPPOL AccessPoint Client (in der Grafik als „Masterarbeit PEPPOL AP“ dargestellt) an den PEPPOL AccessPoint Server des Bundes (in der Grafik als „BRZ PEPPOL AP“ dargestellt) gesendet. Von dort werden e-Rechnungen zur Software-Lösung E-Rechnung an den Bund (ER>B; siehe Kapitel 6.2) weitergeleitet und schlussendlich verarbeitet. Als Antwort versendet ER>B asynchron ein E-Mail mit den Details über die Annahme oder Ablehnung der e-Rechnung.

## 6.2 E-Rechnung an den Bund (ER>B)

Sämtliche Lieferanten des Bundes sind seit 1.1.2014 gesetzlich dazu verpflichtet, Rechnungen an den Bund ausschließlich elektronisch zu legen. Die rechtlichen Grundlagen finden sich im IKT-Konsolidierungsgesetz §5 (https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=20007805 – Stand 13.7.2013). Österreich ist das zweite Land in Europa - nach Dänemark - das die e-Rechnung per Gesetz vorschreibt, und hat damit bei diesem Thema auch eine Vorreiterrolle innerhalb der EU.

Bundesdienststellen, und damit verpflichtende Empfänger von e-Rechnungen, sind alle Bundesministerien und deren nachgeordnete Dienststellen sowie das Parlament, die Präsidentschaftskanzlei, der Verwaltungsgerichtshof, der Verfassungsgerichtshof, die Volksanwaltschaft und der Rechnungshof. Länder, Städte, Gemeinden und ausgegliederte Unternehmen sind von diesem Gesetz nicht betroffen und müssen daher auch keine e-Rechnungen empfangen können.

Zur Einbringung von strukturierten elektronischen Rechnungen an den Bund stehen verschiedene Möglichkeiten (Einlieferungskanäle) zur Verfügung:

- PEPPOL – als europäische Standardlösung
- Online-Formular – zur manuellen Erfassung von e-Rechnung auf der Webseite „E-Rechnung an den Bund“ (ER>B; https://www.erb.gv.at – Stand 11.11.2013)
- Upload – zum manuellen Hochladen von vorgefertigten XML-Dateien auf ER>B
- Webservice – zur automatisierten Einbringung von e-Rechnungen über eine standardisierte Schnittstelle.


**Abbildung 61: Die Startseite von E-Rechnung an den Bund (Stand 4.1.2014)**

Folgende e-Rechnungs-Dateiformate werden von ER>B unterstützt: ebInterface, UBL sowie das ER>B-interne Datenformat AustroFIX. Das häufig verwendete PDF-Format wird von ER>B nicht unterstützt, da es keine strukturierten Rechnungsdaten enthält.

Neben Rechnungen können auch Gutschriften übermittelt werden, sofern das Trägerformat (z.B. ebInterface) das auch unterstützt. Optional können zu allen Rechnungen und Gutschriften noch Beilagen (in bestimmten Formaten) mit einem Gesamtvolumen von bis zu 15 MB übermittelt werden.

### 6.2.1 Voraussetzung für die Verwendung von ER>B

Für alle Einlieferungskanäle außer PEPPOL muss zuerst eine Anmeldung am Unternehmens-serviceportal (USP; https://www.usp.gv.at – Stand 13.7.2013) durchgeführt werden, bevor e-Rechnungen an den Bund eingebracht werden können.

- Manuelle Rechnungseinbringung: für am USP authentifizierte Benutzer erscheint ein neuer Menüpunkt „Rechnungseinbringung“ in ER>B mit dem e-Rechnungen per Online-

Formular erfasst oder manuell hochgeladen (Upload) werden können.


Abbildung 62: ER>B Rechnungseinbringung nach Anmeldung im USP

- Für die Einbringung per Webservice muss zusätzlich ein spezieller USP-Webservice-Benutzer angelegt werden, da auch sämtliche Webservice-Einlieferungen vom USP authentifiziert werden müssen.
- Bei Verwendung der PEPPOL-Transportinfrastruktur - wie im Fall dieses Anwendungsbeispiels - ist keine Registrierung und Authentifizierung am USP notwendig, da durch die spezifischen PEPPOL-Zertifikate bereits ein ausreichendes Vertrauen zum Versender der e-Rechnungen besteht.

### 6.2.2 Verarbeitung einer E-Rechnung

Sämtliche eingehende Rechnungen (und Gutschriften) werden automatisiert in das ER>B-spezifische Rechnungsformat AustroFIX umgewandelt und verarbeitet. Folgende Verarbeitungsschritte werden im Detail vollzogen:

1. Das eingehende XML-Dokument wird gegen das zugrundeliegende XML-Schema (UBL Rechnung oder ebInterface) validiert und in das interne XML-Format umgewandelt.
2. Es werden Prüfungen auf dem internen Format durchgeführt um sicherzustellen, dass alle Pflichtfelder ausgefüllt sind und dass alle notwendigen Zusammenhänge erfüllt sind (Technische Prüfung).
3. Im Erfolgsfall wird eine PDF-Datei für die empfangende Bundesdienststelle erzeugt, in der alle nach UStG §11 notwendigen Merkmale einer Rechnung enthalten sind.
4. Die Original-XML-Datei, die interne XML-Datei, eventuell vorhandene Beilagen und die erzeugte PDF-Datei werden an das Backend-System des Bundes zur weiteren Verarbeitung weitergeleitet.
5. Der Lieferant bekommt eine synchrone, positive Rückmeldung, dass die e-Rechnung angekommen und technische geprüft wurde. Im Falle der Verwendung des Online-Formulars und des Uploads ist das eine entsprechende Meldung auf dem Bildschirm und bei Verwendung des Webservices ist es die synchrone HTTP-Antwort auf die Rechnungseinlieferung.
6. Im Backend-System des Bundes werden zusätzliche inhaltliche Prüfungen durchgeführt (z.B. ob die Referenz auf die Bestellung gültig ist), sämtliche Dokumente in ein Langzeitarchiv gestellt und die e-Rechnung einem oder mehreren Sachbearbeiter(n) zugewiesen.
7. Verläuft die inhaltliche Prüfung erfolgreich, wird der Lieferant informiert. Die Mitteilung erfolgt per E-Mail und enthält neben einer textuellen Beschreibung auch eine PDF-Datei der Rechnung sowie die XML-Rechnungsdaten. Im Falle einer negativen inhaltlichen Prüfung bekommt der Lieferant ebenfalls ein E-Mail, jedoch ohne das Rechnungs-PDF. Es wurde der Weg per E-Mail gewählt, da die Verarbeitung im Backend-System – je nach

Auslastung – eine Durchlaufzeit von ein paar Sekunden bis zu ein paar Tagen haben kann. Da Sicherheit ein wichtiger Aspekt bei der Umsetzung von ER>B ist, werden diese Statusmeldungen nicht in ER>B für eine spätere Abholung gespeichert, sondern direkt an den Lieferanten übermittelt.
Alternativ zum Versenden eines E-Mails besteht auch die Möglichkeit das Ergebnis der inhaltlichen Prüfung als maschinenlesbare Nachricht übermittelt zu bekommen ("Webservice Callback"). Diese Möglichkeit besteht nur bei der Einbringung über die Webservice-Schnittstelle und nicht bei den anderen Einbringungsarten.

Die folgende Grafik zeigt den Einlieferungsprozess, wobei die Zeitachse von oben nach unten dargestellt ist und „HV-SAP“[5] das oben genannte Backend-System des Bundes ist. Die Nummerierung der einzelnen Schritte entspricht der Beschreibung.

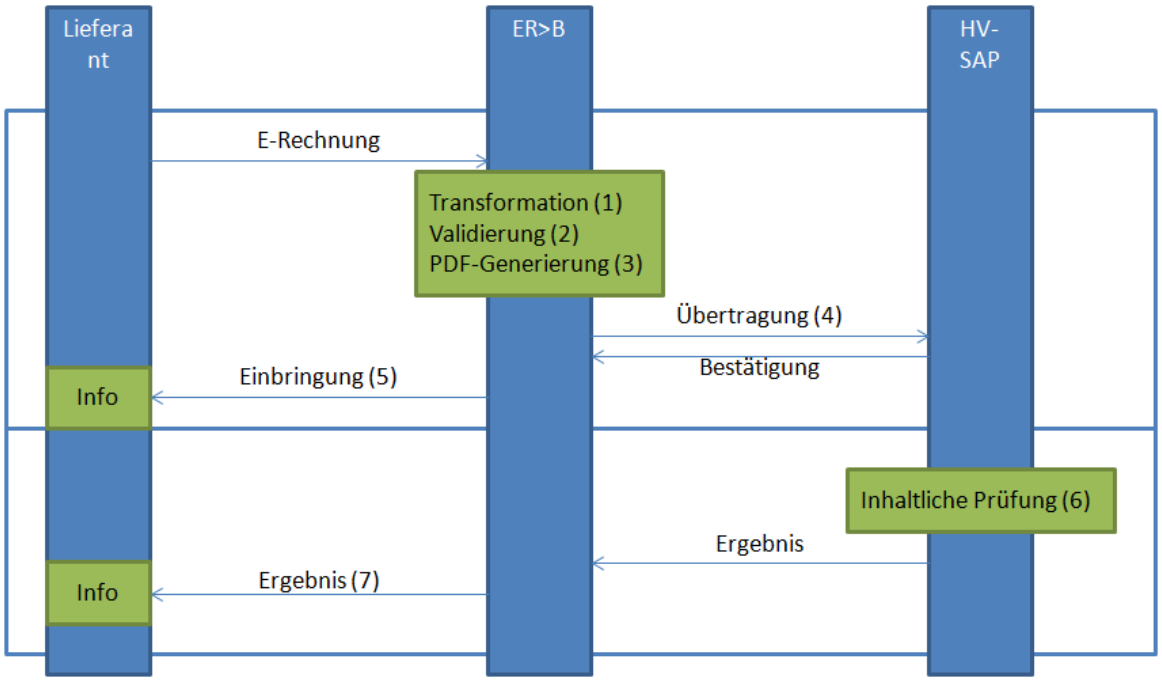


Abbildung 63: Übersicht über die Dokumenteneinbringung in ER>B (Helger 2013)

Falls die e-Rechnung per Webservice eingeliefert wurde, kann statt der E-Mail in Schritt 7 auch eine automatisierte Webservice-Antwort von ER>B abgesetzt werden, sofern der Lieferant eine von ER>B definierte Webservice-Schnittstelle implementiert hat. Dadurch ist eine 100% automatische Übertragung der e-Rechnung – ohne manuelle Zwischenschritte – möglich.

