ELO Indexserver

[Stand: 11.09.2019 | Programmversion: 12.00.000]

Dieses Dokument behandelt verschiedene Themen mit Bezug zum ELO Indexserver.

Inhalt

1 Neuerungen in ELO 12 8
1.1 Architektur 8
1.1.1 ELO Access Manager als Plug-in 8
1.1.2 ELO Document Manager als Plug-in 10
1.1.3 Proxy-Modus 13
1.2 ELOix-API: Deprecated Klassen und Funktionen 14
1.3 Entfernte Funktionen 15
1.4 Optimierungen 15
1.4.1 EditInfoC.mbAll, SordC.mbAll 15
1.4.2 Cache für Client-Skripte 15
1.5 P-Recht: ‚Berechtigungen ändern' 16
1.5.1 API 16
1.5.2 Lesen bestehender ACLs 16
1.5.3 Speichern der neuen ACLs 16
1.5.4 Export und Import 16
1.6 Relationen in der Verschlagwortung 16
1.6.1 Standard-Stichwortliste zu 'TYPE_RELATION' 17
1.7 Java-11-Support 18
1.7.1 Unvermeidbare Meldungen in Tomcat Logs 18
1.8 Sicherheitsoptimierung der ELO Web-Anwendungen (Proxy-Plug-in) 18
1.8.1 Session Hijacking 18
1.8.2 Einfügen von HTTP-Security-Headern 20
1.8.3 Verlagerung aller Dienste hinter den ELO Indexserver (Proxy-Modus) 21
1.9 Überarbeitung der Vertretungsregel 26

1.9.1 Wichtige Hinweise ... 26
1.9.2 Übernahme bestehender Vertretungsregeln ... 27
1.9.3 API-Funktionen ... 27
1.9.4 Wichtige Änderungen in der Berechtigungsübernahme ... 28
1.9.5 Report ... 29
1.10 WebDAV Erweiterung für SAP ILM ... 29
1.10.1 Wichtige Anmerkungen ... 29
1.11 Erweiterung des Verschlüsselungskonzepts ... 30
1.11.1 Verschlüsselungsalgorithmus AES ... 30
1.11.2 Verschlüsselung und Entschlüsselung durch Server ... 31
1.11.3 Erweiterte Schlüsselkreise ... 31
1.11.4 Dokumentenverschlüsselung ... 34
1.11.5 Verschlüsselte Volltextinformation ... 34
1.11.6 Verschlüsselte Vorschaudateien ... 35
1.11.7 ELO iSearch indexiert verschlüsselten Volltext ... 36
1.11.8 Neue Konstanten ... 37
1.12 Änderungen am Datenbankschema ... 37
1.12.1 Primary Keys ... 37
1.13 Änderungen im Textreader ... 38
1.13.1 Allgemeine Änderungen ... 38
1.13.2 OCR ... 40
1.13.3 Änderungen bzgl. Konverter ... 41
1.14 OCR ... 42
1.15 Lizenz ... 43
1.16 Upgrade Index ... 43
1.17 Neue Option in der Konfiguration der ELO iSearch ... 43
1.17.1 Das Problem nicht-lateinische Zeichen ... 43
1.17.2 Analyzer-Option zum Abschalten der Entfernung von Nicht-ASCII-Zeichen 44
1.17.3 Die Option auf der ELO iSearch-Konfigurationsseite ... 44
2 Neuerungen in ELO 11 ... 46
2.1 Änderung im Deployment ... 46
2.2 Änderungen in der Berechtigungsprüfung ... 46

2.2.1 Schlüsselberechtigungen obsolet ... 46
2.2.2 Archivschlüssel ... 46
2.2.3 Recht ‚Dokumentdateien sichtbar' entfernt ... 46
2.2.4 Löschen eines Untereintrags ... 47
2.2.5 Speichern von Attachments ... 47
2.2.6 Erstellen logischer Referenzen ... 47
2.2.7 Speichern neuer Dokumentversionen ... 47
2.2.8 Berechtigungen aktiver Workflows ... 47
2.3 Verschlagwortungsmasken ... 47
2.4 Workflow-Cache ... 48
2.4.1 Performance ... 48
2.4.2 Konfiguration ... 48
2.5 Skripting ... 48
2.5.1 Sealed-Objekte ... 48
2.6 ELO Indexserver-Plugins ... 49
2.7 LDAP-Anmeldung über ELOix ... 49
2.7.1 Funktionalität ... 49
2.7.2 Konfiguration ... 51
2.7.3 Skripting ... 53
2.8 ELO iSearch ... 54
2.8.1 Update der Elasticsearch ... 54
2.8.2 Verschlüsselte Kommunikation mittels SSL ... 54
2.8.3 Feed-Inhalte in der ELO iSearch ... 54
2.8.4 Auftrennung von Sord-Informationen und Volltextdaten ... 54
2.8.5 Aufteilung in unterschiedliche Indizes ... 55
2.8.6 Einführung einer neuen Such-API ... 56
2.8.7 Überarbeitung der Kommunikation bei mehreren ELOix Instanzen ... 56
2.8.8 Überarbeitung der Rechteüberprüfung (seit ELO 10.2) ... 56
2.8.9 Neue Config-Seite ... 56
2.8.10 Backups ... 57
2.9 WebDAV ... 57
2.9.1 Read-only ... 57

2.10 Cluster-Betrieb ... 58
2.11 Ebenen-Typ-Definitionen ... 59
2.12 Beispiel für Formatierungen ... 59
2.13 Änderungen im Textreader ... 59
2.13.1 Allgemeine Änderungen ... 59
2.13.2 OCR ... 60
2.13.3 Änderungen bzgl. Konverter ... 61
3 ELO Indexserver SSO ... 63
3.1 Voraussetzungen ... 63
3.2 NTLM für Indexserver konfigurieren ... 63
3.2.1 Voraussetzungen ... 63
3.2.2 Konfiguration ... 63
3.3 SPNEGO konfigurieren ... 64
3.3.1 Voraussetzungen ... 64
3.3.2 SPNEGO für Linux/MacOS konfigurieren ... 65
3.3.3 Eine Kerberos Konfigurationsdatei bereitstellen ... 67
3.3.4 Integrierte Windows-Authentifizierung auf Windows-PCs ... 67
3.4 Client Anwendungen schreiben, die SSO über NTLM oder SPNEGO nutzen ... 68
3.4.1 Microsoft .NET Plattform ... 68
3.4.2 Java 7.0 Plattform ... 68
4 Indexserver Client Event API ... 69
4.1 Einführungsbeispiel: Untereinträge eines Ordners überwachen ... 69
4.2 Konzeptioneller Aufbau ... 70
4.3 Weitere Informationen ... 73
4.3.1 Umgang mit Ereignissen ... 73
4.3.2 Ereignis-Listener ... 74
4.3.3 Ereignisbusse ... 75
4.3.4 Ereignis-Handler ... 75
4.4 Anwendungsbeispiele ... 76
4.4.1 Ordner in einer Anwendung mit Benutzeroberfläche überwachen ... 76
4.4.2 Client-Client-Kommunikation ... 80
4.5 Technische Realisierung ... 82

5 Indexserver JSON API ... 83
5.1 Einführungsbeispiel: Anzeige eines Dokuments nach Objekt-ID ... 83
5.2 Cross-Domain Anfragen ... 86
5.3 Synchrone und asynchrone Aufrufe ... 87
5.4 Datei Upload ... 92
5.5 Anmeldung ... 96
5.5.1 Anmeldung mit Benutzername und -kennwort ... 96
5.5.2 SSO-Anmeldung: IXConnFactory.createSSO ... 96
5.5.3 Übernahme eines bestehenden Tickets ... 96
5.6 Elementselektoren ... 96
5.7 Anwendungsentwicklung, Debugging ... 98
5.7.1 Entwicklung mit Eclipse ... 98
5.8 Reverse-Proxy mit dem Apache2 Webserver ... 99
5.9 Beispiele ... 101
6 ELOix Lastverteilung ... 102
6.1 Überblick ... 102
6.2 Zusätzlichen Indexserver installieren ... 102
6.2.1 Konfiguration ... 104
6.3 Apache2 als Lastverteilung ... 106
6.3.1 Apache2 installieren ... 106
6.3.2 Apache2 konfigurieren ... 107
6.3.3 Tomcat Konfiguration ergänzen ... 108
6.3.4 Indexserver Option ixUrlBase setzen ... 109
6.3.5 Weitere Informationen ... 109
7 Zugriff auf ein ELO Archiv über WebDAV ... 110
7.1 ELO WebDAV –Server-Adresse ... 110
7.1.1 Verbindung mit vollqualifiziertem Domänennamen ... 110
7.2 ELO WebDAV unter Windows verwenden ... 112
7.2.1 Allgemeiner Hinweis zur Verwendung unter Windows ... 112
7.2.2 Mit Windows Explorer als Netzlaufwerk verbinden (ab Windows Vista) ... 112
7.2.3 Mit Windows XP ... 115
7.2.4 Verwendung des WebDAV Redirectors ... 116

7.2.5 Mit der Kommandozeile ... 116
7.2.6 Mit NetDrive ... 117
7.2.7 Mit BitKinex ... 118
7.2.8 WebDAV unter Microsoft Office Anwendungen verwenden ... 119
7.2.8.1 Allgemeiner Hinweis zur Verwendung unter Microsoft Office Anwendungen 119
7.2.8.2 Objekte aus Archiv in MS Office Anwendungen einbetten ... 120
7.3 WebDAV unter Linux verwenden ... 121
7.3.1 Davfs2 (Kommandozeile) ... 121
7.3.2 Gnome/Nautilus ... 121
7.3.3 KDE/Dolphin ... 121
7.4 WebDAV unter MAC OS X verwenden ... 122
7.4.1 Kommandozeile ... 122
7.4.2 Finder ... 122
7.4.3 Cyberduck ... 122
7.5 WebDAV unter Android verwenden ... 123
7.5.1 WebDAV File Manager ... 123
7.5.2 WebDAV Nav Lite ... 124
7.6 Webdav unter iOS verwenden ... 124
7.6.1 WebDAV Navigator ... 124
7.6.2 Over The Air ... 124
7.7 Übersicht: WebDAV-Clients ... 125
8 Dynamische Stichwortlisten ... 127
8.1 Beispiel ... 127
8.2 Administration ... 127
8.3 Client-Anwendung ... 127
8.4 Beispiel-Skript ... 128
8.4.1 SampleScript.js ... 128
8.5 Konfiguration in der Verschlagwortungsmaske ... 131
8.6 Dynamische Stichwortlisten im ELOwf ... 131
8.6.1 Verwendung aus Maskendefinition wie in anderen Clients ... 131
8.6.2 Verwendung von dynamischen Stichwortlisten für Map-Felder ... 132

9 OCR über den Indexserver ... 134
9.1 Konfiguration ... 134
9.2 Aufruf der processOcr Funktion ... 135
9.2.1 Feststellen, ob OCR möglich ist ... 135
9.3 Landessprachen ausgeben ... 135
9.4 OCR-Analyse synchron durchführen ... 136
9.5 OCR-Analyse asynchron ausführen ... 136
9.6 OCR-Analyse mit Rückgabe der Zeichenpositionen ... 138
9.7 OCR-Analyse eines Rechtecks ... 140
10 Preview-Generierung ... 142
10.1 Technischer Hintergrund ... 142
10.2 Preview erzeugen ... 143
10.3 Administration ... 144
10.3.1 Kontrolle des Externen Prozesses ... 144
10.3.2 Cache ... 145
11 Skriptentwicklung ... 146
11.1 IXConnection Klasse in Ereignisskripten ... 146
12 Feed-Einträge erstellen ... 149

1 Neuerungen in ELO 12

Folgende Neuerungen wurden mit ELO 12 eingeführt.

1.1 Architektur

Dieses Kapitel beschreib die Veränderungen an der Architektur des ELO Servers.

1.1.1 ELO Access Manager als Plug-in

Der ELO Access-Manager wird als Plug-in des ELO Indexservers betrieben und über die Datei *ud-config.xml* (im Konfigurationsverzeichnis des ELO Indexservers) parametrisiert. Es entfällt somit ein separates Konfigurationsverzeichnis.

1.1.1.1 Konfigurationsdatei ‚ud-config.xml'

Mit den Einstellungen in der Datei *ud-config.xml* öffnet das AM-Plug-in die Datenbank, in der die Tabelle *amoptions* enthalten ist. Die Konfigurationsdatei entspricht der Datei *config.xml* der Web-Anwendung *ELO Access Manager* aus den vorigen Versionen. Das Präfix „ud-" der Konfigurationsdatei steht für „User Directory".

Beispiel für eine ud-config.xml:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
<entry key="jdbcurl">jdbc:sqlserver://PCIMIGW-ELO12:1433</entry>
<entry key="dbdriver">com.microsoft.sqlserver.jdbc.SQLServerDriver</entry>
<entry key="dbpwd">137-21-224-42-…</entry>
<entry key="schema"/>
<entry key="database">eloam</entry>
<entry key="dbuser">elodb</entry>
</properties>
```

Der Parameter *instancename* ist nicht mehr erforderlich. Es wird hierfür der Instanzname des ELO Indexservers angenommen.

1.1.1.2 Tabelle ‚amoptions'

In der Tabelle *amoptions* sind die Einstellungen zu LDAP, Kerberos, ELO Document Manager und ELO Windows Client nicht mehr relevant. Die folgende Tabelle listet die in der Version 12 ausgewerteten Zeilen auf.

Instancename	Optid	Optval (Beispiel)	Remark
_ALL	1	/ELO12/data/eloam	Access-Manager-Datenverzeichnis
_ALL	2	…	Lizenzdateiinhalt in Spalte optblob
_ALL	3	1	Verwendungsart: 0: Produktivsystem 1: Testsystem 2: Entwicklungssystem
_ALL	84	false	Administrationsmodus eingeschaltet
_ALL	1001	600	Session-Lebenszeit (Ticket-Lebenszeit) in Sekunden
_ALL	1002	0	Maximale Anzahl der Login-Versuche (0: unbegrenzt)
_ALL	1004	250000	Maximale Anzahl von Benutzern und Gruppen (Standardwert 10000)
<name>	3000	elo12	Archivname
<name>	3001	SERVER12	Server-Name, s. IX-Einstellung *publicUrlBase*
<name>	3002	9090	Server-Port, s. IX-Einstellung *publicUrlBase*
<name>	3003	/ix-elo12/ix	Web-Anwendungspfad, s. IX-Einstellung *publicUrlBase*

Instancename	Optid	Optval (Beispiel)	Remark
<name>	3005…		Weitere Archive und Indexserver-URLs zwischen 3000 + 5n <= optid < 3003 + 5n.
_ALL	4005	20180801115006	Datum der Lizenzübermittlung (temporärer Wert, keine Einstellung)

1.1.1.3 LDAP-Anbindung

Die LDAP-Klassen des ELO Access Managers (ADUser, usw.) werden nicht mehr unterstützt. Die Anbindung an ein LDAP-System erfolgt mittels LDAP-Plug-in, das in der Version 11 eingeführt wurde.

1.1.1.4 Kerberos-Login

Die Login-Methode *loginKerberos* wird nicht mehr angeboten, weil sie bereits seit einigen ELO Versionen durch SSO ersetzt ist.

Die Funktion *IXConnFactory.createKrb* ist dadurch sinnlos und ist nicht mehr in der *IXConnFactory* enthalten.

1.1.1.5 HTTP-Schnittstelle

Aus Kompatibilitätsgründen kann der ELOam zumindest für ELO 12 weiterhin über HTTP angesteuert werden. Die URL ist nach folgendem Schema aufgebaut:

```
http://<Servername>:<Port>/ix-<Archivname>/plugin/de.elo.am.plugin
```

1.1.2 ELO Document Manager als Plug-in

Der ELO Document-Manager ist in ELO12 keine eigenständige Web-Anwendung sondern – wie der ELO Access-Manager – ein Plug-in des ELO Indexservers.

Alle Funktionsaufrufe zum ELOdm sind mit einer Signatur versehen, sodass auf die Übergabe eines Session-Tickets verzichtet werden kann. Dadurch benötigt der ELOdm keine Verbindung zum ELOam

Die Verbindungsparameter zur Datenbank werden dem ELOdm Plug-in vom ELO Indexserver übergeben. Eine eigene Konfigurationsdatei entfällt damit.

1.1.2.1 Tabelle ‚elodmopt'

In der Konfigurationstabelle *elodmopt* des ELOdm entfallen die Einstellungen zum ELOam. Es sind die Zeilen mit optno 11, 101 und 102.

Ferner ist die Option 144 (Private URL) überflüssig. Die private URL des ELOdm wird aus der privaten URL des ELO Indexservers abgeleitet. Beispiel:

http://server:9090/ix-elo120/plugin/de.elo.dm.plugin

Die Centera-Anbindung wird nicht weitergepflegt (siehe *1.1.3.2 Centera nicht mehr unterstützt)*. Die zugehörigen Optionen sind obsolet: 105 (USE_CENTERA), 12 (CENTERA_RETENTION).

Die folgende Tabelle listet die in er Version 12 ausgewerteten Zeilen der Tabelle *elodmopt* auf.

Instancename	Optno	Optval (Beispiel)	Remark
_ALL	1 … 4	Pfad-ID oder 0	Standard-Dokumentenpfade
_ALL	7	0	Restore-Pfad
_ALL	8	0	Verteilung der Dateien über die Standardpfade
_ALL	9	0	Aktuell aktiver Standardpfad aus optno 1…4. Entspricht optno-1.
_ALL	10	3	Archivreport-Modus, s. ELO Administration Console, Reportoptionen
_ALL	100	…….	Kodierte Archivreport-Optionen, s. ELO Administration Console, Reportoptionen
_ALL	13	0	Backup-Modus: 0: kein Backup 1: Backup 2: Backup mit Purge

Instancename	Optno	Optval (Beispiel)	Remark
_ALL	1 … 4	Pfad-ID oder 0	Standard-Dokumentenpfade
_ALL	7	0	Restore-Pfad
_ALL	8	0	Verteilung der Dateien über die Standardpfade
_ALL	9	0	Aktuell aktiver Standardpfad aus optno 1…4. Entspricht optno-1.
_ALL	10	3	Archivreport-Modus, s. ELO Administration Console, Reportoptionen
_ALL	14…17, 120		Purge-Optionen (Altdokumente entfernen), s. ELO Administration Console, „Backup-Tasks“
_ALL	18	3	Version der DB-Tabellen des ELO Document Managers
_ALL	109	false	TSM verwenden
_ALL	19, 110…119		TSM-Einstellungen
_ALL	121	false	Option für das Verschieben von Dateien. false: Datei verschieben (bewegen) true: Datei kopieren und löschen
_ALL	122	false	Schwärzungen erlauben
_ALL	123	false	ACTIVATOR verwenden

Instancename	Optno	Optval (Beispiel)	Remark
_ALL	1 … 4	Pfad-ID oder 0	Standard-Dokumentenpfade
_ALL	7	0	Restore-Pfad
_ALL	8	0	Verteilung der Dateien über die Standardpfade
_ALL	9	0	Aktuell aktiver Standardpfad aus optno 1…4. Entspricht optno-1.
_ALL	10	3	Archivreport-Modus, s. ELO Administration Console, Reportoptionen
_ALL	124…128		ACTIVATOR Einstellungen
_ALL	151		Reserviert
_ALL	169	false	Proxy-Modus verwenden
_ALL	170, 180, 190, 200		Proxy-Slot 1…4

1.1.3 Proxy-Modus

Im Proxy-Betrieb benötigt der ELodm die Einstellung *Proxy-Slot*. Sie wird in der Datei *config.xml* des ELOix eingetragen. Beispiel:

