ELO XML Importer

[Stand: 07.06.2013 | Programmversion : 8.00.010.002 ab Java 1.6]

Das Programm ELO XML Importer ist ein Servlet für den Massendatenimport in ein ELO Archiv. Hierzu werden in einem Kommunikationsverzeichnis die Dokumente zusammen mit einer XML Steuerdatei hinterlegt. Das Servlet kontrolliert in bestimmten Abständen dieses Verzeichnis und arbeitet die neuen Steuerdateien ab. Dabei werden Dokumente und Steuerdateien mit fortschreitender Abarbeitung gelöscht. Falls es dabei zu einer Störung kommt, versucht das Servlet unbegrenzt lange erneut. Ein Neustart ist nicht notwendig.

Inhalt

1 Allgemeines

1.1 Aufgabe des Programms

Das Programm ELO XML Importer ist ein Servlet für den Massendatenimport in ein ELO Archiv. Hierzu werden in einem Kommunikationsverzeichnis die Dokumente zusammen mit einer XML Steuerdatei hinterlegt. Das Servlet kontrolliert in bestimmten Abständen dieses Verzeichnis und arbeitet die neuen Steuerdateien ab. Dabei werden Dokumente und Steuerdateien mit fortschreitender Abarbeitung gelöscht. Falls es dabei zu einer Störung kommt, versucht das Servlet unbegrenzt lange erneut. Ein Neustart ist nicht notwendig.

1.2 Programminstallation

Das Programm ist im ELO Installationsprogramm enthalten. Für den Betrieb muss bereits ein Indexserver für das gewünschte Archiv installiert sein, nicht zwangsläufig auf demselben Rechner/Server. Installiert wird nur ein einfach konfigurierter XML-Importer, dessen Konfiguration in der Datei config.xml (im Installationsverzeichnis z.B. ELOenterprise\config\im-Archivname) und nicht in der Datenbank steht. Werden mehrere Importverzeichnisse oder mehrere XML-Importer gleichzeitig benötigen, dann muss die Datei config.xml nachträglich bearbeiten werden. Wenn nicht, dann kann man das nächste Kapitel überspringen.

2 Aufbau der config.xml

Auszug aus der Datei config.xml (Beispiel für einen XML-Importer):

```
<properties>
<comment>parameters for this web application</comment>
<entry key="import1_encryption_FIBUKey">55-219-106-48-149-511-142-
114</entry>
<entry key="import1_encryption_GFKey">89-106-23-789-111-654</entry>
<entry key="import1_ixurl">http://localhost:8080/ix-elo/ix</entry>
<entry key="import1_fileextension">ESW</entry>
<entry key="import1_directory1">C:\ELOenterprise\data\im-
elo\import1</entry>
<entry key="import1_directory2">C:\ELOenterprise\data\im-
elo\import2</entry>
<entry key="import1_movedirectory">C:\ELOenterprise\data\im-
elo\processed</entry>
<entry key="import1_username">Administrator</entry>
<entry key="import1_passwd">234-167-21-87-88-80-78-122</entry>
<entry key="import1_sigfile">false</entry>
<entry key="import1_ConfirmDir">C:\import_confirm_files</entry>
 <entry key="import1_ConfirmExt">.QIT</entry>
 <entry key="import1_ConfirmOk">Document guid:%GUID% imported</entry>
<entry key="import1_ConfirmErr">Document Import Error</entry>
<entry key="import1_ConfirmStructure">true</entry>
<entry key="import1_ConfirmEmpty">false</entry>
<entry key="import1_sleepTimeUser">120000</entry>
<entry key="import1_keywording_form_content">true</entry>
</properties>
```

Die Parameter in der Konfigurationsdatei setzen sich aus einem Parameternamen (key) und einem Parameterwert zusammen. Der Parametername besteht aus dem Importernamen und dem zugehörigen Parameter. Beide sind durch den ersten Unterstrich (_) getrennt. Zusammengesetzte Parameter werden ebenfalls durch Unterstriche getrennt, z. B.

```
"import1_encryption_FIBUKey":
```

Hier ist "*import1*" Name des Importers und "*encryption_FIBUKey*" der dazugehörige Parameter. Falls mehrere, unabhängig voneinander arbeitende XML-Importer gleichzeitig verwendet werden, kann man weiterer Namen z. B. import2, import3 wählen.