Tritt bei den Schritten 1-4 ein Fehler auf, so gilt die Rechnung als nicht eingelangt, und der Lieferant wird über die aufgetretenen Fehler informiert. Falls erst bei der inhaltlichen Prüfung ein Fehler auftritt (Schritt 6), so wird der Lieferant in Schritt 7 über den aufgetretenen Fehler informiert und die Rechnung gilt ebenfalls als nicht eingebracht.

Die technische und inhaltliche Prüfung müssen beide positiv absolviert werden, um die e-Rechnung bundesintern bearbeiten zu können. Sollten trotz positiver Absolvierung aller genannten Schritte noch Fragen oder Probleme auftreten (z.B. die Rechnung passt nicht zur

[5] HV-SAP ist der Name des Systems, das für die Haushaltsverrechnung des Bundes zuständig ist.

Bestellung, die Verrechnung hat bereits stattgefunden), erfolgt die Problemlösung nicht automatisiert, sondern der Sachbearbeiter des Bundes nimmt telefonisch oder per E-Mail Kontakt mit dem Lieferanten auf.

### 6.2.3 Rechnungsdaten

Der österreichische Bund hat spezifische Anforderungen an die in einer e-Rechnung enthaltenen Daten, die über die im Umsatzsteuergesetz (UStG) §11 definierten Anforderungen hinausgehen.

Es muss zur Identifikation des Lieferanten eine Kreditorennummer (auch Lieferantennummer genannt) angegeben werden. Dabei handelt es sich um die Nummer des Lieferanten in den Systemen des Bundes. Ein Lieferant kann seine Kreditorennummer für alle Bundesdienststellen verwenden, an die er e-Rechnungen senden muss (siehe https://www.erb.gv.at/index.jsp?p=info_channel&tab=yourid; Stand 12.1.2014).

Neben der Kreditorennummer ist auch die Auftragsreferenz ein spezielles verpflichtendes Feld. Mit diesem Wert kann die Rechnung zu einer Bestellung oder zu einer bestellenden Organisationseinheit zugeordnet werden, definiert als den Empfänger der Rechnung. Da der Bund sowohl bestellbezogene als auch nicht-bestellbezogene Rechnungen entgegen nimmt, gibt es unterschiedliche Ausprägungen der Auftragsreferenz. Eine genaue Beschreibung wie Auftragsreferenzen auszusehen haben, befindet sich unter https://www.erb.gv.at/index.jsp?p=info_channel (Stand 11.11.2013).

Um automatisiert Kontakt mit dem Lieferanten aufnehmen zu können, ist die E-Mail-Adresse des Lieferanten ebenfalls ein Pflichtfeld. An diese E-Mail-Adresse wird sämtliche automatisierte Kommunikation von ER>B gesendet.

### 6.2.4 PEPPOL-Schnittstelle

ER>B unterstützt auch die Eingangsrechnungsverarbeitung über PEPPOL. Derzeit werden über PEPPOL nur Rechnungen (PEPPOL-Dokumententyp `busdox-docid-qns::urn:oasis:names:specification:ubl:schema:xsd:Invoice-2::Invoice##urn:www.cenbii.eu:transaction:biicoretrdm010:ver1.0:#urn:www.peppol.eu:bis:peppol4a:ver1.0::2.0`) nach BIS4A (PEPPOL-Prozess `cenbii-procid-ubl::urn:www.cenbii.eu:profile:bii04:ver1.0` - siehe Kapitel 4.3.1) unterstützt. Als Empfänger muss immer die PEPPOL-Participant-ID `iso6523-actorid-upis::9915:b` verwendet werden. Die Daten über den genauen Empfänger innerhalb des Bundes werden aus der Auftragsreferenz in der Rechnung extrahiert.

In den PEPPOL-Spezifikationen gibt es keinen Prozess, um nur Gutschriften zu übertragen. Gutschriften sind zwar ein Teil von BIS 5A (siehe Kapitel 4.3.1), allerdings erfordert diese weitere Prozessschritte, die vom Bund nicht unterstützt werden. Trotzdem hat man sich in ER>B nach Rücksprache mit PEPPOL dazu entschlossen, UBL-Gutschriften über BIS 5A abzuwickeln, auch wenn die anderen Prozessschritte und Dokumentenarten nicht unterstützt werden. Somit können UBL-Gutschriften über die PEPPOL-Schnittstelle an den österreichischen Bund übermittelt werden.

### 6.2.5 Testbarkeit

Um während der Entwicklung einer e-Rechnungs-Lösung an den Bund bestmögliche Unterstützung zu bieten, existiert unter https://test.erb.gv.at (Stand 1.12.2013) eine offene

Seite, auf der sowohl das Online-Formular als auch der Upload „live“ getestet werden können. Die so eingelieferten Rechnungen werden vom Bund nicht bezahlt, sondern werden als reine Testdokumente behandelt. Der Prozess der e-Rechnungs-Einlieferung ist identisch zur Einlieferung auf dem Produktiv-System – lediglich die an den Einbringer zurück gesendeten Daten enthalten an vielen Stellen den Hinweis, dass es sich um eine Test-Einbringung handelt.

Zum Test der PEPPOL-Schnittstelle wurde die spezielle PEPPOL-Participant-ID `iso6523-actorid-upis::9915:test` zur Verfügung gestellt. Damit werden Rechnungen auf das oben erwähnte ER>B-Testsystem geroutet und damit auch nicht bezahlt.

## 6.3 Technische Umsetzung

### 6.3.1 ERPEL-Konfiguration

Um ERPOL für e-Rechnungen den Bund einzusetzen, ist nur der Weg von ERPEL nach PEPPOL verfügbar, da keine Retourmeldungen von ER>B über PEPPOL gesendet werden. Es können in diesem Fall nur Rechnungen (keine Bestellungen) übertragen werden, da ER>B nur für e-Rechnungen ausgelegt ist.

Folgende PEPPOL-Metadaten müssen bei der Übermittlung aus ERPEL angegeben werden:

- Sender ID: die PEPPOL Sender-ID kann beliebig gewählt werden
- Recipient ID: `iso6523-actorid-upis::9915:b` (https://www.erb.gv.at) oder `iso6523-actorid-upis::9915:test` (https://test.erb.gv.at)
- Documenttype ID: `busdox-docid-qns::urn:oasis:names:specification:ubl:schema:xsd:Invoice-2::Invoice##urn:www.cenbii.eu:transaction:biicoretrdm010:ver1.0:#urn:www.peppol.eu:bis:peppol4a:ver1.0::2.0` (Invoice für BIS 4A)
- Process ID: `cenbii-procid-ubl::urn:www.cenbii.eu:profile:bii04:ver1.0` (BIS 4A)
- Endpoint URL: https://www.erb.gv.at/accessPointService/ (aus dem SMP-Eintrag von `iso6523-actorid-upis::9915:b` ermittelt) bzw. https://test.erb.gv.at/accessPointService/ (aus dem SMP-Eintrag von `iso6523-actorid-upis::9915:test` ermittelt) (Stand 2.2.2014)

Die beiden Bundes-spezifischen Felder Lieferantennummer und Auftragsreferenz müssen in einem ERPEL-Dokument in den Feldern `Supplier/SupplierIDissuedByCustomer` (Lieferantennummer) bzw. `Customer/DocumentReference/DocumentID` (Auftragsreferenz) gesetzt sein. Mit diesen beiden Feldern ist die korrekte Zuordnung zu einer Bestellung beim Bund möglich. Die E-Mail-Adresse des Lieferanten ist jene, die im Feld `Supplier/Contact/Email` hinterlegt ist.

### 6.3.2 Test-Durchführung

Zum Testen der Einlieferung von e-Rechnungen an den Bund kann der für ERPOL entwickelte *erpol-web-tester* (siehe Kapitel 5.4.2.3) verwendet werden. Bevor eine (Test-)Rechnung übertragen werden kann muss allerdings der *TO_PEPPOL_INBOX* JMS-Listener in den „ERPEL→PEPPOL Settings“ deaktiviert werden, damit die Rechnung auch vom *cipa-start-*

*jmssender* (und damit vom PEPPOL AccessPoint Client) versendet werden kann. Falls der Listener nicht deaktiviert wird, würde das Senden per PEPPOL-Transportinfrastruktur gemockt werden.


**Abbildung 64: Deaktivierter *TO_PEPPOL_INBOX* JMS Listener im *erpol-web-tester***

Zum eigentlichen Versenden der e-Rechnung an den Bund sind im *erpol-web-tester* bereits die richtigen Einstellungen vorausgewählt:

- Der Sender kann eine beliebige PEPPOL ID sein.
- Die Empfänger-ID muss vom Typ „9915“ sein und den Wert „b“ (https://www.erb.gv.at) oder „test“ (https://test.erb.gv.at) haben. Groß- und Kleinschreibung spielt für dieses Feld keine Rolle.
- Als Dokumententyp muss Invoice für BIS 4A ausgewählt sein.
- Als Prozess muss BIS 4A ausgewählt sein.
- Eine gültige ERPEL-XML-Datei muss vorhanden sein.