```
<entry key="proxyslot">1</entry>
```

1.1.3.1 HTTP-Schnittstelle

Aus Kompatibilitätsgründen kann der ELOdm zumindest für ELO 12 weiterhin über HTTP angesteuert werden. Die URL ist nach folgendem Schema aufgebaut:

```
http://<Servername>:<Port>/ix-<Archivname>/plugin/de.elo.dm.plugin
```

1.1.3.2 Centera nicht mehr unterstützt

Die Centera wird von EMC nicht mehr angeboten und nicht mehr supportet. Deshalb wird die Ansteuerung nicht mehr weitergepflegt.

1.2 ELOix-API: Deprecated Klassen und Funktionen

Die hier aufgelisteten API-Funktionen sind ab ELO 12 als „deprecated“ markiert und sollten nicht mehr verwendet werden.

„Deprecated“	Ersetzt durch	Hinweis
FindDirect	FindByESearch	Seit ELO 11 gibt es neue API-Klassen zur Suche in der ELO iSearch. Dadurch ist die Suchanfrage des Clients unabhängig von der zugrundeliegenden Suchtechnologie. Der Suchterm wird mittels *ESearchParams.query* übergeben, Suchfilter (z. B. *Maske*) mittels *ESearchParams.queryOperator*. Der Suchbereich (z. B. Indexfelder, Volltext) wird in *ESearchParams.searchIn* übergeben.
FindByFulltext	FindByESearch	Siehe *FindDirect*
IXClient/IXClientImpl	IXConnection	Diese Klassen sind nur noch eingeschränkt verwendbar. Einige Datenelemente werden über diese Klassen u. U. nicht mehr gelesen (z. B. *Sord.refPaths*).

1.3 Entfernte Funktionen

Die folgenden Funktionen sind bereits seit einigen ELOix-Versionen als „deprecated“ markiert und nun nicht mehr verfügbar:

- **collectWorkFlows**: deprecated seit ELOix Version 6. Für die Suche nach Workflows steht die Funktion *findFirstWorkflows* zur Verfügung.
- **collectWorkFlowNodes**: deprecated seit ELOix Version 6. Verwenden Sie *findFirstTasks*, um Workflow-Aufgaben zu finden.
- **checkinSubs(), checkoutSubs(), deleteSubs():** Seit der Überarbeitung der Vertretungsregel (siehe Kapitel *1.9 Überarbeitung der Vertretungsregel*) nicht mehr funktional. Die neuen API-Funktionen zum Managen von Vertretungen werden in Kapitel 1.9 erläutert.
- **loginKerberos und IXConnFactory.createKrb:** Werden nicht mehr angeboten (siehe *1.1.1.4 Kerberos-Logins*).

1.4 Optimierungen

1.4.1 EditInfoC.mbAll, SordC.mbAll

Wie die Erfahrung zeigt, wird in vielen Programmen mit dem Elementseketor mbAll gearbeitet, obwohl nicht alle Datenelemente benötigt werden. Weil mbAll viele und z. T. sehr aufwendige Datenbank-Statements generiert, wird also häufig Server-Leistung unnötig vergeudet.

Um dem entgegenzusteuern, werden nun Datenelemente, die selten benötigt aber aufwendig bereitzustellen sind, erst beim Aufruf der zugehörigen getter-Funktion geladen.

Für die folgenden Datenelemente wird das sogenannte „lazy-loading“ praktiziert:

```
Sord.refPaths, EditInfo.sordTypes, EditInfo.mask, EditInfo.keywords.
```



Information: Diese Optimierung gilt nur für Java-Programme, die mit der IXConnection-Klasse aus der Datei *EloixClient.jar* arbeiten. In der Client-Bibliothek für C# und JavaScript werden die Datenelemente weiterhin direkt geladen.

1.4.2 Cache für Client-Skripte

Beim Start des ELO Clients werden viele Skripte aus dem Archiv unter dem Ordner *Administration* geladen. Damit sie nicht für jeden Client-Start wieder vom Speichermedium gelesen werden müssen, gibt es nun einen Cache, der sie im RAM hält.

In diesem Cache werden standardmäßig alle Dateien gehalten, für die gilt:

- Verschlagwortet mit *ELO Scripts*
- Dateiendung *js*, *json*, *properties* oder *xml*
- Kleiner als 100KB
- Nicht verschlüsselt

1.5 P-Recht: ‚Berechtigungen ändern'

Mit ELO 12 gibt es ein weiteres Recht, über das bestimmt wird, ob ein Nutzer bzw. eine Gruppe die Berechtigungen eines Objektes ändern darf. Dieses Recht wird bei allen Objekten, die eine ACL besitzen, wirksam.

1.5.1 API

Das Recht ist Teil der Klasse *AccessC*: LUR_PERMISSION = 32.

Durch die Einführung ändert sich auch der Bitwert der Rechts LUR_ALL = 63.

1.5.2 Lesen bestehender ACLs

Wird eine bisher bestehende ACL ohne das P-Recht gelesen, wird das P-Recht automatisch hinzugefügt, sofern das Recht *Schreiben* (LUR_WRITE) besteht.

1.5.3 Speichern der neuen ACLs

Um neue und alte ACLs in der Datenbank zu unterscheiden werden neue ACL-Strings mit dem Prefix „1-" abgespeichert. Die Encodierung der ACLs wurde erweitert, um das neue Recht speichern zu können. Um Konflikte zu vermeiden, sollte immer mit den AclItems der Objekte und nie mit dem ACL String direkt gearbeitet werden.

1.5.4 Export und Import

Um zu gewährleisten, dass Dokumente, die mit ELO 12 (inkl. des neuen P-Rechts) auch in einer alten ELO Version importiert werden können, wurde die bestehende Struktur der Export-Dateien so erweitert, dass sowohl die alte als auch die neue Version der ACLs geschrieben werden.

Beim Import in ELO 12 oder höher wird, wenn vorhanden, der neue ACL-String ausgelesen und ansonsten der alte.

1.6 Relationen in der Verschlagwortung

Es steht ein neuer Indexfeldtyp zur Verfügung, mit dem ein Indexwert auf ein SORD-Objekt verweisen kann: *DocMaskLineC.TYPE_RELATION*.

Beispielsweise kann damit in einem Rechnungsdokument auf ein SORD verwiesen werden, das in der Verschlagwortung die Kontaktdaten des Lieferanten speichert.

Die referenzierten SORD-Objekte (Lieferanten) müssen bestimmten Verschlagwortungsmaksen zugeordnet sein. Welche Verschlagwortungsmasken erlaubt sind, ist in der Indexfelddefinition im Element *DocMaskLine.allowedMaskIdsForRelation* anzugeben.

Die dort angegebenen Verschlagwortungsmasken müssen wiederum mit der Option *DocMask.details.keywordingRelationAllowed* gekennzeichnet sein.

Im SORD-Objekt (Rechnungsdokument) wird eine Referenz dargestellt durch die GUID des referenzierten Objekts: *Sord.objKeys[.].data[.] = <guid>.*

Um in der Benutzeroberfläche eine verständliche Darstellung für die Referenz anbieten zu können, findet man in *Sord.objKeys[.].displayData[.]* die Kurzbezeichnung des referenzierten Objekts. Dieses Element ist readonly und sollte nicht im Skripting-Kontext verwendet werden.

Beispiel:

```
// Read suppliers Sord object
Sord supplier = conn.ix().checkoutSord("ARCPATH:/suppliers/ELO",
                                       SordC.mbMin, LockC.NO);

// Create invoice document
EditInfo invoiceInfo = conn.ix().createDoc("ARCPATH:/invoices", "Invoice",
                                           "", EditInfoC.mbSordDocAtt);
Sord invoice = invoiceInfo.getSord();
invoice.setName("Invoice 12345");

// Assing reference to supplier (assume index line 0 is of TYPE_REFERENCE)
invoice.getObjKeys()[0].setData(new String[] {supplier.getGuid()});

// ...
conn.ix().checkinDocEnd(invoice, SordC.mbAll, document, LockC.NO);

// Read invoice keywording
Sord invoiceR = conn.ix().checkoutSord(invoice.getGuid(),
                                        SordC.mbAllIndex, LockC.NO);
// Obtain supplier guid and name
String supplierGuid = invoiceR.getObjKeys()[0].getData()[0];
String supplierName = invoiceR.getObjKeys()[0].getDisplayData()[0];
```

1.6.1 Standard-Stichwortliste zu 'TYPE_RELATION'

Der ELO Indexserver bietet für ein Indexfeld mit der Eigenschaft *TYPE_RELATION* standardmäßig eine Stichwortliste an, wenn *DocMaskLine.scriptName* leer ist. Ihre Spalten enthalten die GUIDs, Kurzbezeichnungen und wichtigen Indexfelder der passenden Objekte.

1.7 Java-11-Support

Aufgrund der geänderten Support-Strategie von ORACLE wurde beschlossen, mit der Open-Source-Distribution von Java – dem OpenJDK – zu arbeiten. Weil das OpenJDK nur in seiner aktuellsten Version gepflegt wird, ist es nötig, die Java-Programme des ELO-Systems mit der Version 11 des OpenJDK zu veröffentlichen.

1.7.1 Unvermeidbare Meldungen in Tomcat Logs

Unter Java 11 werden Zugriffe auf interne Strukturen als Warnungen in die Log-Dateien des Apache Tomcat ausgegeben. Diese Zugriffe werden von Bibliotheken anderer Softwarehersteller ausgeführt und liegen außerhalb unserer Einflussmöglichkeiten.

Beispielsweise verursacht das OGSi-Framework Apache Felix diese Warnungen:

```
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.felix.framework.ext…
WARNING: Please consider reporting this to the maintainers …
WARNING: Use --illegal-access=warn to enable warnings of further …
WARNING: All illegal access operations will be denied in a future release
```

Diese Warnungen führen nicht zu einer Fehlfunktion und können ignoriert werden.

1.8 Sicherheitsoptimierung der ELO Web-Anwendungen (Proxy-Plug-in)

Durch die Verteilung der ELO Funktionalität auf mehrere Anwendungen gibt es das Problem, das verschiedene Dienste die gleiche Benutzer-Session benötigen.

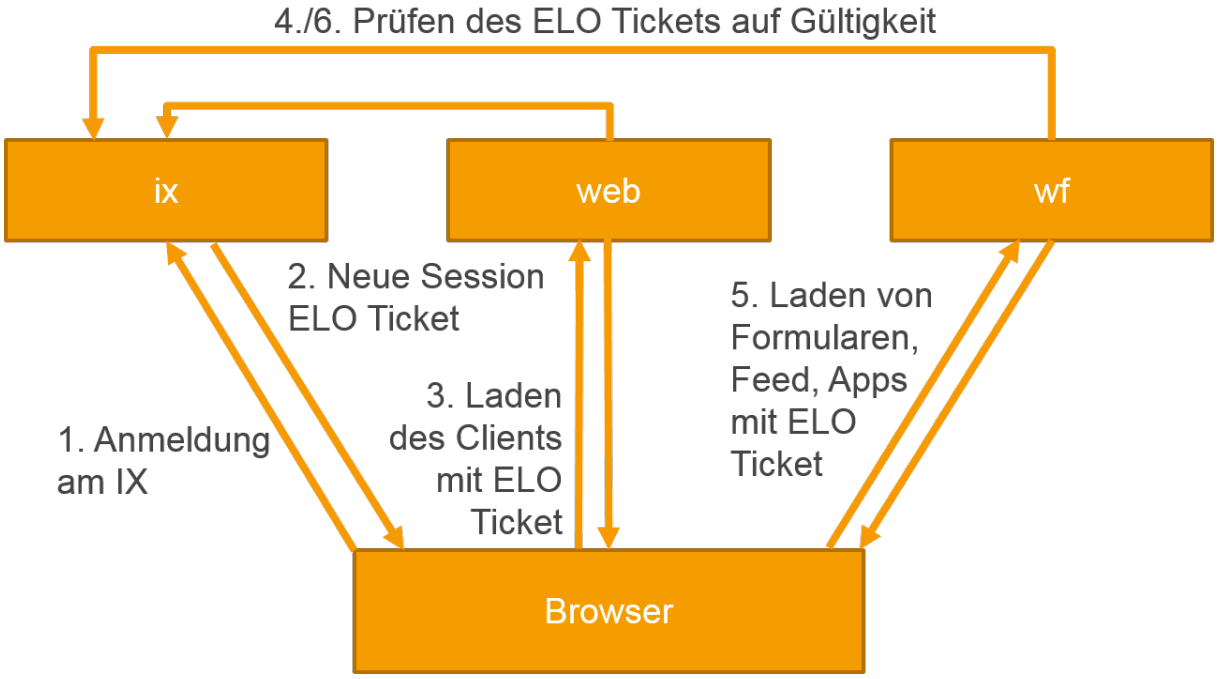
Aktuell arbeitet ein ELO Web Client-Benutzer nicht nur mit dem ELO Web Client Server (web), sondern auch direkt mit dem ELO Indexserver (ELOix) und dem ELO Forms Services (ELOwf). Das Problem dabei besteht darin, die Benutzer-Sessions von allen Anwendungen zu synchronisieren, ohne jeweils eine erneute Anmeldung einzufordern.

Dazu wird das ELO Ticket verwendet. Das ELO Ticket repräsentiert eine Benutzer-Session. Solange diese noch gültig ist, kann man sich mit dem ELO Ticket an allen Diensten anmelden und an dieser Benutzer-Session teilnehmen.

1.8.1 Session Hijacking

Alle Web-Anwendungen sind anfällig für Session Hijacking. Sollte ein Angreifer ein gültiges Session Ticket herausfinden, kann er dieses nutzen, um dadurch an der Session teilzunehmen, obwohl ihm die Anmeldedaten nicht bekannt sind.

Genau dieses Vorgehen wird durch die Verteilung der ELO Dienste ausgenutzt, um eine gültige Session zwischen den Diensten (ELOweb, ELOix und ELOwf) zu verteilen.



Hier wird aufgezeigt, wie das ELO Ticket nach der Anmeldung am ELOix an die anderen beiden Servermodule (ELOweb und ELOwf) verteilt und somit dem Benutzer Zugriff zur gesamten Funktionalität gewährleistet wird.

Ein Angreifer, der Zugriff auf das ELO Ticket erhält, könnte diesen Umstand ausnutzen und ebenfalls mit dem Ticket die Funktionalität nutzen.

Mögliche Angriffsszenarien, um an das Ticket zu gelangen sind:

- Cross-Site-Scripting
- Abhören des Datenverkehrs
- Auslesen des Tickets aus Log-Dateien

1.8.1.1 Gegenmaßnahmen

Log-Dateien vor unbefugtem Zugriff zu schützen und Netzwerkverkehr über SSL/TLS abzusichern sollte mittlerweile eine Selbstverständlichkeit sein. Etwas komplizierter wird es das ELO Ticket wirkungsvoll im Browser selbst (vor allem gegen Cross-Site-Scripting) zu schützen.

Generell behandelt ELO jede gemeldete Cross-Site-Scripting Lücke als Bug und behebt diese umgehend. Durch die Menge des Quellcodes, die hohe Erweiterbarkeit und das Einbinden von Drittsystem, kann ein absoluter Schutz wahrscheinlich nie gewährleistet werden.

Im Folgenden werden einiger Verfahren vorgestellt, wie die Web-Anwendungen abgesichert werden können:

- Einfügen von HTTP-Security-Headern
- Verlagerung aller Dienste hinter den ELO Indexserver (Proxy-Modus)

1.8.2 Einfügen von HTTP-Security-Headern

Der ELO Web Client und die ELO Apps liefern zusätzliche http-Header aus, um die Sicherheit zu erhöhen.

1.8.2.1 Strict-Transport-Security

Der Strict-Transport-Security-Header wird verwendet, um bei aktiver HTTPS-Verbindung dem Browser mitzuteilen, die Seite in Zukunft immer über HTTPS zu laden.

1.8.2.2 X-XSS-Protection

Dieser Header steuert den Browser-Schutz gegen Cross-Site-Scripting.

1.8.2.3 X-Content-Type-Options

Mit diesem Header wird dem Browser signalisiert, dass der MIME-Type in jedem Fall respektiert werden soll. JavaScript-Code in Bildern wird damit ignoriert.

1.8.2.4 Content-Security-Policy

Mit diesem Header signalisiert ein Server dem Browser welche Daten (Content) dieser für die Seite nachladen darf. Somit lassen sich Daten von anderen Servern und unsichere Code-Ausführung blockieren.

Im ELO Web Client kommt es wegen der Verteilung der Serverdienste, Anpassungen durch Drittanwendungen und ELO Apps oft zu Problemen, sodass eine genaue Einschränkung ohne Konfiguration nicht gemacht werden kann.

1.8.2.4.1 Konfigurieren des Content-Security-Headers

Auf der Apache-Tomcat-Einstellungsseite des ELO Web Clients kann der Wert des Content-Security-Headers konfiguriert werden.

Der Standard ist: `script-src 'self' 'unsafe-inline' 'unsafe-eval' <Adresse des ELOix aus der config.xml> <Adresse des ELOwf aus den Einstellungen>;`

Diese Zeile beschränkt das Nachladen von JavaScript-Dateien auf den eigenen Server, auf dem die Anwendung läuft und den Server, auf dem der ELO Indexserver läuft (falls dieser abweicht und CORS eingesetzt wird). Letzterer ist notwendig, um Skripte aus dem Ordner *Webclient Scripting Base* laden zu können.

Da der ELO Web Client Skripte dynamisch lädt und das eingesetzte ExtJS-Framework *eval* verwendet, sind die Einstellung `'unsafe-inline'` und `'unsafe-eval'` nötig.

1.8.2.5 Security Header konfigurieren

Über die Apache-Tomcat-Einstellungsseite des ELO Web Clients kann über die Einstellung *disableSecurityHeaders* die Header abgeschaltet werden. Dies sollte nur gemacht werden, falls Probleme mit dem Header auftreten. Über die Einstellung *contentSecurityPolicy* kann ein individueller Wert angegeben werden. Hier gilt zu beachten, dass zu restriktive Header zu Fehlern im ELO Web Client führen können.

Die gleichen Header werden in den ELO Apps verwendet und können dort im Zusatztext des Ordners *Configuration* (Pfad im Archiv: *Administration // ELO Apps*) über `,disableSecurityHeaders=true` abgeschaltet werden. Auch hier kann über `,contentSecurityPolicy‘` ein individueller Wert für den Content-Security-Header angegeben werden.

1.8.3 Verlagerung aller Dienste hinter den ELO Indexserver (Proxy-Modus)

Um eine Web-Anwendung gegen Session-Hijacking zu schützen wird empfohlen das Session-Ticket weiter abzusichern. Üblicherweise wird ein Session Ticket als Cookie gespeichert. Über spezielle Cookie Flags lassen sich Cookies weiter schützen:

- Secure: Das Cookie wird nur bei aktiver HTTPS-Verbindung übertragen.
- HttpOnly: Das Cookie kann über das JavaScript-Objekt *document.cookie* nicht ausgelesen werden. Damit wird es für einen Angreifer schwer, selbst bei gelungenem Cross-Site-Scripting-Angriff an das Cookie zu gelangen.

Der ELO Web Client hält das ELO Ticket ebenfalls als Cookie vor. Über die Apache-Tomcat-Einstellungsseite des ELO Web Clients kann dieses auf *Secure* gesetzt werden. (Dazu muss natürlich eine HTTPS-Verbindung eingerichtet sein.)

Die Einstellung *HttpOnly* kann jedoch nicht gesetzt werden, da das Ticket verwendet wird, um die URLs für Formulare, Feed-Seiten und ELO Apps zusammenzustellen, damit dort dieselbe Session verwendet werden kann (siehe Schaubild oben).

Zudem liefert der ELO Indexserver das ELO Ticket als Teil des ClientInfo-Objektes bei erfolgreicher Verbindung mit, sodass per JavaScript das Ticket auch dort abgegriffen werden kann.

1.8.3.1 Funktion createFromCookie der JSON-API

Der Indexserver ermöglicht eine neue Art der Anmeldung ab ELO11 (ELOix-Version 11.02). Dabei wird, statt des richtigen ELO Tickets, nach erfolgreicher Anmeldung eine Konstante gesetzt. Das eigentliche Session Ticket verbirgt sich in der *JSESSIONID*, einem Cookie des Apache Tomcats, dieses ist mit *HttpOnly* vor dem Zugriff aus JavaScript geschützt.

Wird die API-Funktion *createFromTicket* verwendet und der Wert des aktuellen Tickets aus der *ClientInfo* (die Konstante) eingesetzt, wird intern ein *createFromCookie* ausgelöst.

Das *createFromCookie* lässt den ELO Indexserver die *JSESSIONID* prüfen und falls es dazu bereits eine gültige laufende Session gibt wird der Zugriff zugelassen. Damit *createFromCookie* funktioniert muss natürlich die *JSESSIONID* übertragen werden, was nur aus dem gleichen Browser heraus (da hier das Cookie gespeichert wurde) und an den gleichen Server (an andere Server hängt der Browser das Cookie nicht an) gemacht werden kann.

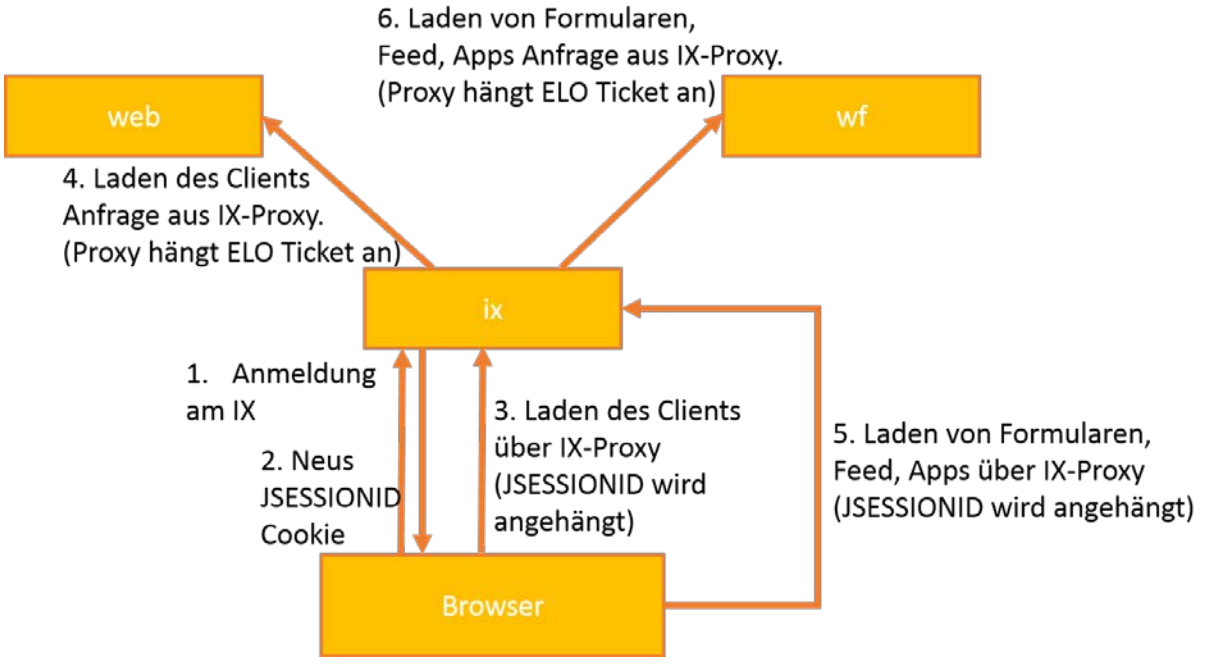
Dies ermöglicht es eine Web-Anwendung über die ELO Indexserver-JSON-API zu entwickeln, die kein gültiges ELO Ticket enthält und ihre Session-Information in einem HttpOnly-Cookie schützt.

1.8.3.2 ELO Indexserver-Proxy

Da das JSESSIONID-Cookie auf den Context-Pfad des ELO Indexservers beschränkt ist, wird es nur bei Anfragen an den ELO Indexserver verwendet und bei Anfragen an andere Web-Anwendungen (selbst wenn diese auf dem gleichen Apache Tomcat laufen) nicht gesendet.

Durch die ELO Ticket-Konstante und das HttpOnly-Flag gibt es keine Möglichkeit an das Session Ticket zu kommen, um dieses (wie bisher) als URL-Parameter an ein Formular, eine Feed-Seite, eine App oder sonstiges weiterzugeben.

Der ELO Indexserver liefert seit ELO 11 das OSGI-Plug-in *proxy-plugin* mit (genaue Version des Plug-ins in ELO 11 muss erfragt werden). Dieses ermöglicht es Web-Anwendungen hinter den Elo Indexserver zu „verlagern". Alle Anfragen werden an den ELO Indexserver gerichtet und dieser leitet die Anfragen an die entsprechende Web-Anwendung weiter. Dabei entscheidet der ELO Indexserver über ein einfaches, konfigurierbares URL-Schema, welche Anfrage an welche Anwendung weitergeleitet wird.



Die Abbildung zeigt, dass alle Anfragen über den ELO Indexserver laufen. Nach der initialen Anmeldung ist das JSESSION-Cookie vorhanden und wird bei allen zukünftigen Anfragen mitgesendet. Der Proxy prüft, ob die Session der JSESSIONID gültig ist und hängt für die Anfrage an die dahinterliegende Web-Anwendung das korrekte ELO Ticket an.

Im Browser gibt es zu keinem Zeitpunkt das korrekte ELO Ticket, da hier nur die Konstante verwendet wird.

Dadurch wird die erhöhte Sicherheit aus dem HttpOnly-Flag des JSESSIONID-Cookies erzielt.

1.8.3.2.1 Installation des proxy-plugins

Benötigt wird eine ELO 11-Installation (genaue Version von ELOix und Plug-in müssen erfragt werden). Es wird angenommen, dass alle Web-Anwendungen bereits installiert sind und laufen.

1. Prüfen Sie im ELO Installationsverzeichnis unter */prog/webapps/*, ob das Verzeichnis *ix-plugins* existiert und sich die Datei *proxy-plugin-<Versionsnummer>.jar* darin befindet.
2. Prüfen Sie, ob das Verzeichnis */prog/webapps/ix-plugins* als *osgiPluginDirectories* in den Einstellungen des ELO Indexservers eingetragen ist.
3. Erstellen Sie eine Konfigurationsdatei *de.elo.ix.plugin.proxy.properties* im Konfigurationsverzeichnis des ELO Indexservers (neben der config.xml) unter */config/ix-<Archivname>/<Tomcat>/*.
4. Füllen Sie die Datei mit folgenden Angaben:

```
/web/=http://<Web Client Tomcat>:<Port>/web-<Archivname>/
/wf/=http://<Web Forms Tomcat>:<Port>/wf-<Archivname>/
/analytics/=http://<Analytics Server>:<Port>/
```

   Die Kürzel vor dem = dürfen dabei nicht geändert werden.

5. Starten Sie den Indexserver neu.

Unter der folgenden Adresse erreichen Sie die entsprechenden Web-Anwendungen:

```
<IX Server Tomcat>/ix-<Archivname>/plugin/de.elo.ix.plugin.proxy/<Kürzel>/
```

Beispielsweise:

/ix-<Archivname>/plugin/de.elo.ix.plugin.proxy/web/

1.8.3.2.2 ELO Analytics

Damit ELO Analytics korrekt funktioniert, muss in der Konfigurationsdatei *kibana.yml* unter */config/eloanalytics/ELO-Analytics/* folgender Eintrag hinzugefügt werden:

```
server.basePath: "/ix-<Archivname>/plugin/de.elo.ix.plugin.proxy/analytics"
```

1.8.3.2.3 ELOas

Der ELOas kann hinter den Proxy konfiguriert werden, damit dieser für direkte Aufrufe z. B. aus gescripteten Formularen funktioniert. Zu beachten ist allerdings das der ELOas Regeln verwenden kann, die kein Benutzerticket benötigen, sondern einfach per Aufruf ausgelöst werden. Falls der ELO Indexserver in der DMZ steht, könnte jeder solche Regeln aus dem Internet auslösen. Aus diesem Grund sollte der ELOas nur hinter dem Proxy konfiguriert werden, falls ein direkter Zugriff unbedingt nötig ist. Kritische Regeln können in einem solchen Fall auf eine zweite ELOas Instanz ausgelagert werden, die nicht über den Proxy erreicht werden kann.

Falls der ELOas hinter den Proxy konfiguriert wird, erhält er in der Konfigurationsdatei das Kürzel /as/.

1.8.3.2.4 Konfiguration der publicUrlBase für Redirects

Um zu verhindern, dass Aufrufe weiterhin direkt an ELO Web Client und ELOwf gemacht werden, ohne über den ELO Indexserver-Proxy zu laufen, führen diese Anwendungen einen Redirect für alle Anfragen durch, die nicht über den Proxy gesendet werden.

Dazu prüfen Sie beim Start, ob ein Proxy eingerichtet wurde und leiten dann alle Anfragen, die nicht über den Proxy kommen (dieser fügt einen spezifischen Header der Anfrage hinzu), an den ELO Indexserver weiter.

Dazu wird die publicUrlBase-Einstellung aus dem ELO Indexserver verwendet. Diese sollte auf einer URL stehen, unter der der ELO Indexserver extern zugänglich ist.

„ELOix generates URLs based on this address for downloading documents from external systems. It must be visible to the external application and has to address an Indexserver instance. E.g. https://server:443/ix-Archive1/ix. If this parameter is not set, ixUrlBase is used."

Es wird angenommen, dass in einem ELO Web Client-Szenario mit externem Zugriff die publicUrlBase die URL ist, mit der auch der ELO Web Client extern den ELO Indexserver erreicht.

Fehlt die Konfiguration der publicUrlBase, wird entsprechend die ixUrlBase und als letzte Möglichkeit die in der config.xml konfigurierte interne ELO Indexserver-URL verwendet.

1.8.3.3 Auswirkungen

Durch das Einrichten des ELO Indexserver-Proxys ändern sich die URLs, mit denen der ELO Web Client und andere Anwendungen aufgerufen werden. Die Redirects sollten dafür sorgen, dass die alten Adressen auch weiterhin die richtigen Web-Anwendungen erreichen.

Der ELOas und ELO-Analytics führen keine solchen Redirects durch. Sollen diese direkt in Lösungen etc. angesprochen werden, könnten Anpassungen nötig sein.

Vorgelagerte Proxys (Apache, IIS, nginx) können auf den ELO Indexserver beschränkt werden, damit wird der direkte Zugriff unmöglich.



Beachten Sie: Durch die Freigabe des ELO Indexserver durch den Proxy ist auch der Zugriff auf alle dort konfigurierten Web-Anwendungen möglich. Bisher wurde zum Beispiel der ELOas selten in der DMZ durch einen Proxy (Apache, IIS, nginx) freigeschaltet. Wird dieser hinter den ELOix-Proxy gesteckt, kann er über eine ELO Indexserver-URL auch aus dem Internet erreicht werden. Hier muss der Zugriff eventuell eingeschränkt werden. Kritische Regeln sollten nie in der DMZ ausführbar sein.

1.8.3.4 Anpassungen in der ELO Administration Console

Auch die in der ELO Administration Console konfigurierten URLs werden durch den Proxy teilweise beeinflusst.

1.8.3.4.1 ELO Forms Services-URL

Der ELO Forms Services (ELOwf) liegt hinter dem Proxy. Um allen Clients hier bei jedem Aufruf ein Redirect auf die neue URL zu ersparen, sollte hier die Proxy-URL nach folgendem Schema konfiguriert werden:

```
http(s)://<Server>:<Port>/ix-<Archivname>/plugin/de.elo.ix.plugin.proxy/wf/
```

1.8.3.4.2 ELO Interface for Microsoft Office Online-URL

Das neue ELOimo Modul läuft ebenfalls hinter dem Proxy (Kürzel /imo/). Da dieser Eintrag durch die Clients ausgelesen wird, sollte hier ebenfalls die Proxy-URL nach folgendem Schema verwendet werden:

```
http(s)://<Server>:<Port>/ix-
<Archivname>/plugin/de.elo.ix.plugin.proxy/imo/
```

1.8.3.4.3 ELO Analytics-URL

Die ELO Analytics-URL wird durch den ELOwf ausgelesen. Dieser stellt den Clients die Kacheln zur Verfügung. Die URL wird im Proxy-Modus nur für die serverseitige Kommunikation von ELOwf zu ELO Analytics verwendet. Da diese Kommunikation bereits hinter dem Proxy stattfindet, muss hier keine Proxy-Adresse eingegeben werden.