Eintrag in der config.xml	Erläuterung
`encryption_`***name***	Verschlüsselungskreis mit dem Passwort. In die Steuerungsdatei wird nur der Verschlüsselungskreis (**name**) eingetragen s.u.
`ixurl`	URL für den Indexserver-Zugriff. Wenn der Server xmlserver und das Archiv elo heißt: http://xmlserver:8080/ix-elo/ix.
`directory1`	Austauschverzeichnis für Dokument- und Steuerdateien.
`directory2`	2. Austauschverzeichnis für Dokument- und Steuerdatei (optional).
`directory`***X***	Beliebige weitere Austauschverzeichnisse.
`fileextension`	ESW oder XML (ohne Einstellung, XML Import).
`movedirectory`	Verzeichnis, in das die Importdateien nach dem Import verschoben werden sollen. Wenn die Dateien nach dem Import gelöscht werden sollen, wird im <entry>-Tag nichts eingetragen: <entry key="*import1_movedirectory*"></entry>.
`username`	ELO Anwendername (z.B. Administrator) für die Anmeldung. Wichtig: Dieser ELO Anwender muss die Rechte "Dokumente bearbeiten" und "Archive bearbeiten" besitzen.
`passwd`	verschlüsseltes (empfohlen) oder unverschlüsseltes Passwort für den ELO Benutzernamen. Sie können das ELOenterprise Password Utility verwenden, um ein neues verschlüsseltes Password zu erhalten.
`sigfile`	**true**, wenn eine Signaldatei (.sig) zum Starten des Importers notwendig ist. Der Name der Signaldatei muss gleich dem Namen der Importdatei sein.

`sleepTimeUser`	Leerlaufpause in ms zwischen Abarbeitung der Austauschverzeichnisse (60000 entspricht 1 Minute). Werte 100, 3000, 10000, 60000 und 12000 dürfen nicht gesetzt werden.
`keywording_form_content`	**true**, wenn Workflow, Verschlüsselungskreis, Marker (Farbe) und Indexaufbau aus der Ablagemaske übernehmen werden sollen. Funktioniert nur, wenn in der XML-Steuerungsdatei diesbezüglich keine Angaben gemacht werden.
`Confirm….`	Beschreibung siehe „Kapitel Quittungsdateien erzeugen“ weiter unten.

3 XML Dokumentenimport

3.1 Aufbau der XML Import Steuerdatei

Beim Import neuer Dokumente können in einer XML-Import Steuerdatei mehrere Dokumente eingetragen werden. Solange es keine zwingenden Gründe dafür gibt, sollte nur ein Dokument pro Datei vorgesehen werden. Diese Variante macht den Wiedereinstieg nach einer Fehlersituation wesentlich einfacher.