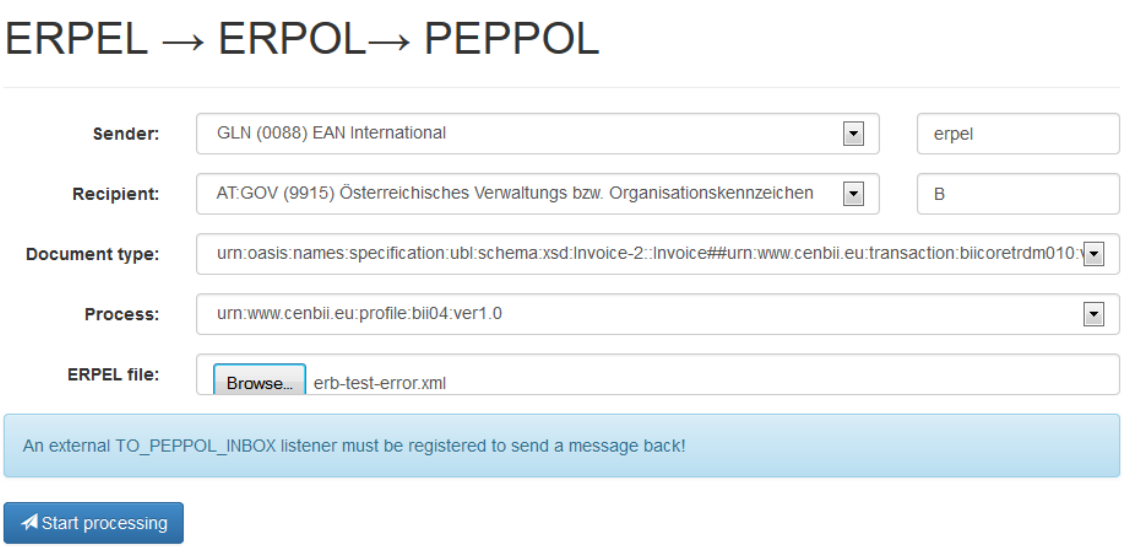


**Abbildung 65: *erpol-web-tester* Maske für den Versand an den österreichischen Bund**

Nach dem Absenden der Nachricht muss nach einiger Zeit eine Antwort an den ERPEL Sender vorliegen. Diese kann im *erpol-web-tester* entweder unter „ERPEL Sender Response Error“ oder unter „ERPEL Sender Response Success“ eingesehen werden – je nachdem ob das Versenden erfolgreich war oder nicht.

Wenn das Versenden erfolgreich war, aber in ER>B inhaltliche Fehler in der Rechnung gefunden wurden, so wird ein E-Mail mit den Details an die in der Rechnung genannte E-Mail-Adresse gesendet.

## 6.4 Zusammenfassung und Ausblick

Nachdem gezeigt wurde, dass mit der ERPOL-Schnittstelle eine echte Rechnung an den österreichischen Bund übertragen werden kann, wird nochmals explizit darauf hingewiesen, dass das gleichermaßen für andere über PEPPOL angeschlossene Empfänger ebenfalls funktioniert. Auch ist der Anwendungsbereich von ERPOL nicht auf B2G beschränkt, sondern es kann auch problemlos im B2B eingesetzt werden.

In OpenPEPPOL, der Nachfolgeorganisation von PEPPOL, wird bereits an neuen BIS-Versionen sowie an einer neuen, auf AS2 (RFC 4130 – siehe http://www.ietf.org/rfc/rfc4130.txt; Stand 9.3.2014) basierenden, Übertragungstechnologie gearbeitet. Bis Ende September 2014 müssen die neuen BIS von jedem PEPPOL-Teilnehmer – also auch vom österreichischen Bund – unterstützt werden, weshalb die möglicherweise abweichenden Anforderungen auch in ERPOL nachgezogen werden müssen. Es ist jedoch der Plan von OpenPEPPOL eine funktionierende Open Source Variante basierend auf Mendelson AS2 zu veröffentlichen, wodurch hoffentlich ein einfacher Umstieg möglich sein wird.

# 7 Zusammenfassung

Die elektronische Rechnung ist einer der großen Hoffnungsträger für die Realisierung von Einsparungen in Unternehmen. Steigt der gesamte öffentliche Bereich vollständig auf e-Rechnungen um, sind alleine in Österreich jährliche Einsparungen von rund 600 Millionen Euro möglich. Mit dem ERPEL-Netzwerk und der PEPPOL-Infrastruktur stehen zwei mächtige Netzwerke im Bereich e-Procurement zur Verfügung, deren Potentiale zum heutigen Zeitpunkt bei Weitem noch nicht ausgeschöpft sind.

Das an der Technischen Universität Wien gestartete Forschungsprojekt ERPEL (E-Business Registry Permitting Enterprise Liaisons) hat sich zum Ziel gesetzt, die Heterogenität von IT-Lösungen zwischen Geschäftspartnern zu harmonisieren. Dabei wurden neben den technischen Aspekten auch prozessuale Aspekte berücksichtigt. ERPEL ist generisch spezifiziert, hat sich aber inhaltlich auf den Bereich e-Procurement fokussiert. Es wurde ein eigenes Dokumentenformat namens „ERPEL“ spezifiziert. Es basiert auf dem nationalen österreichischen e-Rechnungs-Standard „ebInterface“, unterstützt aber mehrere verschiedene e-Procurement-Anwendungsfälle. Neben dem Dokumentenformat wurden einfache Standard-Geschäftsprozesse für KMUs entworfen. Als dritte Säule wurde eine Registry entworfen deren Ziel es war, eine leichte Auffindbarkeit von Geschäftspartnern sicherzustellen. Dabei wurde besonders darauf geachtet, dass die Probleme existierender Registries (z.B. UDDI oder ebXML Registry) mit diesem Ansatz gelöst werden können. Die Vertrauenswürdigkeit von Geschäftspartnern soll dabei über den „Friend-of-a-Friend“-Ansatz sichergestellt werden. Die Übermittlung aller ERPEL-Dokumente erfolgt über den zentralen Messaging Hub. Dieser ist in der Lage Dokumente zu validieren und ggf. auch zwischenzuspeichern, falls der Empfänger zum Zeitpunkt des Einlangens des Dokuments nicht mit dem Netzwerk verbunden ist. Der sichere Austausch von Dokumenten erfolgt via Web Services mittels des Postfachprinzips, um KMUs nicht die Notwendigkeit einer eigenen IT-Infrastruktur für einen Webserver aufzubürden.

PEPPOL (Pan-European Public Procurement On-Line) war ein europäisches Projekt, dessen Ziel es war, den Austausch von elektronischen Beschaffungsdokumenten innerhalb von Europa zu vereinheitlichen. Es wurde zwischen 2009 und 2012 von 11 europäischen Ländern, darunter auch Österreich, umgesetzt. Das in PEPPOL verwendete „4-corner model“ – bestehend aus dem Rechnungssteller, dem Service Provider des Rechnungsstellers, dem Service Provider des Rechnungsempfängers und dem Rechnungsempfänger – hat sich als äußerst flexibler und vor allem praktikabler Ansatz erwiesen. Das in PEPPOL verwendete Identifikationsschema für Netzwerkteilnehmer versucht soweit wie möglich existierende Identifier (z.B. UID- oder GLN-Nummern) zu verwenden und keine neuen Identifier einzuführen. Die Sicherheit der übertragenen Dokumente wird mit elektronischen Zertifikaten, die auf einer eigenen Public Key Infrastruktur (PKI) basieren, erreicht. Das Verzeichnis sämtlicher Netzwerk-Teilnehmer wird in der zentralen SML-Komponente (Service Metadata Locator) verwaltet. Der SML dient als Frontend zum DNS (Domain Name System) und ist in den eigentlichen Datenaustausch nicht involviert. DNS ist als redundantes System konzeptioniert und ist daher kein Single Point of Failure. Der Service Metadata Publisher (SMP) ist der dezentrale Teil der PEPPOL-Registry. Ein SMP speichert pro Teilnehmer jene Prozesse und Dokumententypen, die er unterstützt und unter welcher URL der entsprechende Access Point für den Empfang von Dokumenten

erreichbar ist. Relevante Änderungen für das DNS werden vom SMP automatisiert an den SML kommuniziert, der anschließend das DNS aktualisiert.

Der Dokumentenaustausch in PEPPOL wird über Access Points (APs) durchgeführt. Diese implementieren ein auf Web Service Technologie basierendes Protokoll. Dabei werden neben den bekannten Standards wie SOAP und WSDL auch die Standards WS-Addressing, WS-Security, WS-Transfer, WS-ReliableMessaging und SAML 2.0 verwendet. Für ein funktionierendes Zusammenspiel aller genannten Technologien wurde das von allen APs zu unterstützende START-Profil entwickelt. Es definiert den Datenaustausch zwischen Service Providern im „4-corner-model“. Daneben wurde auch noch das LIME-Profil als potentielle Schnittstelle zwischen Rechnungssender bzw. -empfänger und dessen Service Provider definiert. Da die Unterstützung des LIME-Profils optional ist und in die existierenden Prozesse zwischen dem Service Provider und seinen Kunden eingreift, hat es sich kaum durchgesetzt. Zusätzlich zur Datenübertragung wurden auch Dokumenten-Validierungsregeln auf Basis von XML Schema und Schematron entwickelt. Da der Prozess der Validierung jedoch sehr ungenau spezifiziert ist, kommt es in diesem Zusammenhang immer wieder zu bilateralen Problemen. Eine der Aufgaben des als Nachfolgeorganisation von PEPPOL gegründeten *OpenPEPPOL* besteht darin, diese Ungenauigkeiten zu beseitigen und eine praxistaugliche Lösung anzubieten. Im Rahmen von PEPPOL entstand auch eine Referenzimplementierung in Java, die heute in vielen Unternehmen im Einsatz ist.