```
http(s)://<Server>:<Port>/
```

1.8.3.4.4 Externe ELO Analytics-URL

Die externe URL wurde vom ELOwf als ELO Analytics-URL für die zusammengestellten Kacheln der Clients verwendet. Die Clients sprechen somit ELO Analytics über die externe URL an, während der ELOwf selbst über die normale URL kommuniziert. Im Falle des Proxy-Modus, verwendet der ELOwf statt der externen ELO Analytics-URL die publicIxUrlBase (siehe oben) um die Proxy-Adresse zu ELO Analytics an die Clients zu übergeben. Im Proxy-Modus wird diese URL somit nicht mehr verwendet.

1.8.3.4.5 ELO Online-Hilfe-URL

Die URL zur ELO Online Hilfe hat keine Relevanz für den Proxy-Modus, da hier ein komplett anderes (Archiv-fremdes) System angesprochen wird. Sie bleibt somit unverändert.

1.8.3.5 Eigene Anwendungen

Eigene Web-Anwendungen sollten ebenfalls hinter den Proxy konfiguriert werden, damit diese die gleichen Vorteile genießen und mit einem gültigen ELO Ticket versorgt werden. Falls dies nicht erwünscht ist müssen korrekte CORS-Filter in der Datei *web.xml* des ELO Indexservers konfiguriert werden, um Anfragen vom Server der Web-Anwendung zum ELO Server zu ermöglichen. Der CORS-Filter muss dazu alle hinter dem ELO Indexserver-Proxy befindlichen URLs abarbeiten.

Um eine Anwendung hinter den Proxy zu konfigurieren, reicht es aus, der Anwendung ein eigenes Kürzel in der Proxy-Konfigurationsdatei, zusammen mit der URL, wo die Anwendung läuft, zu geben.

Bei der Implementierung der Anwendung gibt es folgende Punkte zu beachten.

1.8.3.5.1 ELO Ticket

Das ELO Ticket wird durch den Proxy als Cookie angehängt (Name: ‚ticket'). Dies bedeutet, falls die Anwendung mit einem gültigen JSession-Cookie über den Proxy angesprochen wird, kann das ELO Ticket serverseitig aus dem ticket-Cookie ausgelesen und für ein *createFromCookie* verwendet werden.

1.8.3.5.2 Check auf Proxy Request

Um zu erkennen, ob eine Anfrage über den Proxy gestellt wurde oder direkt an die Anwendung gerichtet war, kann ein entsprechender http-Request-Header ausgelesen werden, der durch das Proxy-Plugin gesetzt wird. Fehlt dieser Header, wurde die Anfrage nicht über den Proxy gerichtet. Der Headername ist ‚ELO-IXPROXY', der Wert selbst spielt dabei keine Rolle.

1.8.3.5.3 Ticket Leaks

Damit die Sicherheit bestehen bleibt, darf das ELO Ticket nicht als Antwort oder Inhalt in irgendeiner Form an den Browser (oder Steller der Anfrage) zurückgegeben werden.

1.9 Überarbeitung der Vertretungsregel

Das Konzept der Vertretungsregel wurde zu ELO 12 komplett überarbeitet.

An dieser Stelle werden die neuen API-Funktionen und Änderungen in der Berechtigungsübernahme aus Sicht des ELO Indexservers dargestellt.

1.9.1 Wichtige Hinweise

- Damit ein Nutzer alle Rechte durch eine neue/geänderte/aktivierte Vertretungsregel erhält bzw. entzogen bekommt, muss der Vertreter sich erneut anmelden.

- Von einem Nutzer zu einem Vertreter können mehrere Vertretungen bestehen. Die Rechte werden akkumuliert.
- Es können nur die Rechte von Gruppen, die das Flag2 *AccessC.FLAG2_CAN_BE_SUBSTITUTED* besitzen, weitergegeben werden. Dieses Recht ist neu seit ELO 12 und wird nicht automatisch für Gruppen oder Nutzer hinzugefügt. Das Recht kann nur Gruppen hinzugefügt werden und wird nicht durch Gruppenhierarchien vererbt bzw. durch Vertretungen weitergegeben.
- Definition Vorgesetzter: Die Untergebenen von `“Manager“` werden wie folgt definiert/ermittelt:
    - Alle Nutzer/Gruppen, die `UserInfo.superiorId=“Manager“` gesetzt haben
    - `“Manager“` ist in der Gruppe `“Manager Group“`: Alle Nutzer/Gruppen, die `UserInfo.superiodId=“Manager Group“` gesetzt haben
    - `“Team Group“` hat als `UserInfo.superiorId=“Manager“` oder `“Manager Group“`: gesetzt: Alle Mitglieder der Gruppe *Team* (direkt, nicht rekursiv)
        - d. h. `“Sub Team Group“` (Untergruppe von `“Team Group“` `UserInfo.superiorId` nicht explizit gesetzt, `UserInfo.groups=“Team Group“`) hat als Vorgesetzten NICHT `„Manager“`, da dabei eine rekursive Gruppensuche notwendig ist.

1.9.2 Übernahme bestehender Vertretungsregeln

Bestehende Vertretungsregeln werden bei dem ersten Start des ELOix nach dem Update auf ELO 12 in das neue System übernommen. Da die Überarbeitung ein vollständiges Übernehmen der bestehenden Regeln jedoch nicht erlaubt, werden nur die Werte für `„Nutzer“` und `„Vertreter“` übernommen. Der Benutzer muss für jede bestehende Vertretungsregel die Gruppen, deren Rechte weitergegeben werden, sowie die Zeiträume und weitere Einstellungen manuell eintragen.

1.9.3 API-Funktionen

- **checkinSubstitutions()**: Hinzufügen von neuen und Ändern von bestehenden Vertretungen.
- **checkoutSubstitutions()**: Auslesen von Vertretungen. Diese können z. B. nach Nutzer, Vertreter, GUID oder Ersteller ausgewählt werden.
- **deleteSubstitutions()**: Löschen von Vertretungen
- **activateSubstitution()**: Manuelles Aktivieren einer bestehenden Vertretung, sofern diese es erlaubt
- **deactivateSubstitution():** Manuelles Deaktivieren einer bestehenden, manuell aktivierten Vertretung

- forwardSubstitution(): Weiterleiten bzw. Abgeben an einen anderen Nutzer (neuer Vertreter) einer bestehenden Vertretung.
- createSuperiorSubstitution(): Als Vorgesetzter eine Vertretung für einen Untergebenen erstellen.

Die API-Funktionen, sowie deren Parameter, sind in den JavaDocs dokumentiert.

Ist das Ausführen einer Funktion nicht möglich (z. B. falsch übergebene Parameter), wird immer eine Exception geworfen.

1.9.4 Wichtige Änderungen in der Berechtigungsübernahme

1.9.4.1 Workflows

Damit der Vertreter eines Nutzers einen Knoten des Nutzers sieht, muss der Knoten ursprünglich der Gruppe zugewiesen sein, deren Rechte weitergegeben werden. War der Knoten direkt dem Nutzer zugeordnet, ist es einer Vertretung nicht möglich, diesen Knoten zu sehen.

Bestehende Workflows müssen ggf. an das aktuelle Gruppenkonzept angepasst werden.

Damit ein Workflowknoten vom Vertreter bearbeitet oder weitergeleitet werden kann, muss der Workflowknoten vorher vom Vertreter mit *takeWorkflowNode()* angenommen werden.

Der Vertreter kann diesen Knoten auch nach der Beendigung der Vertretung weiterbearbeiten, sofern er vorher *takeWorkflowNode* aufgerufen hat und der Knoten demnach ihm „gehört". Der zu vertretene Nutzer kann sich diesen Knoten nicht wieder zurückholen. Dazu muss der Vertreter ihm den Knoten weiterleiten.

Wenn für einen Knoten der Wert *WfNode.department2* gesetzt ist, kann ein Vertreter über *findFirstTasks* den Workflow nur finden, wenn `FindTasksInfo.inclDeputy = true` gesetzt ist, auch wenn der Vertreter den Knoten schon angenommen hat.

1.9.4.2 ProcessReleaseLock

Bisher hat ein aktiver Vertreter die SORDs, die von dem Vertretenen gesperrt waren, entsperren können. Mit der Umstellung auf Übernahme von Gruppenberechtigungen ist dies nicht mehr abbildbar.

Der Vertreter kann jetzt diese SORDs nur dann entsperren, wenn er selbst Hauptadministrator ist oder das Recht *Hauptadministrator* durch eine aktive Vertretung erhält.

1.9.4.3 PublicDownloads

Ein Nutzer, der nicht das Recht *Hauptadministrator* hat, kann nur noch seine eigenen PublicDownloads sehen und nicht die von vertretenen Nutzern, da PublicDownloads immer nutzerspezifisch sind.

1.9.5 Report

Jegliche API-Aufrufe der Vertretungen lösen das Schreiben eines Report-Eintrags aus. Die neuen Report-Codes sind in der Klasse *ReportInfoC* zu finden und werden alle unter dem Report-Code *ReportOptionsC.ERP_SUBSTITUTIONS* zusammengefasst.

Zu jedem Report-Eintrag wird die dazugehörige Vertretung gespeichert.

Weitere Details sind in den JavaDocs zu finden.

1.10 WebDAV Erweiterung für SAP ILM

Die Spezifikation *CERTIFICATION SPECIFICATION – BC-ILM 3.0 WEBDAV STORAGE INTERFACE FOR NETWEAVER ILM* wurde implementiert. Diese Erweiterung steht für ELO ab Version 9 zur Verfügung.

1.10.1 Wichtige Anmerkungen

Zur Speicherung der Eigenschaften von SAP wird die Verschlagwortungsmaske *WebDAV ILM* automatisch angelegt, aber nur wenn die erste Anfrage von SAP kommt. Die Nutzung dieser Erweiterung erfordert die gültige Lizenz zur Verwendung der SAP-ArchiveLink-Schnittstelle.

Falls ArchiveLink-Retention verwendet wird: Der Service-Benutzer von *ELO ArchiveLink for SAP* bzw. von *ELO Smart Link for SAP* darf nicht das Recht *Alle Einträge sehen, Berechtigungen ignorieren* haben. Sonst kann er Dokumente über ArchivLink löschen, obwohl er kein Recht dafür hat.

Es gibt sonst keine weiteren Systemvoraussetzungen.

1.11 Erweiterung des Verschlüsselungskonzepts

Ab Version 12 wird die Dokumentenverschlüsselung und die Verwaltung der Schlüsselkreise grundlegend erweitert. Im Folgenden wird beschrieben, welche Komponenten geändert bzw. erweitert wurden.

1.11.1 Verschlüsselungsalgorithmus AES

Für die Verschlüsselung von Dokumenten und auch der Schlüsselkreisinformationen selbst wird auf das Verfahren **AES** umgestellt. Dabei wird eine Schlüssellänge von 256 Bit benutzt, was die größtmögliche und damit die sicherste zur Verfügung stehende Länge darstellt. Die Verwendung dieser Schlüssellänge erfordert ab JavaVM Version 1.8u152 keine gesonderten Einstellungen mehr, d. h. es existieren nun keine Beschränkungen mehr im Sinne der „JCE Unlimited Strength Jurisdiction Policy". Damit muss der ELOix mindestens mit o. g. JavaVM oder höher betrieben werden, was aber durch Umstellung auf OpenJDK 11 automatisch erfüllt ist.

Bisher wurde der Twofish-Algorithmus verwendet, der eine vergleichbare Sicherheit bietet, allerdings im ECB-Modus. Dieser Modus ist u. U. anfällig für Klartextangriffe.

Wir verwenden deshalb AES im CBC-Modus (Cipher Block Chaining). Dabei ist die Verschlüsselung eines Textblockes vom Ergebnis der Verschlüsselung der vorangehenden Blöcke abhängig. Um hiermit den ersten Textblock verschlüsseln zu können, ist ein sogenannter Initialisierungsvektor erforderlich, der für jede Verschlüsselung mittels Zufallszahlen erzeugt wird.

Für jede Verschlüsselung eines Dokuments wird stets ein neu erzeugter Initialisierungsvektor verwendet, um Angriffe durch Vergleich des ersten Chiffrat-Blocks bei bekannten verschlüsselten Dokumenten zu verhindern.

Die Verwendung des CBC-Modus hat zur Folge, dass keine verschlüsselten Teildokumente, sondern nur noch komplette Dokumente ausgeliefert werden können. Diese Funktion wurde bisher von Clients für verschlüsselte Dokumente aber nicht genutzt und stellt deshalb keine Einschränkung dar.



Beachten Sie: Bereits im Archiv existierende und mit Twofish verschlüsselte Dokumente können weiterhin genutzt werden. Das Twofish-Verfahren und die entsprechend angelegten Schlüsselkreise werden weiterhin unterstützt. Es können auch weiterhin mit den Twofish-Schlüsselkreisen Dokumente verschlüsselt werden. Zusätzliche Funktionen des neuen Verschlüsselungskonzeptes (z. B. verschlüsselte Volltextinformationen) können aber nur mit dem neuen Verfahren und den dafür angelegten Schlüsselkreisen realisiert werden.

1.11.2 Verschlüsselung und Entschlüsselung durch Server

Bisher war es möglich, dass Verschlüsselung und Entschlüsselung auch auf der Clientseite durchgeführt werden konnten. Nachteilig bei diesem Verfahren war, dass der interne Key (der nicht änderbar ist) an den Client verraten wird. Ein kompromittierter Client könnte also einen dauerhaften Zugriff auf verschlüsselte Dateien auch nach einer Passwortänderung ermöglichen.

Aus diesem Grund wird ab der Version 12 die Verschlüsselung immer auf der Serverseite durchgeführt. Der Client liefert bei einer Anfrage das jeweilige Passwort (in verschlüsselter Form) mit. Der AES-Key verlässt niemals den Server.

Der Server hält keine entschlüsselten temporären Dateien vor. Wenn ein Dokument mehrfach angefordert wird, wird es jedes Mal neu entschlüsselt.



Beachten Sie: Die Session-Optionen einer Client-Session waren bisher so eingestellt, dass Verschlüsselung und Entschlüsselung auf der Client-Seite stattfinden. Die serverseitige Ver- und Entschlüsselung musste explizit mittels API-Methode *setSessionOptions()* separat eingeschaltet werden. Dies ist nun umgekehrt: die initialen Werte der Session-Optionen sind nun auf serverseitige Ver- und Entschlüsselung eingestellt.

1.11.3 Erweiterte Schlüsselkreise

1.11.3.1 Anzahl

Bisher gab es eine feste Anzahl von 16 Schlüsselkreisen. Dies wurde nun erweitert auf `INTEGER.MAX_VALUE`, d. h. über zwei Milliarden mögliche Schlüsselkreise. Dabei wird die ID eines Schlüsselkreises vom Server vergeben (laufende Nummer einer Datenbank-Sequenz).

Diese Erweiterung hat zur Folge, dass die Datenbank-Tabellen *objekte* und *docmasks* jeweils um ein Feld *encryptionset* erweitert worden sind. Dieses Feld enthält die ID des benutzten Schlüsselkreises.

Die Schlüsselkreis-ID ist intern weiter Bestandteil der `Sord.details` bzw. `DocMask.details`.

Die Schlüsselkreis-ID ist weiterhin zu jeder Dokumentenversion in der Tabelle *elodmdocs* in der Spalte *cryptno* zu finden.

1.11.3.2 Getrennte Bereiche für Benutzer- und Systemzugriffe

Grundsätzliches Prinzip der Schlüsselkreise ist die Verwendung eines Passwortes zur Verschlüsselung eines Keys, mit dem dann die eigentliche Verschlüsselung von Dokumenten durchgeführt wird. Das Passwort ist nur dem Benutzer bekannt und wird beim Anlegen des Schlüsselkreises in der ELO Administration Console eingegeben. Mit diesem Passwort verschlüsselt der Server den (mit Hilfe von Zufallszahlen) ermittelten Key und speichert den verschlüsselten Key zusammen mit den weiteren Schlüsselkreisinformationen (z. B. ID, Name, …) in der Datenbank ab. Das Passwort ist später jederzeit änderbar, der Key eines Schlüsselkreises bleibt gleich.

Ziel ist es, auch für verschlüsselte Dateien Volltextinformationen und Preview-Dateien anbieten zu können. Dies wird nun ermöglicht durch die Einführung eines Systemteils im Schlüsselkreis. Dabei wird, zusätzlich zum Benutzerpasswort, ein Systempasswort vergeben, mit dem der Key ebenfalls verschlüsselt und in der Datenbank abgelegt wird. Beim Anlegen eines Schlüsselkreises mit Systemteil muss ein Systembenutzer oder eine Gruppe angegeben werden, in deren Profiloptionen der Server das Systempasswort in verschlüsselter Form ablegt (getrennt für jeden so angelegten Schlüsselkreis). Dadurch wird diesem Systembenutzer ermöglicht, für z. B. Extrahieren des Volltextinhalts ein verschlüsseltes Archivdokument zu entschlüsseln.

Jeder, ab Version 12 angelegte, Schlüsselkreis ist nun also für zwei Sicherheitsstufen ausgelegt:

- Standardsicherheit: Der Schlüsselkreis enthält neben dem Benutzerteil auch einen Systemteil mit getrenntem Passwort. Systemfunktionen wie z. B. Volltext für verschlüsselte Dokumente wird damit ermöglicht.
- Erweiterte Sicherheit: Der Schlüsselkreis enthält keinen Systemteil. Nur der Besitzer des Benutzerpasswortes hat Zugriff. Verschlüsselte Dokumente können nicht in den Volltext aufgenommen werden und es kann auch kein Preview erzeugt werden.

Ein Schlüsselkreis, der keinen Systemteil enthält, kann auch nachträglich umkonfiguriert werden und der Systemteil erzeugt werden. Ebenso ist es möglich, den Systemteil nachträglich zu entfernen. Damit fallen dann die Systemfunktionen (Volltext usw.) weg. Bereits erzeugte Volltextinformationen bleiben für den Benutzer sichtbar.

Die nachträgliche Änderung des Systembenutzers eines Schlüsselkreises ist ebenfalls möglich.



Beachten Sie: Für bestehende Twofish-Schlüsselkreise kann kein Systemteil erzeugt werden. Die Systemfunktionen sind für diese Schlüsselkreise also nicht nutzbar.

1.11.3.3 Verschlüsselungsverfahren AES

Wie für die Verschlüsselung der Dokumente wird auch für die Verschlüsselung der Key-Informationen das AES-Verfahren im CBC-Modus mit 256 Bit Schlüssellänge verwendet.

Dies hat zur Folge, dass auch hier jeweils ein Initialisierungsvektor erzeugt werden muss. Der Initialisierungsvektor ist nicht geheim, darf aber nur ein einziges Mal verwendet werden. Deshalb wird z. B. bei einer Passwort-Änderung für einen Schlüsselkreis auch jeweils ein neuer Initialisierungsvektor erzeugt, der dann neben dem Passwort zur Verschlüsselung des Keys verwendet wird.

Für Benutzer- und Systemteil werden unterschiedliche Initialisierungsvektoren verwendet.

1.11.3.4 Neue Datenbank-Felder für Schlüsselkreis-Informationen

Schlüsselkreisinformationen werden wie bisher in der Tabelle *elodmcrypt* abgelegt.

Diese Tabelle wurde für die neuen Schlüsselkreisinformationen wie folgt erweitert:

cryptno	ID des Schlüsselkreises, wird nun automatisch vergeben
cryptname	Name des Schlüsselkreises, wurde erweitert auf 64 Zeichen
cryptpwd	Enthält die verschlüsselte KeyInfo für Twofish-Schlüsselkreise. Verwendung nur bei Twofish-Schlüsselkreisen.
cryptversion	**Neues Feld**: VersionsInfo (String) zur Kennzeichnung der Elo-Verschlüsselungs-Art (z. B. AES_v1). Leer bei TwoFish-Schlüsselkreisen.
cryptuser	**Neues Feld**: Enthält die verschlüsselte Benutzer-KeyInfo für AES-Schlüsselkreise. Bleibt leer bei Twofish-Schlüsselkreisen.
cryptsystem	**Neues Feld**: Enthält die verschlüsselte System-KeyInfo für AES-Schlüsselkreise. Bleibt leer bei Twofish-Schlüsselkreisen.
ivuser	**Neues Feld**: Enthält den Initialisierungsvektor für die Benutzer-KeyInfo. Bleibt leer bei Twofish-Schlüsselkreisen.
ivsystem	**Neues Feld**: Enthält den Initialisierungsvektor für die System-KeyInfo. Bleibt leer bei Twofish-Schlüsselkreisen.
systemuserid	**Neues Feld**: Enthält die ID des Systembenutzers (bzw. -gruppe) bei Verwendung des Systemteils.

1.11.3.5 Neue API Funktionen

Die API für die Schlüsselkreisverwaltung wurde um zwei Funktionen erweitert:

1. createNewEncryptionSet(): Zum Anlegen neuer Schlüsselkreise (mit oder ohne Systemteil), wird benutzt von der ELO Administration Console
2. provideSystemCryptPassword(): Zum Bereitstellen der Systempasswortinformation durch Systembenutzer, beispielsweise für Zwecke der Volltextextrahierung

Für das Ändern von Schlüsselkreisdaten wird die bisherige Methode *checkinCryptInfos()* benutzt, die intern die Klasse *CryptInfo* benutzt, deren Felder entsprechend erweitert wurden.

1.11.4 Dokumentenverschlüsselung

Verschlüsselte Dokumente werden wie bisher am ELOdm abgelegt. Die Dateiendung war bisher „.ETF". Mit AES verschlüsselte Dateien tragen nun die Endung „.ECF". Für Dokumente, die weiterhin mit den bestehenden Twofish-Schlüsselkreisen verschlüsselt werden, ist die Endung weiterhin „.ETF".

Eine ECF-Datei enthält einen unverschlüsselten und einen verschlüsselten Header sowie die verschlüsselten Nutzdaten. Der unverschlüsselte Header enthält zu Beginn eine Zeichenkette (sog. „Magic") zur Erkennung. Diese Zeichenkette lautet „EloCrypt". Außerdem sind enthalten: die Versionsinformation, die Dateierweiterung des Originaldokuments, die ID des Schlüsselkreises, die Länge der Nutzdaten und der Initialisierungsvektor.

Mit Hilfe des Initialisierungsvektors und mit Kenntnis des Keys lässt sich die Datei entschlüsseln.

Der verschlüsselte zweite Header enthält einen Prüfstring zum Test, ob die Datei mit dem benutzten Key entschlüsselt werden kann. Der zweite Header und die Nutzdaten sind im CBC-Modus zusammen verschlüsselt.

1.11.5 Verschlüsselte Volltextinformation

Ab Version 12 wird die Volltextinformation für verschlüsselte Dokumente ermöglicht. Die Volltextinformation wird ebenfalls verschlüsselt am ELOdm abgespeichert. Dabei kommt der gleiche Schlüsselkreis zum Einsatz, der zur Verschlüsselung der zugrundeliegenden Dokumentenversion verwendet wurde.

Der Textreader kann bei Kenntnis des Systempasswortes (enthalten in den Profiloptionen) des jeweiligen Schlüsselkreises die Entschlüsselung durch den Server veranlassen, den Volltext extrahieren und anschließend durch den Server verschlüsseln und ablegen lassen.

Wenn der ELO Textreader das Systempasswort nicht kennt, wird kein Volltext extrahiert. Dies ist z. B. der Fall bei:

- Der verwendete Schlüsselkreis enthält keinen Systemteil („Erweiterte Sicherheit").

- Der Systembenutzer des Textreaders ist nicht der, der als *SystemUser* im Systemteil des Schlüsselkreises angegeben ist und gehört auch der Gruppe nicht an, falls im Systemteil eine Gruppe angegeben wurde.
- Der verwendete Schlüsselkreis ist ein bestehender Twofish-Schlüsselkreis, angelegt vor Version 12 und besitzt keinen Systemteil.
- Der Systemteil wurde in der Konfiguration des Schlüsselkreises entfernt (per ELO Administration Console)

1.11.6 Verschlüsselte Vorschaudateien

Ab Version 12 werden Vorschaudateien für verschlüsselte Dokumente ermöglicht. Die Vorschaudatei wird ebenfalls verschlüsselt am ELOdm abgespeichert. Dabei kommt, wie beim Volltext, der gleiche Schlüsselkreis zum Einsatz, der zur Verschlüsselung der zugrundeliegenden Dokumentenversion verwendet wurde.

Auf der Client-Seite muss das Passwort für den Schlüsselkreis des zugrunde liegenden verschlüsselten Dokuments eingegeben worden sein, damit der Server die Vorschauinformationen entschlüsseln und dem Client zur Verfügung stellen kann.

Für eine Reihe von Dateitypen kann der ELO Preview Converter Vorschaudateien erzeugen. Für diesen gelten die gleichen Voraussetzungen wie für den ELO Textreader: Der ELO Preview Converter kann, bei Kenntnis des Systempasswortes (enthalten in den Profiloptionen) des jeweiligen Schlüsselkreises, die Entschlüsselung durch den Server veranlassen, die Dokumentversion lesen, eine Vorschaudatei erzeugen und anschließend durch den Server verschlüsseln und ablegen lassen.

Wenn der ELO Preview Converter das Systempasswort nicht kennt, wird keine Vorschaudatei erzeugt. Dies ist z. B. der Fall bei:

- Der verwendete Schlüsselkreis enthält keinen Systemteil („Erweiterte Sicherheit“).
- Der Systembenutzer des ELO Preview Converters ist nicht der, der als *SystemUser* im Systemteil des Schlüsselkreises angegeben ist und gehört auch der Gruppe nicht an, falls im Systemteil eine Gruppe angegeben wurde.
- Der verwendete Schlüsselkreis ist ein bestehender Twofish-Schlüsselkreis, angelegt vor Version 12 und besitzt keinen Systemteil.
- Der Systemteil wurde in der Konfiguration des Schlüsselkreises entfernt (per ELO Administration Console).

Für bestimmte Bild-Dateitypen (z. B. JPG) erzeugt der Image-Prozessor im ELO Indexserver selbst die Vorschaudateien. Dies wird jeweils von der Clientseite getriggert (z. B. ELO Web-Client). Hierbei entschlüsselt der Server das (Bild-)Dokument, erzeugt die Vorschaudatei und legt sie verschlüsselt am Server ab, falls das Originaldokument verschlüsselt ist. An den Client wird die entschlüsselte Vorschauinformation geliefert. Die Voraussetzung ist natürlich, dass der Client das korrekte Passwort des Schlüsselkreises vom Benutzer erhält und in der ELOix-Session mitliefert (`provideCryptPassword()`).

1.11.7 ELO iSearch indexiert verschlüsselten Volltext

Für Dokumente, die mit einem Schlüsselkreis verschlüsselt sind, der im Modus *Standardsicherheit* (siehe Abschnitt *1.11.3.2 Getrennte Bereiche für Benutzer- und Systemzugriffe*) konfiguriert ist, wird der Index-Prozess der ELO iSearch auch die Volltextinformationen für die Suche mit in den Index aufnehmen. Im Modus *Standardsicherheit* wird der vorhandene Systemteil des Schlüsselkreises genutzt, um den im Archiv abgelegten Volltext zu entschlüsseln und dem Index-Prozess zuzuführen.

Die Voraussetzungen, dass der Volltext von verschlüsselten Dokumenten in der ELO iSearch durchsuchbar ist, sind demnach:

- Der verwendete Schlüsselkreis besitzt einen Systemteil (Modus *Standardsicherheit*).
- Der ELO Textreader kann das Dokument vom Server entschlüsseln lassen. Das ist dann der Fall, wenn der Systembenutzer des ELO Textreaders die erforderlichen Systempasswörter kennt (siehe Abschnitt *1.11.5 Verschlüsselte Volltextinformation*).
- Der Systembenutzer für den Update-/Re-Index-Prozess der ELO iSearch (in den meisten Fällen. *ELO Service*) kennt ebenfalls die erforderlichen Systempasswörter. Dies ist entsprechend bei der Konfiguration des Schlüsselkreises beim hinterlegten Systembenutzer bzw. -Gruppe entsprechend einzustellen.



Beachten Sie: Nach Indexierung des Volltextes von verschlüsselten Dokumenten werden über die ELO iSearch die entsprechenden Dokumente gefunden, wobei unerheblich ist, ob der Benutzer im Client das Passwort für den entsprechenden Schlüsselkreis eingegeben hat oder nicht. Den Inhalt des Dokumentes anzeigen lässt sich aber dann im Client nur mit Kenntnis des Passwortes.

Falls nicht gewünscht ist, dass jeder die Dokumente als Treffer angezeigt bekommt, kann die komplette Erzeugung des Volltextes für verschlüsselte Dokumente unterbunden werden, indem der Modus *Erweiterte Sicherheit* (siehe Abschnitt *1.11.3.2 Getrennte Bereiche für Benutzer- und Systemzugriffe*) für den Schlüsselkreis eingestellt wird. Alternativ kann im Modus *Standardsicherheit* der Systembenutzer des Schlüsselkreises so eingestellt werden, dass er nicht dem Systembenutzer des ELO Textreaders entspricht bzw. der Systembenutzer des ELO Textreaders nicht der Gruppe angehört, falls im Schlüsselkreis eine Gruppe als Systembenutzer angegeben ist.

1.11.8 Neue Konstanten

CryptInfoC.ENCRYPTION_VERSION_AES_V1	Zur Markierung der benutzten Verschlüsselungsversion
CryptInfoC.DELETE_SYSTEM_PART	Zur Markierung, dass der Systemteil eines Schlüsselkreises gelöscht werden soll. Verwendung im Feld *systemPwd* einer CryptInfo-Klasse.
IXServicePortC.CRYPT_INFO	Neue Klasse für CryptInfo-Konstanten
UserProfileC.KEY_PREFIX_SYSTEM_ENCRYPTION_PASSWORD	String zur Markierung der Systempassworteinträge in der ProfileOpts-Tabelle
DetailUtils.MASK_ALL_ENCRYPTIONSET_FLAGS	Interne Benutzung, zur Auswertung der SORD-Details und DocMask-Details

1.12 Änderungen am Datenbankschema

1.12.1 Primary Keys

Für neue Archive werden ab ELO 12 bei einigen Tabellen Primary Keys verwendet. Primary Keys sind Indexe, von denen eine Tabelle höchstens einen haben kann und dessen Spalten einen Eintrag in der Tabelle eindeutig identifizieren.

Für Microsoft SQL Server ergibt sich jedoch eine weitere Eigenschaft: Die Zeilen in der Tabelle werden der Sortierung des Primary Keys entsprechend auf dem Speichermedium abgelegt. Dadurch ergeben sich Leistungsvorteile beim Lesen von Elementen, die aufeinanderfolgend abgespeichert sind. Auch entfällt eine Indirektion beim Lesen von Daten aus der Tabelle, wenn über den Primary Key selektiert wird: da die Leseoperation bereits auf der korrekten Zeile in der Tabelle ist, können weitere Spalten direkt gelesen werden ohne von einem Index zur entsprechenden Zeile in der Tabelle springen zu müssen. Dass die Tabelle nun sortiert gespeichert vorliegt, hat jedoch nicht nur Vorteile. Denn es eignen sich nur Spalten für einen Primary Key, die dafür sorgen, dass neue Zeilen möglichst am Ende der Tabelle eingefügt werden. Aus diesem Grund finden sich nur auf ausgewählten Tabellen Primary Keys.

1.12.1.1 Migration von normalen Indexen

Solange die Mindestvoraussetzungen erfüllt sind, übernimmt das ELO Server Setup die Migration von normalen Indexen nach Primary Keys. Die Einschränkungen entnehmen Sie dem nächsten Abschnitt.

1.12.1.2 Einschränkungen bei der Migration und deren Umgehung

Da das Anlegen von Primary Keys mit dem Umsortieren von Zeilen in der entsprechenden Tabelle einhergeht, kann dies bei großen Tabellen eine lange Zeitspanne in Anspruch nehmen. Aus diesem Grund findet die automatische Migration durch das ELO Server Setup nur dann statt, wenn die Tabelle *objkeys* weniger als 5.000.000 Einträge enthält. Dies gilt dann für alle Primary Keys auf allen Tabellen, nicht nur der objkeys-Tabelle.

Ist die Verwendung von Primary Keys dennoch gewünscht, können Sie das Setup durch ein Flag anweisen, das Erstellen von Primary Keys unabhängig der Tabellengröße von *objkeys* durchzuführen. Das Flag setzen Sie in der Datenbank mit folgendem Statement vor dem Start des ELO Server Setups:

```
insert into eloixopt (ixid, optname, optval)
     values ('_ALL', 'ix.update.primary_keys', 'true')
```



Beachten Sie: Die oben genannte Einstellung ist archivspezifisch. Sind mehrere große Archive vorhanden, können Sie damit kontrollieren, welche davon auf Primary Keys umgestellt werden.

1.13 Änderungen im Textreader

1.13.1 Allgemeine Änderungen

1.13.1.1 Java 11

Der ELOtr ist auf Java 11 umgestellt worden.

1.13.1.2 Verschlüsselte ELO-Dokumente

Der Textreader kann bei Kenntnis des Systempasswortes (enthalten in den Profiloptionen) des jeweiligen Schlüsselkreises die Entschlüsselung durch den Server veranlassen, den Volltext extrahieren und anschließend durch den Server verschlüsseln und ablegen lassen (siehe Abschnitt *1.11.5 Verschlüsselte Volltextinformation*).

1.13.1.3 Verbindung zum ELO Indexserver

Auch nach einer Fehlermeldung des ELO Indexserver versucht der ELO Textreader weiterhin sich anzumelden.

1.13.1.4 Statusseite

Exporter, Importer und die Konverter-Threads können auf der Statusseite ein- und ausgeschaltet werden (und nicht nur in der config.xml).

1.13.1.5 Logging im 'Reportlog'

Ab Version 12 des ELO Textreaders erfolgen die wichtigsten Logging-Ausgaben zusätzlich in dem sogenannten *Reportlog*. Dort wird nur protokolliert, ob ein Dokument exportiert, importiert, konvertiert oder an einen anderen Konverter oder eine andere Anwendung weitergegeben wurde (insbesondere die OCR). Ebenfalls wird mit Angabe des Grundes protokolliert, wenn ein (Teil-)Dokument nicht konvertiert werden konnte.

Die Protokollierung kann eingeschränkt oder erweitert werden durch die Konfiguration des Log-Levels in der *log4j.properties*-Datei:

- Im Level *Error* wird protokolliert, wenn ein Dokument nicht konvertiert werden konnte.
- Im Level *Warn* werden zusätzlich sonstige Probleme protokolliert. Wenn z. B. einige Anhänge aus EML- oder PDF-Dateien nicht extrahiert werden konnten.
- Im Level *Info* wird auch protokolliert, wenn ein Dokument konvertiert worden ist.
- Im Level *Debug* wird schließlich protokolliert, wenn ein Dokument exportiert oder importiert werden konnte oder an eine andere Anwendung übergeben worden ist. Beispielsweise Bilddateien an die OCR.

Die Ausgabe erfolgt in einem festgelegten Format, erklärt am Beispiel der Datei *00000003.xls*:

```
INFO  - HexId >00000003< DocId >3< Filename >00000003.xls< Converted
>C:\ELO11\data\tr-elo110\xls\00000003.xls<
```

Ausgegeben wird sowohl in hexadezimaler als auch dezimaler Form die ELO Dokumenten-ID des zu konvertierenden Dokuments, dann der Dateiname und der Dateipfad, jeweils in spitzen Klammern (zum besseren Filtern der Logging-Datei).

Bei Elementen von sogenannten Container-Dokumenten wird als HexId und DocId immer die Dokumenten-ID des ELO Dokuments ausgegeben. Wenn also z. B. die Datei *0000005E.zip* eine HTML-Datei enthält, wird diese unter dem Namen *0000005E_000.html* extrahiert. Nach der Konvertierung dieser HTML-Datei enthält die Logging-Datei den Eintrag:

```
INFO  - HexId >0000005E< DocId >94< Filename >0000005E_000.html< Converted
>C:\ELO11\data\tr-elo110\htm\0000005E_000.html<
```

Die Datei *log4j.properties* muss angepasst werden, bspw.