Die Wurzel des XML Baums besteht aus der eloobjlist. Diese kann eine beliebige Anzahl von Objekten <obj> enthalten. Jedes Objekt erzeugt ein ELO Dokument.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<eloobjlist ver="1.0">
<obj mode="..." dub="...">
<desc value="..."/>
<idate value="..."/>
<xdate value="..."/>
<deldate value="..."/>
<type value="..."/>
<marker value="..."/>
<sreg value="..."/>
<path value="..."/>
<dtype value="..."/>
<encryption>...</encryption>
<acl><access type="..." name="..."/></acl>
<memo>...</memo>
<indexlist><index name="..." value="…"/></indexlist>
<destlist>
<destination type="..." value="[...]+L1(1,4)+LA + LD + LK"/>
(LA, LD, LK und L1(1,4) siehe Anmerkung hinter der Tabelle
mit Erläuterungen zu den einzelnen Tags
</destlist>
<repl>...</repl>
<map name=”userdefined”><data key=”k1” value=”v1”/><data../></map>
<versioncomment>...</versioncomment>
<workflowlist><workflow template="..." name="..."/></workflowlist>
<structurelist><structure type="…" value="[...]"/></structurelist>
<docfile name="..."/>
<sigfile name="..."/>
<attfile name="..."/>
 </obj>
</eloobjlist>
```

3.2 Erläuterung zu den einzelnen Tags

Tag	ELO Feld	Erläuterung (Bsp. siehe Kap. 3.1)
`obj`		Das <obj>-Tag bildet einen Rahmen um ein Dokument oder Strukturelement. Mehrere Dokumente innerhalb einer Datei werden durch eine Abfolge von <obj> Gruppen dargestellt. **Attribut mode:** Falls Strukturelemente nur dann angelegt werden sollen, falls sie noch nicht vorhanden sind, dann enthält dieses Tag ein zusätzliches Attribut „mode" mit dem Wert „check". In diesem Fall wird zuerst das Ziel aus dem ersten Destination Eintrag geprüft. Wenn das Objekt an dieser Stelle bereits vorhanden ist, wird die Bearbeitung dieses Eintrags abgebrochen, es wird nicht noch ein zweites Mal eingetragen. **Attribut guid**: <obj guid="*(B66B68360A-87F4-4AC6-874A-9FE92DF9E6)*"> Das Objekt wird mit der im Attribut „guid" angegebenen GUID "*(B66B68360A-87F4-4AC6-874A-9FE92DF9E6)*" abgelegt. **Attribut license:** (nur in Verbindung mit guid)<obj guid="*(B66B68B60A-87F4-1AC3-874A-9FE92DF9E6)*" license="*B620BBB2F2B39BCAB6A599D218D5D8*"> Der XML Importer benötigt normalerweise eine Freischaltung über die Seriennummer damit er läuft. Es

		gibt aber eine Ausnahme: er kann auch ohne spezielle Lizenz betrieben werden, wenn die Quelle nur speziell gekennzeichnete XML Dateien liefert. In diesem Fall wird im <obj> Tag ein Attribut guid und ein Attribut license eingetragen. Attribut dup: <obj dup="From"> in Verbindung mit einer Indexzeile <index name="From" value="*XXXX*">. Zunächst wird nach einem Dokument im Archiv gesucht mit der obigen Verschlagwortung. Wenn das Dokument existiert, so wird es heruntergeladen und anstelle des Dokuments aus der XML-Datei importiert.
desc	Kurzbezeichnung	Das value-Attribut darf nicht leer sein.
idate xdate deldate	Ablagedatum (internes Datum) Dokumentendatum (externes Datum) Verfallsdatum	Falls das value-Attribut leer ist, wird das aktuelle Tagesdatum des Importzeitpunkts eingetragen, Dokumentendatum hingegen bleibt leer. Xdate mit Attribut <mode="current">: Das aktuelle Datum wird im Dokumentendatum eingetragen. Die Datumsfelder werden mit einem ISO Datum (YYYYMMTT) gefüllt, die Jahreszahl ist vierstellig anzugeben. Das idate Feld kann zusätzlich zum Datum auch eine Uhrzeit enthalten (YYYYMMTTHHMMSS). Beachten Sie bitte, dass die Sekunden eingetragen werden müssen obwohl sie nicht mit gespeichert werden (Man kann diese fest auf 00 einstellen).

`type`	Verschlagwortungs-maske	Das value-Attribut darf nicht leer sein, er enthält entweder die Nummer der Verschlagwortungsmaske oder den Namen.
`dtype`	Dokumententyp	Hier können Sie über die Werte 254...285 die unterschiedlichen Dokumententypen ansprechen. Auch hier können Sie zusätzlich Strukturelemente erzeugen. Diese dürfen die Werte 1...32 erhalten und keinen Dokumenteneintrag besitzen.
`access`	Schlüssel	Das name-Attribut enthält den Namen. Falls ein Name beim Import nicht zugeordnet werden kann, wird er ignoriert. Das value-Attribut kann ein r (lesender Zugriff), w (schreibender Zugriff), d (delete), e (edit document), l (Listen bearbeiten) oder eine Kombination aus diesen fünf Zeichen enthalten. UND Gruppe: Über das subgroup-Attribut können UND-Gruppen definiert werden, mehrere Gruppen werden durch ein Semikolon „;“ getrennt.
`acl`		Im <acl>-Tag befinden sich alle Schlüssel (<access>). Das <acl>-Tag kann zusätzlich das Attribut mode="new" enthalten. Das Attribut zeigt an, dass alle vorhandenen Dokumentenberechtigungen (Schlüssel) aus der Verschlagwortungsmaske gelöscht und nur die neuen aus der XML Steuerdatei eingetragen werden.
`memo`	Zusatztext	Falls der Zusatztext explizit Zeilenumbrüche enthält, müssen diese mit \n oder \r\n kodiert werden. Zeilenumbrüche der XML Quelle werden nicht übernommen.

`marker`	Farbmarkierung	Gibt die Farbmarkierung des logischen Dokumenteneintrags an. Dabei kann nur die Markernummer, nicht der Name eingetragen werden.
`sreg`	Versionsnummer	Stellt ein, welche aktuelle Versionsnummer im Verschlagwortungsdialog angezeigt wird, max. 8 Zeichen.