Das für die Kopplung zwischen ERPEL und PEPPOL entstandene Software-Paket hört auf den Namen ERPOL. Dieser Name setzt sich aus den Anfangsbuchstaben von „ERpel“ und dem Ende von „pepPOL“ zusammen. Durch die Umsetzung von ERPOL konnte gezeigt werden, dass die Mächtigkeit beider Netzwerke mit vertretbarem Aufwand kombinierbar ist. Zuerst wurde eine Registry-Schnittstelle, die die ERPEL Registry mit einem PEPPOL SMP verbindet, entworfen. Diese wurde mit einer proprietären REST-Schnittstelle umgesetzt, um auf ein sich eventuell änderndes SMP-Datenmodel einfacher reagieren zu können. Die ERPEL-Registry ist in diesem Fall das führende System und ruft nur bei Bedarf diese Schnittstelle auf, damit Änderungen auch im PEPPOL-Netzwerk sichtbar werden. Der SMP kümmert sich dann selbständig um die Kommunikation mit dem SML und der damit verbundenen Aktualisierung des DNS.

Die zweite von ERPOL implementierte Schnittstelle ist die des Dokumentenaustauschs. Als Schnittmenge der von ERPEL und PEPPOL unterstützten Dokumententypen wurden Transformationen von Bestellungen (order) und Rechnungen (invoice) umgesetzt. Jede Transformation muss bidirektional zwischen dem ERPEL-Format und dem passenden UBL 2.0-Dokumentenformat durchgeführt werden. Da UBL ein sehr umfangreiches Format ist, geht die Transformation nach ERPEL in den meisten Fällen mit einem Informationsverlust einher. Der eigentliche Dokumentenaustausch findet immer über JMS (Java Message Service) statt. Durch die Verwendung von einem Message Broker wird die Kommunikation entkoppelt und die starken Abhängigkeiten zwischen den Software-Komponenten reduziert. In Abbildung 66 ist die Struktur des Datenaustauschs zwischen PEPPOL und ERPEL schematisch dargestellt. Der Prozess kann entweder von links nach rechts oder von rechts nach links durchgeführt werden.


Abbildung 66: JMS Provider innerhalb der ERPOL-Datenaustauschschnittstelle

ERPOL nimmt eingehende Nachrichten aus einer JMS-Queue entgegen, validiert diese nach dem Quellformat, transformiert sie ins Zielformat, validiert das Ergebnis gegen das Zielformat und stellt sie in eine JMS-Ausgangsqueue, von wo aus die eigentliche Versendung stattfindet. Um die auszutauschenden Dokumente inklusive aller benötigten Metadaten in eine JMS-Queue stellen zu können, wurden zwei XML-Schemas mit den Namen „WrappedErpel“ und „WrappedPeppol“ entworfen. Außerdem wurden zwei XML-Schemas für die Retourkommunikation des Erfolgs bzw. der Fehler an ERPEL definiert. Das eigentliche Versenden der Dokumente im AP erfolgt über eine eigene Web-Applikation namens „cipa-start-jmssender“. Diese nimmt die Daten aus der JMS-Queue entgegen und versendet mit Hilfe der Referenzimplementierung. Das Empfangen einer Nachricht wird immer vom AP durchgeführt. Um diese Daten in eine JMS-Queue zu stellen, wurde ein AP-Plugin namens „cipa-start-jmsreceiver“ entwickelt, dass sich in die Empfangslogik des AP einhängt, die Dokumente validiert und anschließend zur Weiterverarbeitung per JMS an ERPOL sendet. Sämtliche erstellten Artefakte wurden mit Java 1.6 umgesetzt und sind plattformübergreifend einsetzbar.

Das ausführliche Testen von ERPOL war ein integraler Bestandteil der Umsetzung. Zu diesem Zweck wurde eine eigene Web-Applikation namens „erpol-web-tester“ entwickelt, mit der der Dokumentenaustausch in beide Richtungen getestet werden kann. Dabei kann jede Absender- und Empfängerseite durch eine Testinstanz ersetzt werden, um z.B. nur die Transformation oder nur die JMS-Konfiguration testen zu können.

Anhand der Case Study „Übermittlung einer e-Rechnung an den österreichischen Bund“ konnte gezeigt werden, dass die ERPOL-Schnittstelle wie erwartet funktioniert und die Anforderungen erfüllt. Es wurden dabei e-Rechnungen, die den Anforderungen des österreichischen Bundes entsprechen, von ERPEL über ERPOL an die PEPPOL-Schnittstelle von „E-Rechnung an den Bund“ gesendet und die erwartete Antwort empfangen.

## 7.1 Offene Punkte und Potential

Die Umsetzung von ERPOL hat gezeigt, dass eine bidirektionale Schnittstelle zwischen ERPEL und PEPPOL machbar ist. Durch diese Kopplung ist die Reichweite beider Netzwerke erhöht worden und beide profitieren von Erweiterungen im jeweils anderen Netzwerk. Die ERPOL Schnittstelle kann auch für die Anbindung von ERPEL oder PEPPOL an andere Netzwerke verwendet werden. Auf Grund der generischen, entkoppelten Architektur kann ERPOL die Rolle einer Mini-Middleware einnehmen, sofern entsprechende Dokumententransformationen umgesetzt werden. Es soll aber erwähnt sein, dass ERPOL auf Grund der vielen JMS-Queues und der damit verbundenen Nebenläufigkeit schwierig zu debuggen und zu testen ist (als Beispiel kann das Sequenzdiagramm aus Kapitel 5.3.3.1 betrachtet werden).

Eines der größten praktischen Probleme von PEPPOL ist, dass es keine Spezifikation für die Ablehnung von Dokumenten, die nicht den inhaltlichen Vorgaben entsprechen, gibt. Dieses

Problem könnte aber mit der OpenPEPPOL Message Level Response (MLR) gelöst werden, wenn diese als verpflichtender Dokumententyp eingeführt wird.

An folgenden Punkten wurden Grenzen innerhalb der Arbeit gezogen:

- Es wurden nur die Dokumentenarten Rechnung und Bestellung behandelt. Obwohl das die in der Praxis am weitesten verbreiteten Formate sind, wäre die Unterstützung weiterer Dokumententypen von Vorteil. Dieser Punkt ist nur langfristig zu lösen, da Rechnung und Bestellung zum jetzigen Zeitpunkt die einzigen beiden Dokumententypen sind, die von beiden Systemen unterstützt werden.
- Die Umsetzung der Transformation zwischen dem ERPEL-Format und UBL ist ausbaufähig, da beide Formate sehr umfangreich sind und viele optionale Felder enthalten. Die in ERPOL erstellten UBL-Dokumente sind derzeit für den Empfang durch die PEPPOL-Komponente von ER>B zugeschnitten. Da UBL, wie bereits erwähnt, ein sehr umfangreiches Format ist, und in PEPPOL durch die komplizierten Validierungsregeln sehr unterschiedliche UBL-Inhalte erwartet werden können. Eine vollständige Umsetzung ist in Zukunft aber machbar.

Nachfolgende Aspekte wurden, um den Rahmen dieser Arbeit nicht zu sprengen, offen gehalten:

- OpenPEPPOL hat anstelle des äußerst komplexen START-Web Service-Profils eine leichtgewichtige AS2-Schnittstelle definiert. Da diese Spezifikation aber erst fertig wurde, als die Umsetzung von ERPOL bereits fertig war, und die ersten Open Source-Komponenten mit AS2-Support erst Ende Mai 2014 veröffentlicht wurden, wurde AS2 in dieser Arbeit im Rahmen der Umsetzung nicht berücksichtigt.
- Eine weitere Neuigkeit von OpenPEPPOL ist die Aktualisierung der Business Interoperability Specifications (BIS). Diese haben aber auf diese Arbeit keine allzu großen Auswirkungen, da die neuen BIS nur inkrementelle Änderungen enthalten.
- Nicht berücksichtigt wurde auch die Übermittlung des Dokumententyps Gutschrift. In ERPEL (wie auch in ebInterface) sind Rechnung und Gutschrift in einem Dokumententyp spezifiziert und werden nur durch einen Attributwert unterschieden. In UBL ist die Gutschrift ein eigener Dokumententyp mit teilweise gravierenden Unterschieden zur Rechnung (vor allem in UBL 2.0 – mit UBL 2.1 hat sich die Situation bei Gutschriften stark verbessert). Um ERPOL produktiv einsetzen zu können muss die Gutschrift noch unterstützt werden. Einer zukünftigen Umsetzung steht auch in diesem Fall Nichts im Weg.
- Die im Rahmen von ERPOL entwickelte Test-Web-Applikation hat keine Unterstützung für die ERPOL-Registry-Schnittstelle. Da die Registry-Schnittstelle klein und synchron ist, konnten sämtliche Testfälle mit Unit-Tests abgedeckt werden.

## 7.2 Ausblick

ERPOL erfüllt fast alle Voraussetzungen für einen produktiven Einsatz. Es muss dafür nur noch die Unterstützung für Gutschriften hinzugefügt werden und die Transformation der Formate ineinander muss verbessert werden. Durch das Schnittstellenmodell mit den zusätzlichen Abstraktionsebenen (JMS und eigenes Schnittstellenformat), ist ERPOL auch gut gerüstet, um auf Veränderungen von beiden Seiten schnell reagieren zu können. Durch diese Kapselung wurde

verhindert, dass es bei Schnittstellenänderungen zu einer unnötigen Downtime von ERPEL oder dem PEPPOL AP kommt. Stattdessen wird nur die Schnittstellenlogik aktualisiert und danach können die im JMS-Provider gepufferten Nachrichten verarbeitet werden. Bei allfälligen Datenformatänderungen muss nur berücksichtigt werden, dass ERPOL beide Versionen – die alte und die neue – unterstützen muss.