```
…
log4j.category.reportlog=info, reportlog
log4j.additivity.reportlog=false
…
log4j.appender.reportlog=org.apache.log4j.RollingFileAppender
```

```
log4j.appender.reportlog.File=e:/temp/log/reportlog.log
# DatePattern: one file each day:
log4j.appender.reportlog.DatePattern='.'yyyy-MM-dd'.txt'
log4j.appender.reportlog.layout=org.apache.log4j.PatternLayout
log4j.appender.reportlog.layout.ConversionPattern=%d{ABSOLUTE} %t %1x %-5p - %m%n
# if append is true, the log file is not deleted when the program is started:
log4j.appender.reportlog.append=true
```

Das Log-Level kann zusammen mit dem Log-Level der Standard-Log-Datei auf der Statusseite des ELO Textreaders eingestellt werden.

1.13.1.6 Speicherplatzüberprüfung

- Windows Server 2016 wird erkannt und entsprechende Standardwerte zur Berechnung des minimalen Plattenplatzes in einer Systempartition gesetzt. Für alle Windows Server-Betriebssysteme ab 2016 wird 32 GB als minimaler Plattenplatz der ELO Textreader-Verzeichnisse in einer Systempartition angenommen.
- Die Konverter stoppen nur, wenn der Importer nicht mehr arbeitet, und nicht, wenn kein Speicherplatz für konvertierte Dateien mehr vorhanden ist.
- Die Konverter stoppen aber nicht, wenn der Importer in der config.xml oder auf der Statusseite ausgeschaltet worden ist.
- Der Speicherplatz für Textdateien wird nicht mehr überprüft (wird nur überprüft beim Exportieren aus dem Archiv).
- Der PDF-Konverter verwirft die Konvertierung eines PDF-Dokuments, wenn durch die Extraktion von Bildern oder Attachments der minimale Plattenplatz überschritten würde. Die Konvertierung wird erneut versucht, wenn ausreichender Plattenplatz vorhanden ist.
- Nach Speicherplatzüberschreitungen wartet der Textreader nicht mehr 10 Minuten, sondern nur eine Minute.
- Als Default wird die maximale Anzahl der Dateien in Import- oder Exportordnern nicht mehr überprüft; die Einstellungsmöglichkeit ist in der ELO Administration Console entfernt worden. In Datenbank ist mit der Optid *1062* weiter die Konfigurationsmöglichkeit gegeben.

1.13.1.7 Maximale Fehleranzahl

Der Standardwert für die maximale Fehleranzahl, nach deren Erreichen der ELO Textreader stoppt, wurde auf 10.000 gesetzt.

1.13.2 OCR

- Bei Verbindung zur lokalen OCR wird nach OCR-Fehlern nicht mehr die OCR-Verbindung über den ELO Indexserver getestet (und wenn diese nicht hergestellt werden kann, die OCR-Verarbeitung nicht eingestellt).

- Nach abgebrochener OCR-Verbindung wird immer wieder neu gecheckt, ob die OCR nun zur Verfügung steht. Die Konvertierung pausiert nur solange, wie keine OCR-Verbindung hergestellt werden kann.
- Der Dateiname im ELO Textreader wird auch von der OCR bei der Log-Ausgabe verwendet, damit Fehler besser nachverfolgt werden können.
- Weitere OCR-Meldungen werden erkannt: „Timeout" (Abbruch der Konvertierung), „Connection Refused" (Wiederholung der Konvertierung), „File not found" (Abbruch der Konvertierung) und „Abbyy Engine Error" (OCR-Verarbeitung wird gestoppt).
- Wenn die Abbyy Finereader Engine nicht geladen werden kann, wird in Intervallen von 10 Minuten erneut versucht, mit der OCR eine Verbindung herzustellen.
- Der maximal verfügbare Speicherplatz wurde nicht richtig abgefragt, sodass bei Speichergrößen von mehr als 2 GB evtl. Bilddateien nicht mehr konvertiert wurden.
- Die Zeit, die der ELO Textreader auf das Ergebnis der OCR wartet, wird nicht mehr einmalig konfiguriert (Standardwert war 8 Stunden, nun 15 Minuten), sondern berechnet aus der Anzahl der Seiten und dem Timeout pro Seite. Standard-Timeout-Wert pro Seite ist nun 30 Sekunden.

1.13.3 Änderungen bzgl. Konverter

1.13.3.1 PDF-Konverter

- Aus PDF-Dateien können Anlagen im TIF(F)-, GIF, JP(E)G, PNG, CSV, HTM(L), TXT, RTF, XLS, PDF oder XML-Format extrahiert werden.
- Die maximale Größe von Bildern wird bereits beim Extrahieren gecheckt; zu große Bilder werden nicht extrahiert.
- An der X-Achse gespiegelte Bilder in PDF-Dokumenten werden zurückgespiegelt extrahiert (bei Konvertierung mit PDFBox).
- Inline-Bilder können extrahiert werden (bei Konvertierung mit PDFBox)
- jb2-Bilder werden extrahiert (als Bilder vom Typ PNG).
- Würde beim Extrahieren von Bildern oder Attachments der minimale Speicherplatz unterschritten, wird mit der Konvertierung dieses PDFs gewartet, bis ausreichender Plattenplatz vorhanden ist.
- Wenn Fehler bei der Konvertierung durch die PDFBox auftreten, können die PDFs zur Konvertierung an die OCR geschickt werden (muss aktiviert werden, siehe Handbuch).
- Als Default ist die maximale Anzahl von extrahieren Bildern pro PDF-Datei hoch gesetzt worden von 2000 auf 20.000; einstellbar in der Datenbank mit dem Offset 19 auf die Optid für PDF-Dateien (z. B. `132019`, wenn die Optid für PDF `1320` ist).

1.13.3.2 Word

- Word-Dokumente vom Dateityp DOC, die im Format *Office Open XML* gespeichert sind, können konvertiert werden.

1.13.3.3 EML

- Die minimale Pixelanzahl von Grafiken wird auch bei aus EML-Dateien extrahierten Grafiken überprüft.
- Bei EML-Anhängen ohne oder mit unsinnigem Dateityp wird versucht, den Dateityp aus den Content-Informationen des EML-Elements zu ermitteln.

1.13.3.4 MSG

- Die minimale Pixelanzahl von Grafiken wird auch bei aus MSG-Dateien extrahierten Grafiken überprüft.
- Inhalt im HTML-Format wird in die Volltextdatei aufgenommen.
- Anhänge aus signierten MSG-Dateien werden extrahiert (auch Anhänge mit unsinniger Dateiendung, sowie Anhänge, deren Namen nicht festgestellt werden können).

1.13.3.5 Nicht unterstützte Textdateitypen

Es ist jetzt möglich, die Konvertierung von Text-Dateien mit einem anderen (vom ELO Textreader nicht unterstützten) Dateityp (z. B. SQL in der ELO Administration Console zu konfigurieren, sodass diese Dateien direkt in den TXT-Ordner exportiert und von dort aus als Volltextdateien wieder importiert werden.

1.13.3.6 Neue Konverter

7z- bzw. 7zip- und vsdx-Dateien können konvertiert werden.

1.14 OCR

- OCR Whitespace Character können durch Java Whitespace Character ersetzt werden.
- Der Temp-Ordner wird korrekt an AbbyyFinereader weitergeleitet.
- Der Garbage Collector kann über die Konfigurationsdatei aktiviert werden.
- Nullpointer Exceptions während des Zusammenstellens von temporären Resultaten nach Abbruch eines Workers werden vermieden.
- Keine Out-of-Memory-Fehler mehr beim Empfangen/Senden von Dateien.

1.15 Lizenz

Die Lizenz besteht nun aus einer Datei mit signiertem Inhalt. Wenn die Lizenz direkt in der Datenbank eingetragen werden soll, dann ist dafür der komplette Dateiinhalt in die Tabelle *eloamoptions*, Spalte *optblob* bei *optno=2* einzutragen.

1.16 Upgrade Index

Bei einem Update von ELO 11 auf ELO 12 ist kein Re-Index mehr notwendig.

Wird ein Update von ELO 10 auf ELO 12 gemacht, kann nicht mehr die Funktion *Upgrade Index* des ELO Server Setups verwendet werden. Die separate Installation, die hier gemacht wird, passt die Datenbank für ELO 12 an. Dadurch können manchen Funktionen, wie z. B. die Vertretungsregelungen, in dem produktiven ELO 10 System nicht mehr komplett verwendet werden.

Um den Re-Index trotzdem im Hintergrund durchführen zu können, muss die SQL-Datenbank auf einen zweiten Server kopiert werden. Danach kann im ELO Server Setup *Upgrade Index* ausgeführt werden. Die separate Installation muss auf die Kopie der SQL-Datenbank verweisen. Wichtig ist, dass diese zweite Installation die Dokumentenpfade des Produktivsystems kennen muss.

Am Ende kann dann das Produktivsystem auf ELO 12 gehoben werden und der Index der Elasticsearch kann von dem Re-Index-System in das Produktivsystem kopiert werden.

1.17 Neue Option in der Konfiguration der ELO iSearch

1.17.1 Das Problem nicht-lateinische Zeichen

Der Text-Analyseprozess in der ELO iSearch ist darauf ausgelegt, für die in der ELO iSearch unterstützten Sprachen mit Hilfe von sog. Analyzern und Filtern so zu bearbeiten, dass in der zugrundeliegenden Elasticsearch eine effektive und performante Suche möglich ist.

Während dieses Prozesses werden bei Textfeldern u. a. auch sämtliche nicht-alphanumerischen Zeichen im Sinne des westeuropäischen Zeichensatzes jeweils am Anfang und Ende jedes Wortes entfernt. Dies ist für den Analyseprozess unbedingt notwendig für westeuropäische Sprachen (z. B. Deutsch, Englisch).

Bei Kunden, deren Volltext-Dokumente auch Text in einer nicht-unterstützten Sprache mit nicht-lateinischem Alphabet vorliegt (z. B. Japanisch oder Russisch), führt das dazu, dass sämtliche Textbestandteile, die in dieser Sprache im Volltext vorhanden sind, nicht in den Suchindex aufgenommen werden und damit auch nicht im Client von einer Suche gefunden werden können.

Es ist im Augenblick nicht geplant, die Liste der unterstützten Sprachen in der ELO iSearch zu erweitern. Um dennoch für betroffene Kunden einen Kompromiss zu erreichen, ist folgender Ansatz in ELO12 umgesetzt worden.

1.17.2 Analyzer-Option zum Abschalten der Entfernung von Nicht-ASCII-Zeichen

Es ist nun möglich, die Entfernung von nicht-ASCII-Zeichen durch Setzen einer Option in der ELO iSearch-Konfiguration zu verhindern. Damit werden diese Zeichen in Wörtern mit indiziert und der z. B. japanische Volltext kann nach vorkommenden Wörtern durchsucht werden.

Dabei sind folgende Rahmenbedingungen zu beachten:

- Die Option **wirkt nur auf den Volltext**. Kurzbezeichnung und Indexfelder sowie Feed sind nicht betroffen.
- Die Option wirkt nur dann, wenn durch die eingebaute Spracherkennung für ein Dokument keine der in der ELO iSearch unterstützten Sprachen erkannt wurde und somit der sog. Fallback-Analyzer verwendet wird. Bei Mischtexten, die durch genügend häufiges Vorkommen von z. B. deutschen Wörtern als „deutsch“ erkannt wurden, findet das Entfernen der nicht-ASCII-Zeichen also weiterhin statt.
- Die Option beeinflusst durch das Nicht-Entfernen von Sonderzeichen auch die Analyse und damit die Indizierung von lateinischen Textbestandteilen, falls sie vom Fallback-Analyzer behandelt werden. Zum Beispiel werden die Klammern in „(zur Ansicht)“ nicht entfernt und das Dokument wird u. U. nicht gefunden, wenn nach „Ansicht“ gesucht wird. Es kann vorkommen, dass z. B. auch deutsche Texte vom Fallback-Analyzer behandelt werden, wenn die Spracherkennung auf Grund von zu wenig deutschem Wortschatz im ersten Textteil (z. B. hohem Ziffernanteil) den Test nicht eindeutig klassifizieren kann.
- Die Option steht standardmäßig auf „Aus“.
- Die Option wird erst nach einem Reindex des Archivs wirksam.

Aus obigen Hinweisen ist zu entnehmen, dass es ratsam ist, diese Option nur in Spezialfällen einzuschalten, wenn man sich der Auswirkungen bewusst ist und diese in Kauf nehmen kann.

1.17.3 Die Option auf der ELO iSearch-Konfigurationsseite

Die neue Option *Do not cut Ascii-Chars* ist auf der Konfigurationsseite der ELO iSearch unter dem Tab *Settings* im Abschnitt *Analyzer Settings* zu finden.

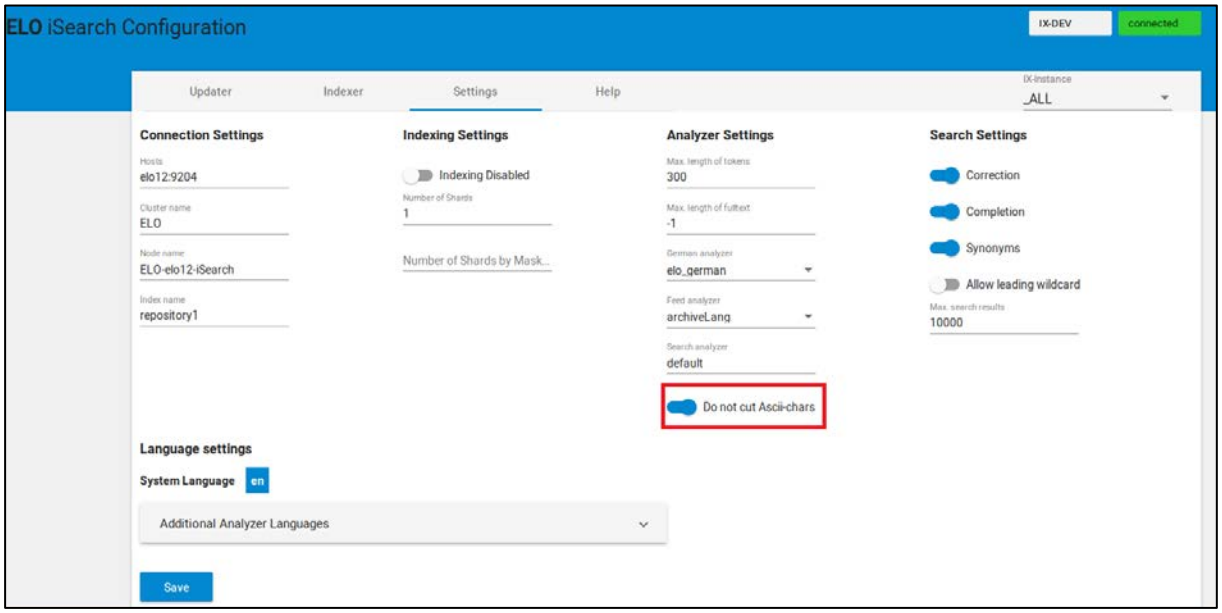


Abb. 1: Option ‚Do not cut Ascii-chars'

2 Neuerungen in ELO 11

Folgende Neuerungen wurden mit ELO 11 eingeführt.

2.1 Änderung im Deployment

Der ELO Indexserver besteht nicht mehr ausschließlich aus der Datei *ix.war*. Weitere JAR-Dateien, die wir Plugins genannt haben, müssen installiert werden (siehe Kapitel 2.6).

Beispielsweise befindet sich die Elasticsearch-Anbindung in einem solchen Plugin. Die Aufteilung ermöglicht uns, in Zukunft mit einer älteren Version des ELO Indexservers auf einen neueren Elasticsearch-Index zuzugreifen. Beim Update von 11 auf 12 wird das möglicherweise erforderlich sein.

Für ein manuelles Update des ELO Indexservers ist wie folgt vorzugehen:

- Stoppen Sie ELO Server 1.
- Kopieren Sie *Indexserver.zip/ix.war* nach *<instdir>/prog/webapps/ix.war*.
- Löschen Sie *<instdir>/servers/ELO-…1/webapps/ix**.
- Neu: Kopieren Sie *Indexserver.zip/ix-plugins/** nach *<instdir>/prog/webapps/ix-plugins/**.
- Starten Sie ELO Server 1.
- Stellen Sie sicher, dass die ELO Indexserver-Option *osgiPluginDirectories* auf der Optionenseite des ELO Indexservers gesetzt ist. Der Wert ist *<instdir>/prog/webapps/ix-plugins*. Nach einer Änderung muss der ELO Indexserver neu gestartet werden.

2.2 Änderungen in der Berechtigungsprüfung

2.2.1 Schlüsselberechtigungen obsolet

Schlüssel werden nicht mehr geprüft.

2.2.2 Archivschlüssel

RID12176: Um zu verhindern, dass sich bestimmte Benutzer anmelden können, schränkt man den Zugriff auf die Mitglieder einer Gruppe ein.

Hierzu wurde die Funktion *lockArchive* geändert: Sie nimmt nun eine Gruppen-ID (-name, -GUID) entgegen. Nur Mitglieder dieser Gruppe können sich im Archiv anmelden.

2.2.3 Recht ‚Dokumentdateien sichtbar' entfernt

RID12239: Das Benutzerrecht *Dokumentdateien sichtbar* wird nicht mehr ausgewertet. Auch wenn ein Benutzer das Recht nicht hat, kann er somit Dokumente sehen.

2.2.4 Löschen eines Untereintrags

RID12256: Zum Löschen eines Untereintrags – sowohl für Originale als auch für logische Referenzen – wird das ACL-Recht *Listen* benötigt. Diese Prüfung wurde bisher nur für logische Referenzen durchgeführt.

2.2.5 Speichern von Attachments

RID12247: Zum Speichern von Attachments wird nur noch das Benutzerrecht *Dokumente bearbeiten* benötigt. Das Recht *Dokumentdateien sichtbar* entfällt (s. Kapitel 2.2.3).

2.2.6 Erstellen logischer Referenzen

RID12244: Zum Erstellen einer logischen Referenz wird kein bestimmtes Benutzerrecht benötigt.

2.2.7 Speichern neuer Dokumentversionen

RID12239: Für das Speichern neuer Dokumentversionen wird nur noch das Benutzerrecht *Dokumente bearbeiten* geprüft. Das Recht *Dokumentdateien sichtbar* entfällt (s. Kapitel 2.2.3).

2.2.8 Berechtigungen aktiver Workflows

RID10750: Aktive Workflows können durch Angabe einer ACL berechtigt werden. Sie bestimmt, wer den Workflow sehen, ändern, löschen und beenden darf. Ist keine ACL angegeben, erfolgt die Prüfung wie zuvor.

2.3 Verschlagwortungsmasken

RID12277: Die Verwendung einer Verschlagwortungsmaske kann durch zwei neue Optionen eingeschränkt sein. In der ELO Administration Console finden Sie die neuen Einstellungen unter *Vorbestimmung*.

Zum einen kann für eine Verschlagwortungsmaske in *DocMask.maskIdsForChildEntries* festgelegt werden, welche Verschlagwortungsmasken für Untereinträge verwendet werden dürfen.

Zum anderen kann mit *DocMaskDetails.enabledByParent* eingestellt werden, dass eine Verschlagwortungsmaske nur dann benutzt werden darf, wenn der übergeordnete Eintrag dies explizit erlaubt.

Mit dieser Änderung ergibt sich ein neues Verhalten der Funktionen *createSord*, *createDoc*, *checkoutSord*, *checkoutDoc*. Im zurückgegebenen Objekt *EditInfo* sind nicht mehr alle Verschlagwortungsmasken im Array-Element *maskNames* enthalten, sondern nur noch diejenigen, die nach den oben genannten Regeln erlaubt sind.



Information: Im Programmierhandbuch für den ELO Indexserver wird in Beispiel 70 beschrieben, wie Sie eine Liste aller Ablagemasken erhalten. Damit das Beispiel auch in ELO 11 funktioniert, ist der Funktion *createSord* ein Leerstring im Parameter *parentId* zu übergeben.