`map`	„Weitere Infos“ im Verschlagwortungs-dialog (bei vordefinierter Map-Domäne)	Im <map>-Tag befinden sich alle Name-Wert-Paare im Tag <data>, der die Attribute key und value enthält. Der <map>-Tag kann das Attribut name erhalten, das eine benutzerdefinierte Map-Domäne anlegt. (siehe Kap. 16 Maps (8.0) IX-Buch für mehr Details) Anmerkung: über das Attribut *name* erzeugte, benutzerdefinierte Maps, können im Client weder angezeigt, noch bearbeitet werden.
`destination`	Ablageort	Das value-Attribut enthält ein Ablageziel im „Lookup Index“ Format von ELO. Der erste Ablageort wird im ELO automatisch zum Haupteintrag, alle weiteren Orte werden Referenzen (siehe weitere Anmerkungen unter der Tabelle). Wenn ein Ablageort über die Form ¶Schrank¶Ordner… definiert wird, erzeugt der Importdienst bei Bedarf die benötigten Zwischenknoten. Optional kann der Ablagepfad auch in der Index-Notation des ELO Clients, zur automatischen Generierung aus den Indexdaten oder über die Dokument ID, angegeben werden. In diesem Fall muss der Ablageort vorhanden sein. Über das type-Attribut können Sie die Verschlagwortungsmaske für neu angelegte Strukturelemente festlegen.

		Falls dieser Parameter fehlt, wird die Verschlagwortungsmaske Ordner verwendet. <destination type="Rechnung" value="[…] + LA"/> Platzhalter LA, LK usw. siehe Beschreibung unten
`index`	Indexzeile	Das name-Attribut enthält entweder die Nummer der Indexzeile (1..51) oder die Gruppenbezeichnung der Indexzeile. Da letzteres nicht unbedingt eindeutig ist (da ein Gruppeneintrag mehrfach auftreten darf), muss in diesen Fällen die numerische Form gewählt werden. Lautet die Gruppenbezeichnung „ELO_FNAME“, so wird der Wert in der 51. Indexzeile (Zeile für den Dateinamen) eingetragen.
`structure`	Zusätzliche Register	Das value-Attribut enthält ein Ablageziel, das leer angelegt wird, falls es nicht existiert. Im Unterschied zum Destination-Tag wird hier aber keine Referenz auf das Dokument eingetragen, das Ziel bleibt leer. Über diesen structure-Tag können Sie sicherstellen, dass bei der Anlage eines Ordners gleich eine komplette Registerstruktur erzeugt wird (auch solche, die durch diesen Import gar nicht gefüllt werden, siehe weitere Anmerkungen unten). Diese Funktion sollten Sie aber sparsam einsetzen, da jeder structure-Pfad bei jedem Ablagevorgang auf Existenz überprüft werden muss. Je nach Registerstruktur kann das spürbare Performanceeinbrüche mit sich bringen. Über das type-Attribut können Sie die Verschlagwortungsmaske für neu

		angelegte Strukturelemente festlegen. Falls dieser Parameter fehlt, wird die Maske Ordner verwendet.
`versionscomment`	Kommentar	Versionskommentar
`docfile`	Dokumentdatei	Das name-Attribut enthält den Dateinamen des Objekts. Es kann leer bleiben, dann wird ein Dokument ohne Dokumentendatei erzeugt. Falls ein Name ohne Pfad angegeben wird, bezieht sich dieser auf das Verzeichnis, welches die XML Datei enthält. Jedes Dokument muss eine eigene Dokumentendatei enthalten. Falls mehrere Dokumente in einer Datei gepackt vorliegen, müssen diese in einem Extra-Durchlauf erst entpackt werden bevor die XML Datei an ELO übergeben wird. Der Dateiname wird im Versionskommentar der DocHistory Tabelle gespeichert.
`sigfile`	Signaturdatei	Über diesen optionalen Eintrag wird dem neu angelegten Dokument gleich eine Signatur mitgegeben. Die Signaturdatei muss durch die ELO Signaturkomponente erzeugt worden sein. Eine Signaturdatei kann nur angegeben werden, wenn eine Dokumentendatei vorhanden ist.
`attfile`	Anhang	Über diesen optionalen Eintrag wird dem neu angelegten Dokument eine Dateianbindung eingetragen. Ein Anhang kann nur angegeben werden, wenn eine Dokumentendatei vorhanden ist.
`path`	Ablagepfad	Nummer des Ablagepfades für dieses Dokument bei der Neuablage. Wenn dieses Feld fehlt, wird die Vorgabe aus der Verschlagwortungsmaskendef. verwendet. Wenn auch hier kein Wert eingetragen ist, wird der aktuelle

		Standardablagepfad gewählt. ACHTUNG: wird ein ungültiger Wert vorgegeben, dann kann das Dokument nicht abgelegt werden.
`repl`	Replikationskreise	Im Normalfall erbt ein XML Eintrag die Replikationskreisliste des Zielregisters. Falls ein Dokument in die Chaosablage gelegt werden soll oder für die Dokumente individuelle Replikationskreise eingerichtet werden sollen, dann kann man über diesen Tag die gewünschten Replikationskreise angeben. Die Liste enthält kommaseparierte 32 Bit Integerwerte welche einen Bitvektor für die gewünschten Kreise ergeben.
`encryption`	Verschlüsselungs-kreis	Name des Verschlüsselungskreises.
`workflow`		Über das template-Attribut wird ein bereits definierter Workflow festgelegt. Das name-Attribut: Name unter dem der Workflow gestartet wird.

Anmerkung:

Für die Ablagepfade (Tag <destination> und <structure>) können folgende Platzhalter verwendet werden:

LA: Das Ablagedatum wird in den Ablagepfad übernommen

LD: Das Dokumentendatum wird in den Ablagepfad übernommen

LK: Die Kurzbezeichnung wird in den Ablagepfad übernommen

L2(1,15): Der Inhalt der 2. Indexzeile, ab dem 1. Zeichen mit einer Länge von 15 Zeichen, wird in den Ablagepfad übernommen

Ist die XML Steuerungsdatei UTF-8 kodiert, dann wird in der ersten Zeile der XML Datei encoding="UTF-8" eingetragen. Sonst encoding="ISO-8859" wie oben angegeben.
Folgende Zeichen müssen wie folgt maskiert werden:

```
<       &lt;
>       &gt;
&       &amp; Beispiel: <destination value="[¶Müller&amp;Meier]"/>
"       &quot;
'       &apos;
```

3.3 Importvorgang starten

Der ELO XML-Importer überwacht per Konfiguration eines oder mehrere Verzeichnisse. Sobald ein Datenstrom entdeckt worden ist, wird die XML Datei eingelesen, die benötigten Ablagestrukturelemente erzeugt und die Dokumente abgelegt. Die Entscheidung, ob ein neuer Strom vorliegt, wird vom Server anhand der exklusiven Verwendbarkeit der XML Datei vorgenommen. Das Erzeugen des Datenstromes kann also folgendermaßen erfolgen:

1. Die Quelle erzeugt eine XML Datei
2. Für jedes Dokument wird die Dokumentendatei in das Zielverzeichnis kopiert und die XML Datei um den entsprechenden <obj> Eintrag ergänzt.
3. Die XML Datei wird geschlossen

Nachdem die Quelle die XML Datei geschlossen hat, kann ELO mit der Verarbeitung beginnen.

Alternativ hierzu kann die Quelle die XML Datei auch unter einer anderen Extension als xml (z. B. .$$$) aufbauen. Erst wenn der Vorgang komplett abgeschlossen ist und die Datei geschlossen wurde, wird diese Datei dann von der Quellapplikation in .xml umbenannt. Dieser Weg ist im Normalfall am sichersten, setzt aber voraus, dass die Quelle eine Umbenennung durchführen kann.

Für den Fall, dass die Quelle die XML Datei nicht dauerhaft geöffnet hält und eine Umbenennung nicht möglich ist, gibt es noch die Alternative, dass ELO den Abschluss der Aktion nicht anhand der unbeschränkten Verfügbarkeit sondern anhand einer weiteren Steuerdatei erkennt.

1. Die Quelle erzeugt die XML Datei xyz.xml
2. Die Dokumente werden geschrieben
3. Die XML Datei wird geschlossen
4. Die Signaldatei xyz.sig wird erzeugt (leer)

ELO beginnt mit der Verarbeitung, sobald die Signaldatei vorhanden ist.

Eine andere Möglichkeit den Importvorgang zu starten bietet die HTML-Statusseite im Webbrowser. Mit dem Link „Import now“ wird die Bearbeitung manuell gestartet. Dabei wird der Cache des dazugehörigen Servers erneuert, was sinnvoll ist, wenn zwischenzeitlich die Konfiguration z. B. über den ELO-Client geändert wurde (neue Verschlagwortungsmaske, Benutzer usw.).

3.4 Spaltenindex

ELO kennt die Möglichkeit zu einer Indexzeile mehr als nur einen Wert zu hinterlegen, im COLD Bereich „Spaltenindex" genannt. Diese Form ist dann sinnvoll, wenn zu einem Eintrag eine Reihe gleichwertiger Daten zur Verfügung steht (z. B. zu einem Dokument „Rechnung" eine Liste von Bestellnummern). Die Verwendung dieses Spaltenindex-Eintrags über die XML Steuerdatei ist besonders leicht. Wenn innerhalb der <indexlist> mehrere <index> Einträge mit gleichem name-Attribut vorhanden sind, werden diese automatisch zu einem Spaltenindex zusammengefasst.

Beispiel: Die Indexzeile 1 trägt die Bestellnummern zu der Rechnung, es gibt die Nummern 123, 140 und 210. Dann sieht der Ausschnitt aus der XML-Datei folgendermaßen aus:

```
<indexlist>
<index name="1" value="123"/>
<index name="1" value="140"/>
<index name="1" value="210"/>
</indexlist>
```

Die Indexeinträge können hierbei in beliebiger Reihenfolge kommen, sie müssen nicht sortiert vorliegen. Beachten Sie bitte, dass ein Eintrag der Form:

```
<index name="1" value="123,140,219"/>
```

ein völlig anderes Ergebnis erzeugt. Hier wird kein Spaltenindex erzeugt sondern einfach nur ein großer Einzeleintrag. Ein effizientes Suchen nach der Bestellnummer 140 ist dem SQL Server in dieser Form nicht möglich.

3.5 Struktur erzeugen

Die benötigte Ablagestruktur wird über die <destination> und <structure> Tags bei Bedarf automatische erzeugt. Hier haben Sie allerdings nur eingeschränkten Einfluss auf die Verschlagwortung und den Verschlagwortungsmaskentyp der erzeugten Elemente. Es gibt aber die Möglichkeit die Struktur explizit durch eigene <obj> Einträge zu erzeugen. Sie haben dann vollen Einfluss auf die Objektparameter und die Verschlagwortungsvergabe. Damit das Strukturelement nicht jedes Mal neu erzeugt wird und somit fälschlicherweise mehrfach vorhanden ist, muss <obj> Eintrag als *geprüft* markiert werden was über das mode-Attribut pass (<obj mode="check">).

Ein Beispiel für eine Strukturerstellung:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<eloobjlist ver="1.0">
 <obj mode="check">
       <desc value="Ordner99"/>
       <xdate value="20041011"/>
       <type value ="Entwicklungsprojekt"/>
       <dtype value="2"/>
       <indexlist>
             <index name="EWP_TYP" value="Testprojekt 2"/>
       </indexlist>
       <destlist>
             <destination type="Entwicklung" value="[¶XML Docs]"/>
       </destlist>
 </obj>
</eloobjlist >
```