Ein großer offener Punkt ist die Umstellung der PEPPOL Transport-Infrastruktur von einer Webservice-basierten Lösung auf eine AS2-basierte Lösung. AS2 ist ein HTTP-basiertes Protokoll, das mittels S/MIME verschlüsselte Nachrichten überträgt und entweder synchron oder asynchron eine MDN (Message Disposition Notification) als Empfangsbestätigung zurücksendet. Diese technische Neuerung muss in allen PEPPOL Access Points bis zum 1. September 2014 eingeführt sein, damit auch weiterhin Dokumente ausgetauscht werden können. Im Zuge dessen wird auch die Verwendung des UN/CEFACT Standard Business Document Headers (SBDH) als Containerformat verpflichtend. Dieses XML-basierte Format ersetzt die proprietären Metadaten, die derzeit als SOAP-Header übertragen werden und soll eine Vereinfachung darstellen. Allerdings halte ich das für problematisch, da damit Routing-Informationen in den Body der SOAP-Nachricht gelangen. Es muss immer die gesamte XML-Nachricht geparst werden (zeitaufwändig), um die Metadaten zu extrahieren. Weitere Neuerungen von OpenPEPPOL sind die Aktualisierung auf neue BIS Versionen und die Verwendung der Message Level Response (MLR). Die MLR ist ein spezieller Dokumententyp, der darauf abzielt, die Ergebnisse der Dokumentenübermittlung als eigenes Geschäftsdokument zurückzuliefern. Das impliziert aber, dass sowohl Sender als auch Empfänger in einem SMP registriert sind. Derzeit muss nur der Empfänger von Dokumenten registriert sein – senden kann jeder, der ein gültiges PEPPOL-Access Point-Zertifikat hat.

# 8 Danksagung

Zuallererst danke ich Karin, die über all die Jahre meines Studiums nie das Vertrauen in mich verloren hat. Danke für die Motivation, danke fürs Korrekturlesen, danke für alles!

Danke auch an Prof. Dr. Christian Huemer von der Business Informatics Group am Institute of Software Technology and Interactive Systems der TU Wien dafür, dass er mir die Möglichkeit gegeben hat, meine Arbeit zu diesem spannenden und aktuellen Thema zu schreiben.

Ein besonderer Dank gilt Marco Zapletal von der E-Commerce Group und Philipp Liegl von der Business Informatics Group - beide am Institute of Software Technology and Interactive Systems der TU Wien - die immer ein offenes Ohr für meine Anfragen und Ideen hatten. Sogar an Sonn- und Feiertagen waren sie für mich erreichbar. Danke für die konstruktive Zusammenarbeit.

Zuletzt möchte ich mich noch beim Bundesrechenzentrum bedanken, die mir die Möglichkeit gegeben haben, in so einem spannenden und zukunftsträchtigen Gebiet wie der e-Rechnung zu arbeiten. Ein besonderer Dank geht an Robert Grim dafür, dass er sich das PEPPOL-Kapitel durchgelesen und Feedback gegeben hat.

# 9 Abbildungsverzeichnis

Ich habe mich bemüht, sämtliche Inhaber der Bildrechte ausfindig zu machen und ihre Zustimmung zur Verwendung der Bilder in dieser Arbeit eingeholt. Sollte dennoch eine Urheberrechtsverletzung bekannt werden, ersuche ich um Meldung bei mir.

# 10 Literaturverzeichnis

Bausà Peris, O. (2012): Validation Artefacts for Post Award BIS. Online verfügbar unter https://joinup.ec.europa.eu/svn/peppol/PEPPOL_EIA/1-ICT_Architecture/1-ICT-PostAward_eProcurement/14-ICT-Services-Components/ICT-PostAward-BIS_Validation_Artifacts-200.pdf, zuletzt geprüft am 21.10.2012.

Borresen, P.; Pedersen, K. V. (2010): PEPPOL Business Interoperability Profile Guideline. Online verfügbar unter https://joinup.ec.europa.eu/svn/peppol/PEPPOL_EIA/1-ICT_Architecture/1-ICT-PostAward_eProcurement/13-ICT-Models/ICT-PostAward-PEPPOL_BIS_Guideline-100.pdf, zuletzt geprüft am 19.10.2012.

Bundesministerium für Finanzen: ER>B - Vorteile der e-Rechnung. Online verfügbar unter https://www.erb.gv.at/erb?p=info_adv, zuletzt geprüft am 08.06.2014.

Deutsche Bank Research (2010): E-Invoicing - Krönung einer effizienten Rechnungsbearbeitung. Online verfügbar unter http://www.dbresearch.de/PROD/DBR_INTERNET_DE-PROD/PROD0000000000253980.pdf, zuletzt geprüft am 07.06.2014.

EUROPEAN COMMISSION (2010): Annex 2 to the Communication from the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of Regions 'Towards interoperability for European public services'. Online verfügbar unter http://ec.europa.eu/isa/documents/isa_annex_ii_eif_en.pdf, zuletzt geprüft am 19.10.2012.

FeRD (2013): ZUGFeRD-Format Release Candidate. Online verfügbar unter http://www.ferd-net.de/upload/ZuGFeRD-Infopaket.zip, zuletzt geprüft am 29.03.2014.

Gundel, T.; Pedersen, K. V. (2011): Trust Network Certificate Policy. Online verfügbar unter https://joinup.ec.europa.eu/svn/peppol/PEPPOL_EIA/1-ICT_Architecture/1-ICT-Transport_Infrastructure/13-ICT-Models/ICT-Transport-Trust_Network_Certificate_Policy-100.pdf, zuletzt geprüft am 19.10.2012.

Hedler, M.; Kutscherauer, N.; Montero Pineda, M.; Zaeri, M. (2011): Schematron. Effiziente XML Business Rules für XML-Dokumente, 1. Auflage, Heidelberg: dpunkt-Verlag, ISBN: 978-3-89864-721-2.

Helger, P. (2012a): PEPPOL-Silicone How to deploy. Online verfügbar unter https://peppol-silicone.googlecode.com/svn/trunk/docs/v2.3.1/ICT-Transport-Silicone_Java_How_to_deploy-231.pdf, zuletzt geprüft am 25.09.2013.

Helger, P. (2012b): PEPPOL-Silicone SMP Developer Guide. Online verfügbar unter https://peppol-silicone.googlecode.com/svn/trunk/docs/v2.3.1/ICT-Transport-Silicone_SMP_Developer_Guide-231.pdf, zuletzt geprüft am 25.09.2013.

Helger, P. (2012c): PEPPOL-Silicone START AP Developer Guide. Online verfügbar unter https://peppol-silicone.googlecode.com/svn/trunk/docs/v2.3.1/ICT-Transport-Silicone_START_Developer_Guide-231.pdf, zuletzt geprüft am 25.09.2013.

Helger, P. (2013): ER>B aus technischer Sicht. Online verfügbar unter https://www.erb.gv.at/files/ERB-technisch-2013-06-27.pdf, zuletzt geprüft am 12.01.2014.

Helger, P.; Rasmussen, S. (2012): Policy for use of Identifier v2.2. Online verfügbar unter https://joinup.ec.europa.eu/svn/peppol/PEPPOL_EIA/1-ICT_Architecture/1-ICT-Transport_Infrastructure/13-ICT-Models/ICT-Transport-Policy_for_using_Identifiers-220.pdf, zuletzt geprüft am 19.10.2012.

Hevner, A. R.; March, S. T.; Park, J.; Ram, S. (2004): Design Science in Information Systems Research. In: Management Information Systems Research Center, Carlson School of Management, University of Minnesota (Hg.): MIS Quarterly, Bd. 28, S. 75–105.

Hofreiter, B.; Huemer, C.; Klas, W. (2002): ebXML: Status, Research Issues and Obstacles. In: Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems, 2002. RIDE-2EC 2002., San Jose, CA, 24.2.-25.2.2002, IEEE, S. 7–16. Online verfügbar unter http://eprints.cs.univie.ac.at/1203/, zuletzt geprüft am 30.08.2014.

Huemer, C.; Kappel, G.; Krenn, P.; Liegl, P.; Mayrhofer, D.; Schuster, R.; Topf, M.; Werthner, H.; Zapletal, M. (2012): ERPEL: Enabling Seamless Ad-hoc Cross Enterprise Collaborations. In: SRII Global Conference (SRII), 2012 Annual, Bd. 2012, San Jose, California, USA, 24.7.2012-27.7.2012, SRII, S. 478–487.

Liegl, P.; Zapletal, M.; Ilger, G. (2013): Erpel Wiki. Online verfügbar unter http://doku.erpel.at, zuletzt geprüft am 01.02.2013.

Liegl, P.; Zapletal, M.; Pichler, C.; Strommer, M. (2010): State-of-the-art in business documents standards. In: 2010 8th IEEE International Conference on Industrial Informatics (INDIN), Osaka, Japan, 13.7.2010-16.7.2010, IEEE, S. 234–241.

Motal, T.; Zapletal, M.; Werthner, H. (2009): The Business Choreography Language (BCL). A Domain-Specific Language for Global Choreographies. In: IEEE World Congress on Services 2009, Bangalore, India, 21.09.2009-25.09.2009, IEEE, S. 150–159. Online verfügbar unter http://www.isis.tuwien.ac.at/node/13675, zuletzt geprüft am 30.08.2014.

National Institute of Standards and Technology (1996): FIPS PUB 161-2. Online verfügbar unter http://niatec.info/GetFile.aspx?pid=556, zuletzt geprüft am 23.07.2014.

OASIS (2011): Business Document Exchange Architecture. Online verfügbar unter https://www.oasis-open.org/committees/download.php/41668/Business_Document_Exchange_Architecture_BEDA_v0p3.pdf, zuletzt geprüft am 19.10.2012.

OASIS (2012): Business Document Exchange (BDXR) TC. Online verfügbar unter https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=bdxr, zuletzt geprüft am 16.09.2012.

Pedersen, K. V.; Andersen, J. J. (2011): ICT Architecture Strategy. Online verfügbar unter https://joinup.ec.europa.eu/svn/peppol/PEPPOL_EIA/1-ICT_Architecture/1-ICT-Transport_Infrastructure/11-ICT-Strategy/ICT-Transport-ICT_Architecture_Strategy-100.pdf, zuletzt geprüft am 19.10.2012.

Pedersen, K. V.; Skulason, B. (2012): PEPPOL PostAward eProcurement ICT Architecture Models. Online verfügbar unter https://joinup.ec.europa.eu/svn/peppol/PEPPOL_EIA/1-ICT_Architecture/1-ICT-PostAward_eProcurement/13-ICT-Models/ICT-PostAward-ICT_Architecture_Models-120.pdf, zuletzt geprüft am 21.10.2012.

PEPPOL (2012): PEPPOL Elements. Online verfügbar unter http://www.peppol.eu/peppol_elements, zuletzt geprüft am 26.09.2012.

Skulason, B.; Pedersen, K. V. (2012a): BIS 4a – Basic Invoice Only. Online verfügbar unter https://joinup.ec.europa.eu/svn/peppol/PEPPOL_EIA/1-ICT_Architecture/1-ICT-PostAward_eProcurement/13-ICT-Models/ICT-PostAward-PEPPOL_BIS_4a-301.pdf, zuletzt geprüft am 19.10.2012.

Skulason, B.; Pedersen, K. V. (2012b): BIS 5a – Billing. Online verfügbar unter https://joinup.ec.europa.eu/svn/peppol/PEPPOL_EIA/1-ICT_Architecture/1-ICT-PostAward_eProcurement/13-ICT-Models/ICT-PostAward-PEPPOL_BIS_5a-310.pdf, zuletzt geprüft am 19.10.2012.

Skulason, B.; Pedersen, K. V. (2012c): BIS 6a – Procurement. Online verfügbar unter https://joinup.ec.europa.eu/svn/peppol/PEPPOL_EIA/1-ICT_Architecture/1-ICT-PostAward_eProcurement/13-ICT-Models/ICT-PostAward-PEPPOL_BIS_6a-200.pdf, zuletzt geprüft am 19.10.2012.

Skulason, B.; Pedersen, K. V.; Leutgeb, A. (2012): BIS 3a – Basic Order Only. Online verfügbar unter https://joinup.ec.europa.eu/svn/peppol/PEPPOL_EIA/1-ICT_Architecture/1-ICT-PostAward_eProcurement/13-ICT-Models/ICT-PostAward-PEPPOL_BIS_3a-301.pdf, zuletzt geprüft am 19.10.2012.

Snyder, B.; Bosnanac, D.; Davies, R. (2011): ActiveMQ in action, Stamford, Ct: Manning, ISBN: 1933988940.

Stefano, G. de; Bertini, L. (2012): BIS 1a – Catalogue only. Online verfügbar unter https://joinup.ec.europa.eu/svn/peppol/PEPPOL_EIA/1-ICT_Architecture/1-ICT-PostAward_eProcurement/13-ICT-Models/ICT-PostAward-PEPPOL_BIS_1a-300.pdf, zuletzt geprüft am 19.10.2012.

Stefano, G. de; Bertini, L.; Gromholt, K. (2012): BIS 12a - eCatalogues as part of a Tender. Online verfügbar unter https://joinup.ec.europa.eu/svn/peppol/PEPPOL_EIA/1-ICT_Architecture/1-ICT-PreAward_eProcurement/13-ICT-Models/ICT-PreAward-eCAT-PEPPOL BIS 12a_Specification-131.doc, zuletzt geprüft am 21.10.2012.

Sylvest, G.; Andersen, J. J.; Pedersen, K. V.; Brun, M. H. (2010a): BusDox Common Definitions. Online verfügbar unter https://joinup.ec.europa.eu/svn/peppol/PEPPOL_EIA/1-ICT_Architecture/1-ICT-Transport_Infrastructure/13-ICT-Models/ICT-Transport-BusDox_Definitions-101.pdf, zuletzt geprüft am 19.10.2012.

Sylvest, G.; Andersen, J. J.; Pedersen, K. V.; Brun, M. H.; Edwards, M. (2010b): Service Metadata Locator (SML). Online verfügbar unter https://joinup.ec.europa.eu/svn/peppol/PEPPOL_EIA/1-ICT_Architecture/1-ICT-Transport_Infrastructure/13-ICT-Models/ICT-Transport-SML_Service_Specification-101.pdf, zuletzt geprüft am 19.10.2012.

Sylvest, G.; Andersen, J. J.; Pedersen, K. V.; Brun, M. H.; Fremantle, P. (2010c): Lightweight Message Exchange (LIME). Online verfügbar unter https://joinup.ec.europa.eu/svn/peppol/PEPPOL_EIA/1-ICT_Architecture/1-ICT-Transport_Infrastructure/13-ICT-Models/ICT-Transport-LIME_Specification-101.pdf, zuletzt geprüft am 19.10.2012.

Sylvest, G.; Andersen, J. J.; Pedersen, K. V.; Brun, M. H.; Fremantle, P. (2010d): Secure Trusted Asynchronous Reliable Transport (START). Online verfügbar unter https://joinup.ec.europa.eu/svn/peppol/PEPPOL_EIA/1-ICT_Architecture/1-ICT-Transport_Infrastructure/13-ICT-Models/ICT-Transport-START_Service_Specification-101.pdf, zuletzt geprüft am 19.10.2012.

Sylvest, G.; Andersen, J. J.; Pedersen, K. V.; Brun, M. H.; Fremantle, P. (2010e): Service Metadata Publishing (SMP). Online verfügbar unter https://joinup.ec.europa.eu/svn/peppol/PEPPOL_EIA/1-ICT_Architecture/1-ICT-Transport_Infrastructure/13-ICT-Models/ICT-Transport-SMP_Service_Specification-101.pdf, zuletzt geprüft am 19.10.2012.

TU Vienna, Institute of Software Technology and Interactive Systems (2009): FIT-IT Projektantrag, Wien.

# 11 Glossar

| Name | Bedeutung | Beschreibung |
|---|---|---|
| **AP** | Access Point | Adapter einer Business-Anwendung um PEPPOL Dokumente austauschen zu können |
| **AS2** | Application Statement 2 | Ein Übertragungsprotokoll für den elektronischen Datenaustausch, welches in OpenPEPPOL anstelle von START verwendet werden soll |
| **B2B** | Business to Business | Beschreibt den Austausch von Daten zwischen zwei Firmen |
| **B2G** | Business to Government | Beschreibt den Austausch von Daten zwischen einer Firma und einer öffentlichen Einrichtung |
| **BIS** | Business Interoperability Specification | Definiert einen PEPPOL Standardprozess inkl. der Geschäftssemantik und dem Datenaustauschformat |
| **BPMN** | Business Process Model and Notation | Grafische Spezifikationssprache für Geschäftsprozesse |
| **BSI** | Business Service Interface | Ein technischer Adapter in ERPEL der in bestehende Software integriert wird um bestimmte Dokumente austauschen zu können |
| **CCTS** | UN/CEFACT Core Components Technical Specification | Ein Beispiel für einen syntaxneutralen Dokumentenaustauschstandard |
| **CEN** | European Committee for Standardization | Europäisches Standardisierungsgremium |
| **CWA** | CEN Workshop Agreement | Publizierte Ergebnisse eines CEN Workshops |
| **DNS** | Domain Name System | Zentrales Service um Namen in IP-Adressen umzuwandeln |
| **DSL** | Domain Specific Language | Eine für einen bestimmten Zweck entwickelte Programmiersprache |
| **ebXML** | Electronic Business using XML | Ein XML Dialekt für den Austausch von Dokumenten in elektronischen Geschäftsprozessen |
| **EC** | European Commission | Die Europäische Kommission |
| **EDI** | Electronic Data Interchange | Der Überbegriff für den elektronischen Datenaustausch |
| **ER>B** | E-Rechnung an den Bund | Die Webseite des österreichischen Bundes auf der e-Rechnungen eingebracht werden müssen |
| **ERP** | Enterprise Resource Planning | Eine Anwendungssoftware für die Verwendung in Unternehmen |
| **ERPEL** | E-business Registry Permitting Enterprise Liaisons | Forschungsprojekt der TU Wien zum Austausch von elektronischen Dokumenten in heterogenen Systemen |
| **EUPL** | European Union Public License | Eine Open Source Software Lizenz der Europäischen Union |
| **IKTKonG** | IKT-Konsolidierungsgesetz | Das österreichische Gesetzt das u.a. die Verpflichtung zum Versand von e-Rechungen |

| | | |
|---|---|---|
| | | an den Bund definiert. |
| **JAXB** | Java Architecture for XML Binding | Java-Standard-Bibliothek zum Lesen und Schreiben von XML-Dokumenten, basierend auf POJOs. |
| **JDK** | Java Development Kit | Entwicklungsumgebung für Java-Applikationen |
| **JMS** | Java Message Service | Java Standardtechnologie zum Austausch von Daten über Applikationsgrenzen hinweg |
| **JRE** | Java Runtime Environment | Laufzeitumgebung für Java-Applikationen |
| **KMU** | Kleine und Mittlere Unternehmen | Definierte die Gruppe der kleinen und mittleren Unternehmen (< 250 Beschäftigte und Jahresumsatz < 50 Millionen €) |
| **LIME** | Lightweight Message Exchange | PEPPOL Protokoll zum Austausch von Dokumenten. Abgespeckte Variante von START |
| **MDN** | Message Disposition Notification | Eine in AS2 spezifizierte Empfangsquittung |
| **MLR** | Message Level Response | Ein von OpenPEPPOL eingeführter Dokumententyp zur inhaltlichen Empfangsquittierung |
| **MPL** | Mozilla Public License | Eine Open Source Software Lizenz von der Mozilla Foundation |
| **OASIS** | Organization for the Advancement of Structured Information Standards | Non-Profit Organisation zur Pflege von Standards im IT-Bereich |
| **OSS** | Open Source Software | Allgemeine Bezeichnung für quelloffene Software |
| **OWL** | Web Ontology Language | XML basierte Beschreibungssprache für Ontologien |
| **PEPPOL** | Pan-European Public Procurement On-Line | Europäisches Projekt zur Entwicklung und Umsetzung von technologischen Standards zum einheitlichen Austausch von elektronischen Beschaffungsdokumenten mit europäischen Regierungsbehörden |
| **POJO** | Plain Old Java Object | Definiert ein „ganz normales“ Java-Objekt im Vergleich zu einem Java Entity-Bean |
| **SML** | Service Metadata Locator | Zentraler PEPPOL Service zum Schreiben und Löschen von DNS-Einträgen |
| **SMP** | Service Metadata Publishing | PEPPOL „Registry“ wo die Endpunkt-Informationen zum Dokumentenaustausch gespeichert werden |
| **SPI** | Service Provider Interface | Ein Java-Standard-Verfahren zur losen Kopplung von Komponenten |
| **START** | Secure Trusted Asynchronous Reliable Transport | WebService basiertes Protokoll zum sicheren Austausch von PEPPOL Dokumenten. Muss von jedem AP unterstützt werden |
| **SVN** | Subversion | Eine Versionierungs-Software die speziell in der Softwareentwicklung häufig eingesetzt wird |
| **UBL** | Universal Business Language | Ein XML Dialekt für den Austausch von Dokumenten in elektronischen Geschäftsprozessen |