In diesem Beispiel wird zuerst geprüft, ob in dem Strukturelement *XML Docs* ein Ordner mit dem Namen *Ordner99* vorhanden ist. Wenn ja, dann wird der <obj> Eintrag ignoriert. Andernfalls wird dieser Ordner nun mit den Informationen aus der XML Steuerdatei erzeugt wobei man jetzt einen sehr weit reichenden Einfluss auf die Basisparameter (Verschlagwortungsmaske, Datumseinträge usw.) und Verschlagwortung hat.

Falls es in diesem Beispiel auch das Strukturelement *XML Docs* noch nicht gibt, dann wird dieses ebenfalls automatisch angelegt.

3.6 Quittungsdateien erzeugen

Eintrag in der config.xml	Erläuterungen
`ConfirmDir`	Hier hinterlegen Sie das Verzeichnis, in welches Quittungsdateien geschrieben werden. Dieser Eintrag darf (wie alle Verzeichniseinträge in den XML-Profilen) kein \ am Ende besitzen, z. B. E:\ELOEnterprise\confirm.
`ConfirmExt`	Dateikennung für die Quittungsdateien (.QIT)
`ConfirmOk`	Text in der Quittungsdatei für „Dokumente korrekt abgelegt“ Sie können hier folgende Platzhalter verwenden: %ID%: gibt die Objekt-Id des neuen Eintrags aus. %GUID%: gibt die GUID des neuen Eintrags aus. %DESC%: gibt die Kurzbezeichnung des neuen Eintrags aus. %IX:nn%: gibt die Indexzeile nn (beginnend mit 0) aus.
`ConfirmErr`	Text in der Quittungsdatei für „Fehler bei der Ablage“.
`ConfirmStructure`	Bei *true* Quittungsdatei für Strukturelemente.
`ConfirmEmpty`	Bei *true* werden auch leere Quittungsdateien geschrieben.

Über die Einträge *ConfirmDir*, *ConfirmExt*, *ConfirmOk*, *ConfirmErr*, *ConfirmStructure* und *ConfirmEmpty* können Quittungsdateien zu den importierten Datensätzen erzeugt werden. Die Quittungsdatei erhält den gleichen Namen wie die XML Steuerdatei mit der unter *ConfirmExt* eingestellten Dateikennung (Erweiterung). Pro Dokumenteneintrag in der XML Steuerdatei findet man in der Quittungsdatei eine Zeile, welche entweder den eingestellten *ConfirmOk* oder *ConfirmErr* Text enthält.

AUSNAHME: Wird die Prüfung der Existenz der Dokumente eingestellt und deshalb kein neues Dokument bzw. Strukturelement angelegt, so wird nichts in die Quittungsdatei geschrieben. Das Erzeugen einer leeren Quittungsdatei kann per Option in der config.xml eingestellt werden. Wenn die XML-Struktur der Eingabedatei

unvollständig oder beschädigt ist, kann die Verarbeitung vor Erreichen des Dokumentenendes abgebrochen werden, in diesem Fall findet man in der Quittungsdatei weniger Zeilen als Blöcke in der XML Steuerdatei vor.