| | | |
|---|---|---|
| **UDDI** | Universal Description, Discovery and Integration | Ein Verzeichnisdienst für Web Services |
| **UMM** | UN/CEFACT Modelling Methodology | Modellierungsmethodik für B2B Geschäftsprozesse |
| **UN/CEFACT** | United Nations Electronic Data Interchange for Administration, Commerce and Transport | Ein altes und bekanntes EDI Datenaustauschformat der Vereinten Nationen |
| **USP** | Unternehmensserviceportal | Ein Webseite des österreichischen Bundes auf der man sich anmelden muss um e-Rechnungen an den Bund senden zu können |
| **UStG** | Umsatzsteuergesetz | Das österreichische Gesetz das u.a. den Inhalt einer gültigen Rechnung definiert. |
| **WSDL** | Web Service Definition Language | XML-Dialekt zur Beschreibung von Web Service Schnittstellen |
| **WSRM** | Web Service Reliable Messaging | Standard zum verlässlichen Austausch von Web Service Nachrichten |
| **XML** | Extensible Markup Language | Eine strukturierte Auszeichnungssprache zur Darstellung von strukturierten hierarchischen Daten |
| **XSLT** | Extensible Stylesheet Language Transformation | XML-Dialekt zur Beschreibung von Transformationen zwischen XML-Dokumenten |

# 12 Appendix A – XML-Schemas

## 12.1 WrappedErpel XML-Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns="http://www.helger.com/ns/erpel-wrapped/1.0"
           targetNamespace="http://www.helger.com/ns/erpel-wrapped/1.0"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified"
           xmlns:erpel="http://erpel.at/schemas/1p0/documents">
  <xs:annotation>
    <xs:documentation>
      This schema defines a wrapped ERPEL request.
      It contains metadata and the ERPEL-XML payload.
      It is used for both directions: ERPEL -&gt; PEPPOL and PEPPOL -&gt;
ERPEL
      Last modification: 2013-04-28, Philip Helger
    </xs:documentation>
  </xs:annotation>

  <xs:import namespace="http://erpel.at/schemas/1p0/documents"
             schemaLocation="include/documents.xsd" />

  <xs:complexType name="WrappedErpelType">
    <xs:sequence>
      <xs:element name="MessageID" type="xs:string">
        <xs:annotation>
          <xs:documentation>The PEPPOL Message ID</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="SenderID" type="xs:string">
        <xs:annotation>
          <xs:documentation>The PEPPOL identifier of the
sender</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="RecipientID" type="xs:string">
        <xs:annotation>
          <xs:documentation>The PEPPOL identifier of the
receiver</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="DocumentTypeID" type="xs:string">
        <xs:annotation>
          <xs:documentation>The PEPPOL document type
identifier</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="ProcessID" type="xs:string">
        <xs:annotation>
          <xs:documentation>The PEPPOL process identifier</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Document" type="erpel:DocumentType">
        <xs:annotation>
          <xs:documentation>The ERPEL document</xs:documentation>
```

```
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="WrappedErpel" type="WrappedErpelType">
    <xs:annotation>
      <xs:documentation>The root element</xs:documentation>
    </xs:annotation>
  </xs:element>
</xs:schema>
```

## 12.2 ERPEL2PEPPOL Error XML-Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns="http://www.helger.com/ns/erpel2peppol-error/1.0"
           targetNamespace="http://www.helger.com/ns/erpel2peppol-error/1.0"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
  <xs:annotation>
    <xs:documentation>
      This schema defines the error for a wrapped ERPEL request.
      Last modification: 2013-04-26, Philip Helger
    </xs:documentation>
  </xs:annotation>

  <xs:complexType name="Erpel2PeppolErrorType">
    <xs:sequence>
      <xs:element name="MessageID" type="xs:string">
        <xs:annotation>
          <xs:documentation>The PEPPOL and ERPEL Message ID - created by
ERPEL</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="ErrorID" type="xs:string">
        <xs:annotation>
          <xs:documentation>The ERPOL error ID</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="SubErrorID" type="xs:string" minOccurs="0">
        <xs:annotation>
          <xs:documentation>The ERPOL nested error ID (e.g. from
transformation)</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="ErrorMessage" type="xs:string">
        <xs:annotation>
          <xs:documentation>The human readable error
message</xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="Erpel2PeppolError" type="Erpel2PeppolErrorType">
    <xs:annotation>
      <xs:documentation>The root element</xs:documentation>
    </xs:annotation>
  </xs:element>
```

```
</xs:schema>
```

## 12.3 ERPEL2PEPPOL Success XML-Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns="http://www.helger.com/ns/erpel2peppol-success/1.0"
           targetNamespace="http://www.helger.com/ns/erpel2peppol-success/1.0"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
  <xs:annotation>
    <xs:documentation>
      This schema defines the success for a wrapped ERPEL request.
      Last modification: 2013-04-26, Philip Helger
    </xs:documentation>
  </xs:annotation>

  <xs:complexType name="Erpel2PeppolSuccessType">
    <xs:sequence>
      <xs:element name="MessageID" type="xs:string">
        <xs:annotation>
          <xs:documentation>The PEPPOL and ERPEL Message ID - created by
ERPEL</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="ProcessingTime" type="xs:int" minOccurs="0">
        <xs:annotation>
          <xs:documentation>The overall processing time in milli
seconds</xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="Erpel2PeppolSuccess" type="Erpel2PeppolSuccessType">
    <xs:annotation>
      <xs:documentation>The root element</xs:documentation>
    </xs:annotation>
  </xs:element>
</xs:schema>
```

# 13 Deutscher Abstract

Die elektronische Rechnung ist einer der großen Hoffnungsträger für die Realisierung von Einsparungen in Unternehmen. Steigt der gesamte öffentliche Bereich vollständig auf e-Rechnungen um, sind alleine in Österreich jährliche Einsparungen von rund 600 Millionen Euro möglich. Mit dem ERPEL-Netzwerk und der PEPPOL-Infrastruktur stehen zwei mächtige Netzwerke im Bereich e-Procurement zur Verfügung.

Das an der technischen Universität Wien gestartete Forschungsprojekt ERPEL (E-business Registry Permitting Enterprise Liaisons) hat sich zum Ziel gesetzt, die Heterogenität von IT-Lösungen zwischen Geschäftspartnern zu harmonisieren. Dabei wurden neben den technischen Aspekten auch prozessuale Aspekte berücksichtigt. ERPEL ist generisch spezifiziert, hat sich aber inhaltlich auf den Bereich e-Procurement fokussiert. Es wurde ein eigenes Dokumentenformat namens „ERPEL“ spezifiziert und einfache Standard-Geschäftsprozesse für KMUs entworfen. Als dritte Säule wurde eine Registry entworfen deren Ziel es war, eine leichte Auffindbarkeit von Geschäftspartnern sicherzustellen. Dabei wurde besonders darauf geachtet, dass die Probleme existierender Registries mit diesem Ansatz gelöst werden können. Die Vertrauenswürdigkeit von Geschäftspartnern soll dabei über den „Friend-of-a-Friend“-Ansatz sichergestellt werden. Die Übermittlung aller ERPEL-Dokumente erfolgt über den zentralen Messaging Hub. Dieser ist in der Lage Dokumente zu validieren und ggf. auch zwischenzuspeichern, falls der Empfänger zum Zeitpunkt des Einlangens des Dokuments nicht mit dem Netzwerk verbunden ist. Der sichere Austausch von Dokumenten erfolgt via Web Services über SMTP, um KMUs nicht die Notwendigkeit einer eigenen IT-Infrastruktur für einen Webserver aufzubürden.

PEPPOL (Pan-European Public Procurement On-Line) war ein europäisches Projekt, dessen Ziel es war, den Austausch von elektronischen Beschaffungsdokumenten innerhalb von Europa zu vereinheitlichen. Es wurde zwischen 2009 und 2012 von 11 europäischen Ländern, darunter auch Österreich, umgesetzt. Das in PEPPOL verwendete „4-corner model“ hat sich als äußerst flexibler und vor allem praktikabler Ansatz erwiesen. Das verwendete Identifikationsschema für Netzwerkteilnehmer versucht soweit wie möglich existierende Identifier zu verwenden und keine neuen Identifier einzuführen. Die Sicherheit der übertragenen Dokumente wird mit elektronischen Zertifikaten, die auf einer eigenen PKI basieren, erreicht. Das Verzeichnis sämtlicher Netzwerk-Teilnehmer wird in der zentralen SML-Komponente verwaltet. Der SML dient als Frontend zum DNS und ist in den eigentlichen Datenaustausch nicht involviert. Der SMP ist der dezentrale Teil der PEPPOL-Registry. Ein SMP speichert pro Teilnehmer jene Prozesse und Dokumententypen, die er unterstützt und unter welcher URL der entsprechende Access Point für den Empfang von Dokumenten erreichbar ist. Relevante Änderungen für das DNS werden vom SMP automatisiert an den SML kommuniziert, der anschließend das DNS aktualisiert.

Der Dokumentenaustausch in PEPPOL wird über Access Points (APs) durchgeführt. Diese implementieren ein auf Web Service Technologie basierendes Protokoll. Für ein funktionierendes Zusammenspiel aller verwendeten Technologien wurde das von allen APs zu unterstützende START-Profil entwickelt. Es definiert den Datenaustausch zwischen Service Providern im „4-corner-model“. Zusätzlich zur Datenübertragung wurden auch Dokumenten-Validierungsregeln auf Basis von XML Schema und Schematron entwickelt. Im Rahmen von

PEPPOL entstand auch eine Referenzimplementierung in Java, die heute in vielen Unternehmen im Einsatz ist.

Das für die Kopplung zwischen ERPEL und PEPPOL entstandene Software-Paket hört auf den Namen ERPOL. Durch die Umsetzung von ERPOL konnte gezeigt werden, dass die Mächtigkeit beider Netzwerke mit vertretbarem Aufwand kombinierbar ist. Zuerst wurde eine Registry-Schnittstelle, die die ERPEL Registry mit einem SMP verbindet, entworfen. Diese wurde mit einer proprietären REST-Schnittstelle umgesetzt, um auf ein sich eventuell änderndes SMP-Datenmodel einfacher reagieren zu können. Die ERPEL-Registry ist in diesem Fall das führende System und ruft nur bei Bedarf diese Schnittstelle auf, damit Änderungen auch im PEPPOL-Netzwerk sichtbar werden. Der SMP kümmert sich dann selbständig um die Kommunikation mit dem SML und der damit verbundenen Aktualisierung des DNS.

Die zweite von ERPOL implementierte Schnittstelle ist die des Dokumentenaustauschs. Als Schnittmenge der von ERPEL und PEPPOL unterstützten Dokumententypen wurden bidirektionale Transformationen von Bestellungen und Rechnungen umgesetzt. Der eigentliche Dokumentenaustausch findet immer über JMS statt. Durch die Verwendung von einem Message Broker wird die Kommunikation entkoppelt und die starken Abhängigkeiten zwischen den Software-Komponenten reduziert.

ERPOL nimmt eingehende Nachrichten aus einer JMS-Queue entgegen, validiert diese nach dem Quellformat, transformiert sie ins Zielformat, validiert das Ergebnis gegen das Zielformat und stellt sie in eine JMS-Ausgangsqueue, von wo aus die eigentliche Versendung stattfindet. Das eigentliche Versenden der Dokumente im AP erfolgt über eine eigene Web-Applikation, die die Daten aus der JMS-Queue entgegen nimmt und mit Hilfe der Referenzimplementierung versendet. Das Empfangen einer Nachricht wird immer vom AP durchgeführt. Um diese Daten in eine JMS-Queue zu stellen, wurde ein AP-Plugin entwickelt, dass sich in die Empfangslogik des AP einhängt, die Dokumente validiert und anschließend zur Weiterverarbeitung per JMS an ERPOL sendet. Sämtliche Artefakte wurden mit Java 1.6 umgesetzt und sind plattformübergreifend einsetzbar.

Zum Testen von ERPOL wurde eine eigene Web-Applikation entwickelt, mit der der Dokumentenaustausch in beide Richtungen getestet werden kann. Dabei kann jede Absender- und Empfängerseite durch eine Testinstanz ersetzt werden, um z.B. nur die Transformation oder nur die JMS-Konfiguration testen zu können.

Anhand der Case Study „Übermittlung einer e-Rechnung an den österreichischen Bund“ konnte gezeigt werden, dass die ERPOL-Schnittstelle wie erwartet funktioniert und die Anforderungen erfüllt. Es wurden dabei e-Rechnungen, die den Anforderungen des österreichischen Bundes entsprechen, von ERPEL über ERPOL an die PEPPOL-Schnittstelle von „E-Rechnung an den Bund“ gesendet und die erwartete Antwort empfangen.

# 14 English Abstract

Electronic invoicing is one of the main areas where major savings can be achieved in modern business entities. If the whole public sector in Austria would switch to electronic invoicing, yearly savings up to 600 million Euros could be acquired. The ERPEL network and the PEPPOL project are two powerful examples of electronic networks which exploit great potentials in this area.

ERPEL (E-business Registry Permitting Enterprise Liaisons) is a research project at the Vienna University of Technology. Its goal is to reduce the heterogeneities of IT solutions between business partners considering both technical and procedural aspects. ERPEL is designed in a generic way but focuses on e-Procurement. It defines its own “ERPEL” document type and simple default business processes for SMEs. ERPEL also designed a registry aiming for easy finding of business partners. A special focus was put on the avoidance of the errors made in other registries. Trust between business partners is ensured with the “friend-of-a-friend” model. The transmission of all ERPEL documents happens centrally via the so called “Messaging Hub”. Its purpose is to route and validate documents and to temporarily save them if the receiver is not available to the network at the time of transmission. Technically the transmission happens with Web Services to avoid fostering SMEs to build up their own IT infrastructure.

PEPPOL (Pan-European Public Procurement On-Line) was a European project aiming to harmonize the exchange of e-Procurement documents within Europe. It was implemented between 2009 and 2012 by 11 European countries including Austria. The “4-corner-model” has proven to be a flexible and feasible solution for PEPPOL’s document exchange. The identification scheme for participants reuses existing and avoids assigning new identifiers where possible. Security of transmitted documents is ensured with digital signatures that are based on a special PEPPOL PKI. The list of all PEPPOL participants is managed by the central SML component that is a frontend to the DNS. The SML is only invoked during the creation, modification or deletion of a participant but not during a document transmission. The SMP is the decentralized part of the PEPPOL registry storing the supported processes, document types and Access Point endpoint URLs on a per-participant base. DNS relevant changes in an SMP are automatically communicated to the SML which in turn updates the DNS.

The main document transmission in PEPPOL is achieved by using Access Points (APs). They are implementing a Web service based protocol for achieving an asynchronous, secure and non-repudiable document transmission. The assembly of all these Web service standards was specified by PEPPOL as the “START” profile. PEPPOL also defined document validation rules based on XML Schema and Schematron. Finally within PEPPOL a reference implementation of all components was created that is still in use in many companies.

The ERPEL-PEPPOL coupling artifact created in this thesis is called ERPOL. By creating ERPOL it was shown that both networks are easily connectable with reasonable effort. First an interface from the ERPEL registry to the PEPPOL SMP was designed. It was implemented using a custom REST interface to add flexibility for an eventually changing data model within the SMP. The ERPEL registry is the leading system and invokes the SMP interface only when necessary. The SMP than communicates with the SML which in turn updates the DNS.

Besides the registry interface ERPOL contains a document exchange interface for e-Invoices and e-Orders including the transformation between the ERPEL format and the UBL format. Since UBL is a very large and complex format the transformation from UBL may incorporate some loss of information due to missing fields in the receiver format. The main document exchange happens via JMS (Java Message Service) which is an excellent tool to decouple the communication and reduce strong dependencies between software components. Incoming documents are validated, transformed and validated against the target format by ERPOL before they are passed to the outbound JMS queue. The main transmission of documents via PEPPOL happens through a separate web application which takes a message from a JMS queue and sends it using the facilities of the reference implementation. The receiving of PEPPOL messages is done by a special software plugin of the Access Point which puts incoming messages into a JMS queue for further processing by ERPOL. All ERPOL artifacts were developed using Java 1.6 and can be used platform agnostic.

Testing of ERPOL was an integral part of the development. Besides unit tests a separate web application was developed that helps in testing the document exchange in both directions. Both sender and receiver side can be replaced with mock objects to test ERPOL in standalone environments.

Finally the case study “Sending an e-Invoice to the Austrian government” has shown that ERPOL works as designed and is capable of transmitting an ERPEL invoice via PEPPOL to the Austrian government.

# 15 Lebenslauf

### *Persönliche Daten*

Name: Philip Helger, Bakk. rer. soc. oec.

E-Mail-Adresse: philip@helger.com

Geburtsdatum: 6. Juli 1978, Freiburg im Breisgau/Deutschland

Staatsbürgerschaft: Deutschland

### *Schulische Ausbildung*

08/1984 - 07/1988 Anne-Frank-Grundschule, Freiburg i. Brsg./Deutschland

08/1988 - 07/1989 Wentzinger Gymnasium, Freiburg i. Brsg./Deutschland

09/1989 - 07/1990 Radetzky Gymnasium, Wien 3

09/1990 - 07/1996 Realgymnasium Hollabrunn, abgeschlossen mit Matura (guter Erfolg)

### *Universitäre Ausbildung*

09/1996 - 11/2005 Studium der Wirtschaftsinformatik an der Universität Wien. Zwei Bakkalaureatsarbeiten mit den Titeln „Modellierungsmethode für XPDL und XPDL-Generierung“ sowie „UMM BTV nach WS-CDL Transformator“. Abschluss mit dem Titel „Bakk. rer. soc. oec.“

11/2005 – dato Magisterstudium der Wirtschaftsinformatik an der Universität Wien. 2012 Umstieg auf Masterstudium mit neuem Studienplan.

### *Beruflicher Werdegang*

04/1997 - 03/2006 BOC GmbH, Software Architekt und Entwickler für ADONIS

10/2001 - 06/2005 Tutor an der Universität Wien für die Fächer „Unternehmensmodellierung und Business Engineering“ sowie „Projektpraktikum im betrieblichen Umfeld“

03/2006 - dato Bundesrechenzentrum GmbH, Senior Software Architect – E-Rechnung an den Bund, PEPPOL, Middleware, Jobsteuerung

09/2006 – 09/2014 Gründung der „Gregorcic & Helger – IT systems OG“ (phloc systems) als Systemhaus. Tätigkeit als Geschäftsführer und Projektmanager.