4 Automatischer Verschlagwortungsupdate

Der Bulk Importdienst kann auch für den automatischen Verschlagwortungsabgleich eingesetzt werden. Hier werden die Dokumente im ELO mit nur einer minimalen Verschlagwortung (z. B. einem Barcode) abgelegt. Aus einer Applikation wird dann eine XML Datei mit der vollständigen Verschlagwortung erzeugt. Über den Dienst wird nun automatisch die Verschlagwortung ergänzt. Das Format der XML Datei ist dabei im Wesentlichen identisch zur Importdatei.

```
<?xml version="1.0"  encoding="utf-8" ?>
<eloupdatelist ver="1.0">
  <obj>
    <source value="…"/>
    <desc value="…"/>
    <idate value="…"/>
    <xdate value="…"/>
    <acl>
      <access type="…" name="…" />
      …
    </acl>
    <memo>…</memo>
    <indexlist>
      <index name="…" value="…"/>
      …
    </indexlist>
    <destlist>
      <destination value="…"/>
      …
    </destlist>
    <workflowlist>
       <workflow template="..." name="..."/>
    </workflowlist>
    <docfile name="..."/>
  </obj>
  …
</eloupdatelist>
```

4.1 Unterschiede bei den einzelnen Tags

Tag	Name im Importdaten-satz	Erläuterung
`Eloupdate-list`	`eloobjlist`	Dieser Unterschied ermöglicht es dem Dienst zwischen den beiden Listenarten zu unterscheiden.
`obj`	`obj`	**Attribut mode:** Hier kann der Wert „versioning“ eingetragen werden. Dann wird das Dokument angelegt wenn es nicht vorhanden ist.
`source`	./. nicht vorhanden	Über dieses Feld wird der bestehende, mit minimalen Angaben verschlagwortete ELO Eintrag angesprochen. Das kann direkt die interne ELO ObjektId sein (#1234 -> ObjektId 1234) oder auch eine oder mehrere Indexzeile „BARCODE=1234567;REGNR=C2Z68 “ wenn über die Indexzeile BARCODE der Eintrag mit dem Code 1234567 und der Eintrag REGNR mit dem Code C2Z68 ausgewählt werden soll.
`desc`	`desc`	Falls das value-Attribut dieses Tags leer bleibt, wird automatisch die Kurz-bezeichnung aus dem vorhandenen Datensatz beibehalten.
`idate, xdate`	`idate, xdate`	Falls das value-Attribut dieser Tags leer bleibt, wird automatisch die

		Datumsinformation aus dem vorhandenen Datensatz beibehalten.
index	index	Das Indextag kann ein zusätzliches Attribut „mode“ enthalten. Wenn dieser auf den Wert „new“ gesetzt wird, dann überschreibt dieser Eintrag alle bereits vorhandenen Einträge dieser Indexzeile. Wenn dieser nicht gesetzt ist, werden neue Indexwerte zu bereits vorhandenen in Form eines Spaltenindex hinzugefügt.
destlist	destlist	Die <destlist> kann leer bleiben, in diesem Fall bleibt der Eintrag an der alten Stelle im Archiv. Falls die <destlist> nicht leer ist, wird das Dokument vom ursprünglichen Ablageort an den ersten Eintrag der Liste verschoben. An allen weiteren Listeneinträgen werden Referenzen hinterlegt. Der erste Eintrag kann auch einfach mit einem * gefüllt werden. In diesem Fall bleibt das Dokument an seinem originalen Ablageort und es werden die zusätzlichen Referenzen angelegt.
docfile	docfile	Falls in einem Update-Datensatz ein docfile angegeben ist, wird es als neue Version an das logische Dokument gehängt.
attfile	attfile	Falls in einem Update-Datensatz ein attfile angegeben ist, wird eine neue Version des Anhangs an das Dokument gehängt.
acl		Im <acl> befinden sich alle Schlüssel (<access>). Das <acl> Tag kann zusätzlich das Attribut mode=“new“

		enthalten. Das Attribut zeigt an, dass alle vorhanden Schlüssel gelöscht und nur die neuen eingetragen werden.
`dtype`	Dokumenten-typ	Hier können Sie den Dokumententyp, Werte 254…285, und den Typ der Strukturelemente, Werte 1…32, ändern. ACHTUNG: Hier findet keine gesonderte Überprüfung statt, ob Sie ein Dokument oder ein Strukturelement ändern. Achten Sie bitte deshalb besonders darauf richtige Werte in der Steuerungsdatei zu setzen.
`map`		<map> kann ein Attribut mode="new" enthalten. Dann wird die alte Map gelöscht, eine neue Map angelegt.

Alle anderen Tags werden wie im Importdatensatz verwendet.