```
EditInfo ed = conn.Ix.createSord("", "", editZ);
```

Dieser Aufruf liefert auch in ELO 9 und ELO 10 alle Verschlagwortungsmasken.

2.4 Workflow-Cache

Aktive Workflows werden im Hauptspeicher des ELO Indexservers gehalten, um die Aufgabenlisten und Workflow-Übersichten schneller ausliefern zu können. Änderungen an Workflows werden dagegen weiterhin direkt in die Datenbank geschrieben.

2.4.1 Performance

Aus Performance-Tests wurde ein ungefährer Speicherbedarf von 1GB RAM für 500.000 Knoten oder 5000 Workflows à 100 Knoten ermittelt. Diese Zahlen sollten nur als Anhaltspunkte für die Dimensionierung des Tomcat-Heap-Speichers aufgefasst werden. In der Praxis können sich abhängig von Knotenverbindungen und Knoteneigenschaften andere Werte ergeben.

2.4.2 Konfiguration

Wenn mehrere ELO Indexserver für ein Archiv betrieben werden, ist darauf zu achten, dass die Option *privateUrlBase* korrekt gesetzt ist. Sie gibt die URL an, unter der sich die ELO Indexserver gegenseitig erreichen können. Dadurch können sie den Workflow-Cache untereinander abgleichen.

Für den Cache-Abgleich sendet ein ELO Indexserver allen anderen ELO Indexservern desselben Archivs einen Aktualisierungshinweis per http. Damit diese Kommunikation nicht zu einer wahrnehmbaren Verzögerung der Workflow-Verarbeitung führt, sollten nicht mehr als drei ELO Indexserver für ein Archiv betrieben werden.

2.5 Skripting

2.5.1 Sealed-Objekte

Die Funktionen *checkoutUsers* und *checkoutSubs* geben Objekte aus dem internen Cache des ELO Indexservers zurück. Damit sie nicht versehentlich geändert werden und damit der Cache in einen inkonsistenten Zustand kommt, werden sie „versiegelt“ (sealed). Ändern Sie eine Eigenschaft eines versiegelten Objekts, erhalten Sie die Exception … *sealed object cannot be modified* ….

Das Objekt muss zuerst kopiert werden, bevor es geändert werden kann. Zum Kopieren steht der Copy-Konstruktor zur Verfügung.

```
var userInfo = ixConnect.ix().checkoutUsers(…)[0];
// userInfo.name = „new name“; throws „sealed object cannot be modified“
userInfo = new UserInfo(userInfo); // copy
userInfo.name = „new name“; // OK
```

2.6 ELO Indexserver-Plugins

Die Funktionalität des ELO Indexservers kann neben dem serverseitigen Skripting durch Plugins erweitert werden. Es handelt sich dabei um OSGi-Bundles, die vom OSGi-Framework Felix (Apache Group) geladen werden, das im ELO Indexserver integriert ist.

Die Plugins liegen als JAR-Dateien standardmäßig im Ordner *\ELO11\prog\webapps\ix-plugins*. In den Optionen des ELOix ist dieses Verzeichnis in der Einstellung *osgiPluginDirectories* vorgegeben. Beim Start werden alle JARs in den *osgiPluginDirectories* und den Unterverzeichnissen (rekursiv) gesammelt.

Das im Download bereitgestellte ELO Indexserver-Paket enthält bereits einige Plugins, die standardmäßig im Unterordner *private* enthalten sind.

Zur Entwicklung eigener Plugins gibt es die separate Dokumentation *IX Plugins Dokumentation* im Download-Bereich.

2.7 LDAP-Anmeldung über ELOix

Alternativ zur LDAP-Anbindung im ELO Access Manager bietet der ELO Indexserver eine Authentifizierung über LDAP an. Hierbei gibt es erweiterte Konfigurationseinstellungen und die Möglichkeit, durch Skripting Einfluss auf den Anmeldeprozess zu nehmen.

Die Funktionalität zur LDAP-Anbindung ist in einem Plugin enthalten, das vom ELO Serversetup ins Verzeichnis *<instdir>/prog/webapps/ix-plugins/private* installiert wird. Die Konfiguration des Plugins wird jedoch noch nicht vom Setup unterstützt.

2.7.1 Funktionalität

Es werden alle wesentlichen Funktionen der LDAP-Anbindung des ELOam angeboten. Die wichtigsten davon werden im Folgenden mit den zugehörigen Optionen in der Konfigurationsdatei aufgeführt. Eine Beispielkonfiguration finden Sie im Abschnitt 2.7.2.

2.7.1.1 Mehrere Domänen

Die LDAP-Anmeldung kann an verschiedenen Domänen erfolgen.

Option *domainPrefix*: Wenn im Anmeldename ein Domänenpräfix enthalten ist (z. B. *ELO\Müller*), wird das Login nur gegen die Domäne mit diesem Domänenpräfix versucht. Ohne Domänenpräfix im Anmeldename werden alle konfigurierten Domänen in Betracht gezogen.

2.7.1.2 Einstellungen je Domäne

Es kann pro Domäne u. a. angegeben werden:

- Optionen *personSearchDNList*, *groupSearchDNList*: welche Pfade im LDAP nach Benutzern und Gruppen durchsucht werden
- Option *attributesToRead*: welche Benutzer-Attribute aus dem LDAP in ELO übernommen werden
- Optionen *servers*, *ldapUrl*, *priority*: mehrere Server-URLs zur Lastverteilung und Ausfallsicherheit
- Optionen *defaultUserName*, *defaultUserPasswordEncr*: standardmäßig verwendetes LDAP-Konto zum Suchen von Benutzern und Gruppen

2.7.1.3 Neue Benutzer erstellen

Option: *createNewUser*

Nach erfolgreicher Anmeldung wird geprüft, ob der Benutzer bereits in ELO angelegt ist. Ist dies nicht der Fall, kann der Benutzer in ELO erstellt werden. Sollen neue Benutzer nicht in ELO übernommen werden, führt die Anmeldung eines neuen Benutzers zu einem Fehler.

2.7.1.4 Gruppenabgleich

Option: *assignGroups*

Bei der Anmeldung eines Benutzers kann eine im LDAP zugewiesene Gruppe auch in ELO gesetzt werden. Voraussetzung dafür ist, dass es die Gruppe in ELO mit demselben Namen bereits gibt.

2.7.1.5 Interne Anmeldung

Option: *notResponsibleForUserOrGroupIds*

Für bestimmte ELO Benutzer, wie bspw. *Administrator* und *ELO Service*, kann angegeben werden, dass die Anmeldung nicht über LDAP erfolgen soll, sondern über die ELO Benutzerdatenbank.

In Verbindung mit dem Standardkonto für LDAP-Zugriffe ermöglicht diese Option runAs-Anmeldungen am ELOix, bei denen das technische Konto nicht notwendigerweise ein LDAP-Konto sein muss.

Die Option kann auch Gruppen-IDs/-GUIDs enthalten. Benutzer dieser Gruppen werden damit über die interne ELO Benutzerdatenbank authentifiziert.

2.7.2 Konfiguration

Das LDAP-Plugin sucht seine Konfigurationsdatei an den folgenden Orten:

- Im Dateisystem: *<instdir>/config/ix-archivname/ldap.json* oder *ldap.cfg*
- Im Archiv: */Administration/Indexserver Scripting Base/_ALL/ldap* oder *…/Instanzname/ldap* (ohne Dateiendung)

Bei der Datei *ldap.json* handelt es sich um eine Textdatei im JSON-Format (s. Beispiel unten).

Um den Umstieg von der LDAP-Anbindung des ELOam zu vereinfachen, wird mit einer *ldap.cfg* auch dessen Format akzeptiert. Nach dem Auslesen wird sie in eine *ldap.json* konvertiert.

Wenn keine Konfiguration gefunden wird, erstellt das Plugin automatisch eine Konfigurationsvorlage in der Datei *ldap-auto.json* im o. g. Verzeichnis. Die Einstellungen darin werden aus der Systemumgebung des Servers ermittelt und sind als Vorschlag anzusehen. Sie werden vom Plugin ignoriert. Um die Einstellungen zu aktivieren, muss die Datei in *ldap.json* umbenannt werden.

2.7.2.1 Beispiel für eine *ldap.json*

```
{
  "createNewUser": true,
  "assignGroups": true,
  "notResponsibleForUserOrGroupIds": [
    "0",
    "(E10E1000-E100-E100-E100-E10E10E10E42)"
  ],
  "domains": [
    {
      "name": "ELO.LOCAL",
      "domainPrefix": "ELO\\",
      "defaultUserName": "ELO Service",
      "defaultUserPasswordEncr": "password",
      "servers": [
        {
          "ldapUrl": "ldap://srvp03.elo.local:389",
          "priority": 0.3333333333333333,
        },
        {
          "ldapUrl": "ldap://srvp04vm.elo.local:389",
          "priority": 0.3333333333333333,
        },
        {
          "ldapUrl": "ldap://srvp05vm.elo.local:389",
          "priority": 0.3333333333333333,
        }
      ],
      "personSearchDNList": [
        "DC=ELO,DC=LOCAL"
      ],
      "groupSearchDNList": [],
    }
  ]
}
```



Information: Aus technischen Gründen erscheinen in der Konfigurationsvorlage *ldap-auto.json* Attribute namens *changedMembers*. Sie können ignoriert werden.

2.7.3 Skripting

Neben der statischen Konfiguration kann auf einzelne Schritte der Anmeldung per Skripting Einfluss genommen werden. Hierzu wird ein Objekt der Klasse *de.elo.ix.client.ldap.LdapUserDirectoryHandler* in den Prozess eingehängt. Beispiel:

```
var interfaceName = "de.elo.ix.client.plugin.UserDirectory";
var UserInfoC = Packages.de.elo.ix.client.UserInfoC;

function init() {

 // Get LDAP service
 var ldapService = ixPlugins.getService(interfaceName);

 // Create a factory object derived from Ldap…HandlerFactory
 var factory = new JavaAdapter(
  Packages.de.elo.ix.client.ldap.LdapUserDirectoryHandlerFactory, {

   // Called from LDAP service to get a callback handler
   createHandler : function(defaultHandler,
        config, userName, password, runAsName) {

     // Create a callback handler
     var adapter = new JavaAdapter(
     Packages.de.elo.ix.client.ldap.LdapUserDirectoryHandlerWrapper, {

      // Mandatory: return the default handler supplied in the constructor
      getInnerHandler : function() {
       return defaultHandler;
      },

      // Optional: e.g. assign additional properties after authenticated
      assignUserName : function(ui, domain, userName, attributes) {
        defaultHandler.assignUserName(ui, domain, userName, attributes);
        ui.userProps[UserInfoC.PROP_ACTION] = "hello ldap";
      }

    });

    return adapter;
   }
 });

 ldapService.setHandlerFactory(factory);
}

init();
```

2.8 ELO iSearch

2.8.1 Update der Elasticsearch

Die Elasticsearch wurde auf Version 5.6.3 aktualisiert.

2.8.2 Verschlüsselte Kommunikation mittels SSL

Seit ELO 10.2 wird das Plugin *Searchguard* verwendet.

Über das Setup werden die benötigten Zertifikate und Passwörter erstellt.

Weitere Informationen zum Plugin: https://github.com/floragunncom/search-guard

2.8.3 Feed-Inhalte in der ELO iSearch

Die Inhalte des Feeds sind mit ELO 11 über die ELO iSearch durchsuchbar.

2.8.4 Auftrennung von Sord-Informationen und Volltextdaten

Auftrennung der SORD-Informationen und Volltexte. Diese beiden Teile eines SORDs werden ab ELO 11 getrennt voneinander gespeichert.

2.8.4.1 Hintergrund

Bisher musste bei jeder Änderung an einem SORD-Objekt das komplette Dokument inklusive Volltext erneut in der Elasticsearch gespeichert werden. Dies ist, vor allem bei längeren Dokumenten, sehr zeitintensiv.

Durch die Trennung werden die Volltexte nur erneut an die Elasticsearch geschickt, wenn sich der Volltext geändert hat.

2.8.4.2 Auswirkungen

Durch die Aufteilung von Metainformationen und Volltext ändert sich das Suchverhalten:

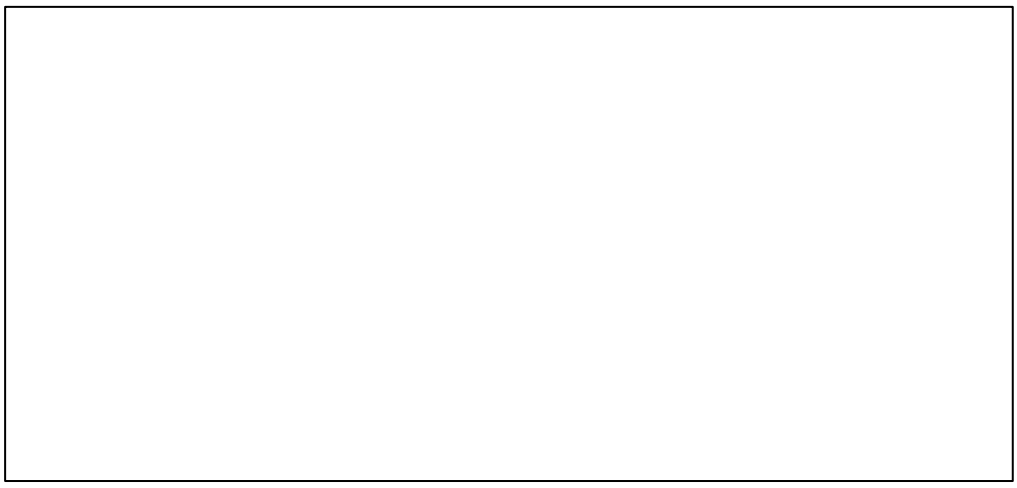


Abb. 2: Suchverhalten

Wird z. B. nach `Haus AND Baum` in allen Bereichen gesucht, wird ein Dokument, bei dem `Haus` im Volltext und `Baum` in einem Indexfeld steht, nicht mehr gefunden.

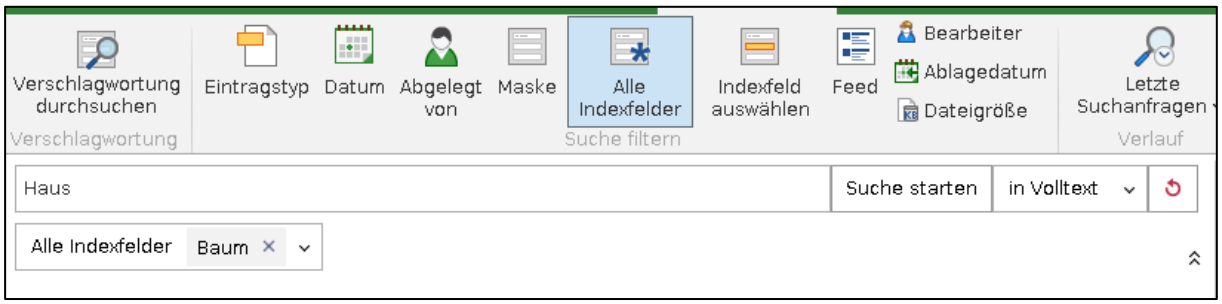


Abb. 3: Filter 'Alle Indexfelder'

Um dieses Dokument trotzdem zu finden, steht dem Benutzer jetzt die Möglichkeit bereit, die Suche genauer zu formulieren und nach `Haus` im Volltext und `Baum` in allen Indexfeldern zu suchen.

2.8.5 Aufteilung in unterschiedliche Indizes

Bisher wurde der gesamte Inhalt in einem Elasticsearch-Index gespeichert. Dies hat den Nachteil, dass die Dokumente, die indexiert werden, sehr lückenhaft („sparse“) sind. Das liegt daran, dass je Maske unterschiedliche Indexfelder benutzt werden, allerdings alle gemeinsam indexiert werden. Dies hat negative Auswirkungen auf die Suchleistung.

Ab ELO 11 wird ein Index pro Maske angelegt. Dadurch sind die Dokumente in der Elasticsearch vollständiger. Eine Suche, die einen Filter auf eine Maske beinhaltet, durchsucht dadurch auch nur noch einen Index. Weiterhin können alle Masken über einen sogenannten Alias-Namen angesprochen werden.

Folgende Abfrage über den Browser zeigt die erzeugten Indizes:

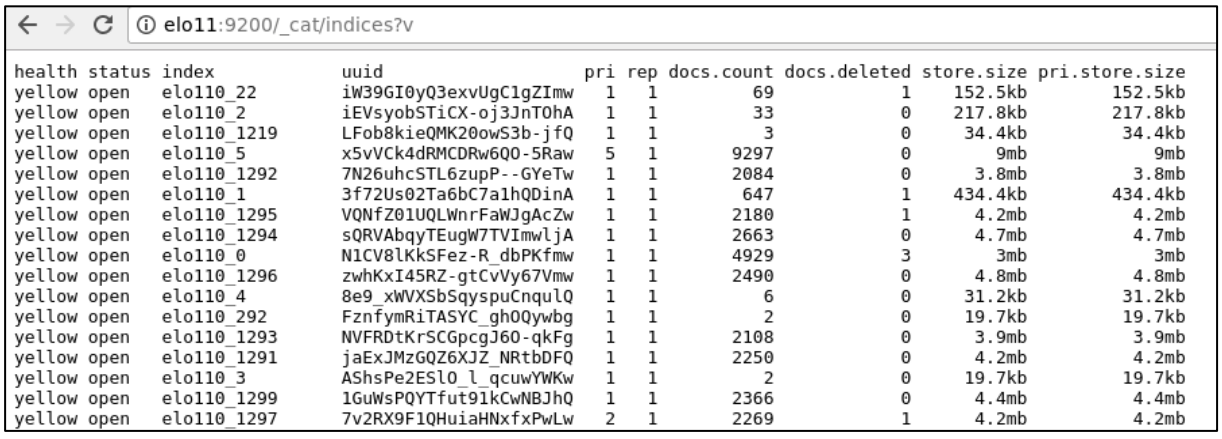


Abb. 4: Anzeige der Indizes

2.8.5.1 Festlegung der Anzahl an Shards pro Index

Mit ELO 11 wird standardmäßig ein Shard (https://www.elastic.co/guide/en/elasticsearch/reference/current/_basic_concepts.html) pro Index angelegt (bisher: 5). Diese Anzahl kann für alle Indizes über die Seite *iSearch Config* oder den Eintrag *esearch.settings.shards* in der Tabelle *eloixopt* geändert werden. Außerdem kann für einzelne Masken die Anzahl an Shards angepasst werden, z. B. für Masken mit sehr vielen Dokumenten. Im obigen Beispiel wurde für die Maske mit der ID 5 die Anzahl auf 5 gesetzt, für die Maske mit der ID 1297 auf 2.

Dies wird über die Seite *iSearch Config* oder den Eintrag *esearch.settings.shardsByMask* in der Tabelle *eloixopt* festgelegt. Folgende Syntax muss verwendet werden: *<Masken ID>:<Anzahl Shards>;<Masken ID>:<Anzahl Shards>*, für obiges Beispiel also *5:5;1297:2*.

2.8.6 Einführung einer neuen Such-API

Bis ELO 10.2 wird die Suchanfrage zur Suche in der Elasticsearch vom Client zusammengestellt und vom ELOix nur an die Elasticsearch (mit geringen Änderungen) weitergeleitet.

Nachteil dieser Lösung ist, dass Änderungen am Aufbau der Elasticsearch von jedem Client implementiert und korrekt umgesetzt werden müssen.

Ab ELO 11 wird eine neue API vom ELOix zur Verfügung gestellt, sodass Suchanfragen unabhängig von der zugrundeliegenden Suchmaschine aufgebaut werden.

Die alte Suchschnittstelle wird weiterhin unterstützt, jedoch können die Ergebnisse von denen in ELO 10 abweichen, z. B. durch Änderungen wie in Kapitel *2.8.4* beschrieben.

2.8.7 Überarbeitung der Kommunikation bei mehreren ELOix Instanzen

Bisher wurde über ein Sord-Objekt im Archiv festgelegt, ob aktuell ein Update-Prozess aktiv ist. Dies führte häufig zu Problemen, da das Objekt nicht korrekt entsperrt wurde.

Die Kommunikation zwischen mehreren ELOix Instanzen beim Update- und Reindex-Prozess wird mit ELO 11 auf http umgestellt. Möchte ein ELOix einen Prozess starten, wird mittels http-Request abgefragt, ob gerade ein anderer ELOix aktiv ist. Ein Prozess auf einem ELOix wird nur gestartet, wenn gerade kein anderer ELOix aktiv ist. Der Prozess wird auf dem ELOix gestartet, dessen Name am weitesten vorne im Alphabet steht. Ausnahme: Wird ein Prozess aktiv von einem Administrator über die Config-Seite gestartet, wird der Prozess auch dann gestartet, wenn der Name des ELOix nicht am weitesten vorne im Alphabet steht.

2.8.8 Überarbeitung der Rechteüberprüfung (seit ELO 10.2)

Die Überprüfung der Rechte von SORD-Objekten über die Elasticsearch wurde zu ELO 10.2 überarbeitet, sodass sie deutlich beschleunigt wurde.

2.8.9 Neue Config-Seite

Alle Informationen und Einstellungen zur ELO iSearch sind auf einer Webseite untergebracht. Die Kommunikation mit dem ELOix findet über die ELO Indexserver-API statt.

2.8.10 Backups

Sobald das Erzeugen des Index ein zeitaufwendiger Prozess ist, ist es sinnvoll, ein Backup des bestehenden Elasticsearch-Index anzulegen. Dies ist stark abhängig von der Größe des Archivs und der Datenstruktur (z. B. davon, ob viele Volltextdateien vorhanden sind).

Zum Erstellen eines Backups gibt es zwei Möglichkeiten. Bei beiden Lösungen ist zu beachten, dass die Indizes zwischen unterschiedlichen ELOix Versionen nicht zwingend kompatibel sind.

2.8.10.1 Bestehenden Index kopieren

Der bestehende Index kann aus dem Dateisystem kopiert werden und zu einem späteren Zeitpunkt wieder eingespielt werden. Dazu wird der komplette Ordner unter *<Installationsverzeichnis>/data/<…><Servername>/index* kopiert. Bevor der Index kopiert wird, sollte der iSearch-Dienst beendet werden.

2.8.10.2 Snapshot mittels Elasticsearch erstellen

Elasticsearch selbst bietet die Möglichkeit, sogenannte Snapshots der bestehenden Indizes anzulegen. Werden regelmäßig Snapshots erzeugt, wird nicht jedes Mal eine komplette Kopie erzeug. Stattdessen werden nur die geänderten Dateien gespeichert; so wird der benötigte Speicherplatz reduziert.

Die Anleitung von Elasticsearch ist hier zu finden: https://www.elastic.co/guide/en/elasticsearch/reference/5.4/modules-snapshots.html

Beachten Sie: Um einen Snapshot wiederherstellen zu können, muss der Konfigurationsdatei der Elasticsearch (*<Installationsverzeichnis>/config/elastic/<Servername>/elasticsearch.yml*) folgende Zeile hinzugefügt werden:

```
searchguard.enable_snapshot_restore_privilege: true
```

2.9 WebDAV

2.9.1 Read-only

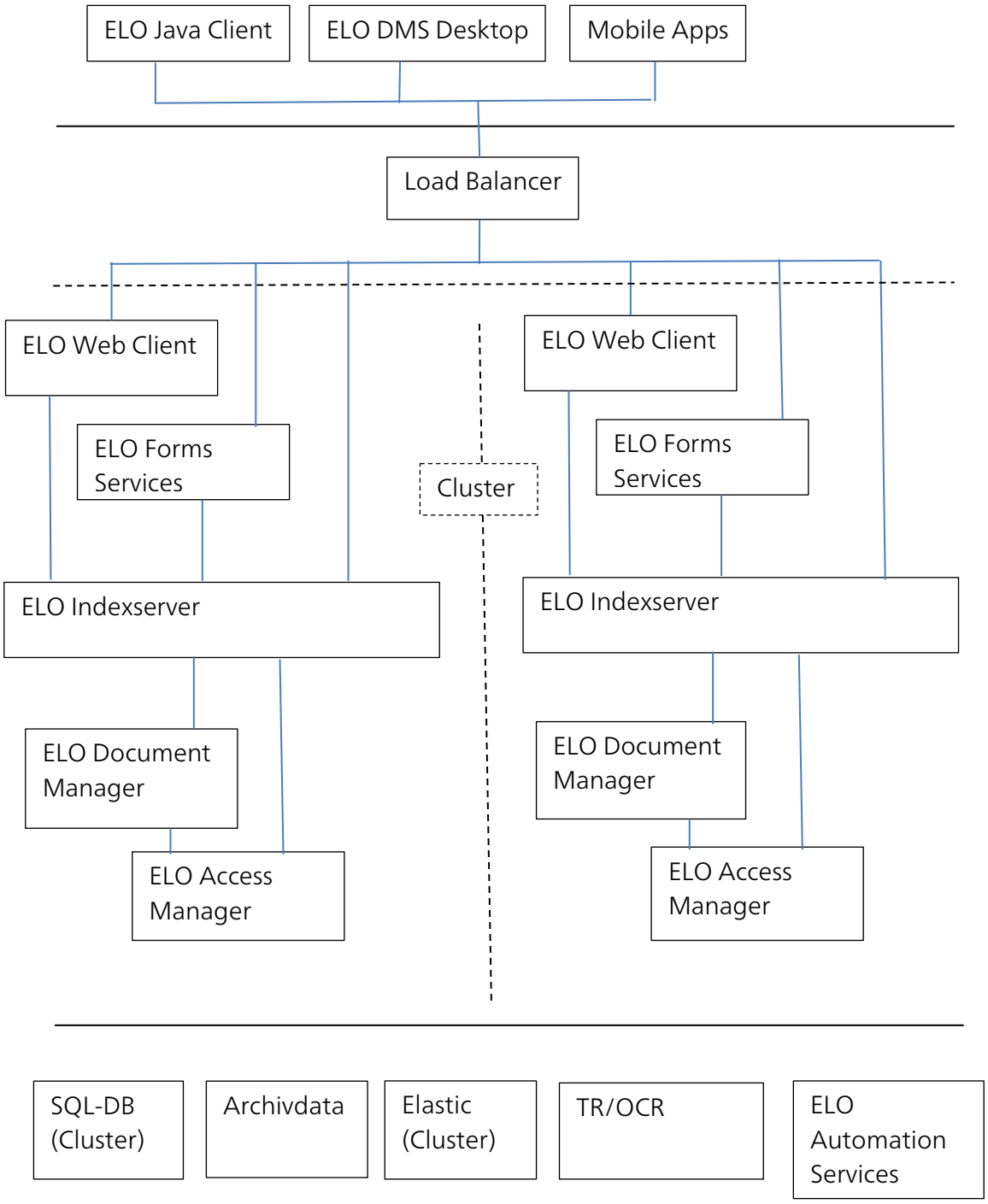
Mit der Einstellung

<entry key="webdav.readonly">true</entry>

In der *config.xml* des ELOix kann der WebDAV-Zugriff für alle Benutzer auf Lesezugriff eingeschränkt werden.

2.10 Cluster-Betrieb

Neben ELO Indexserver und ELO Access Manager kann nun auch der ELO Document Manager mit mehreren Instanzen in einem Load-Balancing-Cluster betrieben werden. Damit sind alle Programme clusterfähig, die vom Setup standardmäßig auf dem ersten Tomcat installiert werden: ELO Access Manager, ELO Document Manager, ELO Indexserver, ELO Forms Services, ELO Web Client.



2.11 Ebenen-Typ-Definitionen

Die Definition der Ebenen- und Dokumenttypen sind nun in der Archiv-Datenbank gespeichert und nicht wie in vorigen Versionen in der Datei *elodoc.ini* in einem Konfigurationsverzeichnis des ELO Access Managers.

Die Icons liegen im Ordner *Administration/Sord Type Icons*.

2.12 Beispiel für Formatierungen

Quellcode

```
public class MyServiceForScripting {

  public int add(int a, int b) {
    return a + b;
  }
}
```

Information: Im Skripting muss die Klasse des Services nicht bekannt sein, weil die JavaScript-Engine über Reflection auf alle Elemente und Funktionen des Services zugreift.

2.13 Änderungen im Textreader

Die Informationen zu den Änderungen beim ELO Textreader sind für ELO 11 und ELO 12 nahezu identisch. Nachfolgend finden Sie die Informationen, die für ELO 11 gültig sind.

2.13.1 Allgemeine Änderungen

2.13.1.1 Java 11

Der ELOtr ist auf Java 11 umgestellt worden.

2.13.1.2 Verbindung zum ELO Indexserver

Auch nach einer Fehlermeldung des ELO Indexserver versucht der ELO Textreader weiterhin sich anzumelden.

2.13.1.3 Statusseite

Exporter, Importer und die Konverter-Threads können auf der Statusseite ein- und ausgeschaltet werden (und nicht nur in der config.xml).

2.13.1.4 Speicherplatzüberprüfung

- Windows Server 2016 wird erkannt und entsprechende Standardwerte zur Berechnung des minimalen Plattenplatzes in einer Systempartition gesetzt. Für alle Windows Server-Betriebssysteme ab 2016 wird 32 GB als minimaler Plattenplatz der ELO Textreader-Verzeichnisse in einer Systempartition angenommen.
- Die Konverter stoppen nur, wenn der Importer nicht mehr arbeitet, und nicht, wenn kein Speicherplatz für konvertierte Dateien mehr vorhanden ist.
- Die Konverter stoppen aber nicht, wenn der Importer in der config.xml oder auf der Statusseite ausgeschaltet worden ist.
- Der Speicherplatz für Textdateien wird nicht mehr überprüft (wird nur überprüft beim Exportieren aus dem Archiv).
- Der PDF-Konverter verwirft die Konvertierung eines PDF-Dokuments, wenn durch die Extraktion von Bildern oder Attachments der minimale Plattenplatz überschritten würde. Die Konvertierung wird erneut versucht, wenn ausreichender Plattenplatz vorhanden ist.
- Nach Speicherplatzüberschreitungen wartet der Textreader nicht mehr 10 Minuten, sondern nur eine Minute.
- Als Default wird die maximale Anzahl der Dateien in Import- oder Exportordnern nicht mehr überprüft; die Einstellungsmöglichkeit ist in der ELO Administration Console entfernt worden. In Datenbank ist mit der Optid *1062* weiter die Konfigurationsmöglichkeit gegeben.

2.13.1.5 Maximale Fehleranzahl

Der Standardwert für die maximale Fehleranzahl, nach deren Erreichen der ELO Textreader stoppt, wurde auf 10.000 gesetzt.

2.13.2 OCR

- Bei Verbindung zur lokalen OCR wird nach OCR-Fehlern nicht mehr die OCR-Verbindung über den ELO Indexserver getestet (und wenn diese nicht hergestellt werden kann, die OCR-Verarbeitung nicht eingestellt).
- Nach abgebrochener OCR-Verbindung wird immer wieder neu gecheckt, ob die OCR nun zur Verfügung steht. Die Konvertierung pausiert nur solange, wie keine OCR-Verbindung hergestellt werden kann.
- Der Dateiname im ELO Textreader wird auch von der OCR bei der Log-Ausgabe verwendet, damit Fehler besser nachverfolgt werden können.
- Weitere OCR-Meldungen werden erkannt: „Timeout“ (Abbruch der Konvertierung), „Connection Refused“ (Wiederholung der Konvertierung), „File not found“ (Abbruch der Konvertierung) und „Abbyy Engine Error“ (OCR-Verarbeitung wird gestoppt).

- Wenn die Abbyy Finereader Engine nicht geladen werden kann, wird in Intervallen von 10 Minuten erneut versucht, mit der OCR eine Verbindung herzustellen.
- Der maximal verfügbare Speicherplatz wurde nicht richtig abgefragt, sodass bei Speichergrößen von mehr als 2 GB evtl. Bilddateien nicht mehr konvertiert wurden.
- Die Zeit, die der ELO Textreader auf das Ergebnis der OCR wartet, wird nicht mehr einmalig konfiguriert (Standardwert war 8 Stunden, nun 15 Minuten), sondern berechnet aus der Anzahl der Seiten und dem Timeout pro Seite. Standard-Timeout-Wert pro Seite ist nun 30 Sekunden.

2.13.3 Änderungen bzgl. Konverter

2.13.3.1 PDF-Konverter

- Aus PDF-Dateien können Anlagen im TIF(F)-, GIF, JP(E)G, PNG, CSV, HTM(L), TXT, RTF, XLS, PDF oder XML-Format extrahiert werden.
- Die maximale Größe von Bildern wird bereits beim Extrahieren gecheckt; zu große Bilder werden nicht extrahiert.
- An der X-Achse gespiegelte Bilder in PDF-Dokumenten werden zurückgespiegelt extrahiert (bei Konvertierung mit PDFBox).
- Inline-Bilder können extrahiert werden (bei Konvertierung mit PDFBox)
- jb2-Bilder werden extrahiert (als Bilder vom Typ PNG).
- Würde beim Extrahieren von Bildern oder Attachments der minimale Speicherplatz unterschritten, wird mit der Konvertierung dieses PDFs gewartet, bis ausreichender Plattenplatz vorhanden ist.
- Wenn Fehler bei der Konvertierung durch die PDFBox auftreten, können die PDFs zur Konvertierung an die OCR geschickt werden (muss aktiviert werden, siehe Handbuch).
- Als Default ist die maximale Anzahl von extrahieren Bildern pro PDF-Datei hoch gesetzt worden von 2000 auf 20.000; einstellbar in der Datenbank mit dem Offset 19 auf die Optid für PDF-Dateien (z. B. `132019`, wenn die Optid für PDF `1320` ist).

2.13.3.2 Word

- Word-Dokumente vom Dateityp DOC, die im Format *Office Open XML* gespeichert sind, können konvertiert werden.

2.13.3.3 EML

- Die minimale Pixelanzahl von Grafiken wird auch bei aus EML-Dateien extrahierten Grafiken überprüft.
- Bei EML-Anhängen ohne oder mit unsinnigem Dateityp wird versucht, den Dateityp aus den Content-Informationen des EML-Elements zu ermitteln.

2.13.3.4 MSG

- Die minimale Pixelanzahl von Grafiken wird auch bei aus MSG-Dateien extrahierten Grafiken überprüft.
- Inhalt im HTML-Format wird in die Volltextdatei aufgenommen.
- Anhänge aus signierten MSG-Dateien werden extrahiert (auch Anhänge mit unsinniger Dateiendung, sowie Anhänge, deren Namen nicht festgestellt werden können).

2.13.3.5 Nicht unterstützte Textdateitypen

Es ist jetzt möglich, die Konvertierung von Text-Dateien mit einem anderen (vom ELO Textreader nicht unterstützten) Dateityp (z. B. SQL in der ELO Administration Console zu konfigurieren, sodass diese Dateien direkt in den TXT-Ordner exportiert und von dort aus als Volltextdateien wieder importiert werden.

2.13.3.6 Neue Konverter

7z- bzw. 7zip- und vsdx-Dateien können konvertiert werden.

3 ELO Indexserver SSO

Anwendungen, die mit Single Sign On (Automatische Systemanmeldung) arbeiten, nutzten die im Betriebssystem bekannte Identität des Benutzers zur Authentifizierung. Der Vorteil dabei ist, dass sich Benutzer nicht explizit an der Anwendung anmelden müssen.

Über eine HTTP Verbindung kann SSO mittels NTLM oder SPNEGO ausgeführt werden. Das ELO System setzt dafür voraus, dass die Benutzer im Betriebssystem und in ELO mit dem gleichen Benutzernamen konfiguriert sind - das Passwort kann unterschiedlich sein.

Zur Unterstützung der SSO Funktion ist in der Indexserver WAR Datei die WAFFLE Bibliothek eingebaut. Sie funktioniert nur unter Windows. Um SSO auf anderen Betriebssystemen abbilden zu können, muss im Deployment-Deskriptor der WAR Datei ein entsprechender Servlet-Filter eingesetzt werden.



Information: Für den Benutzer *Administrator* kann kein SSO eingerichtet werden.

3.1 Voraussetzungen

- Indexserver 9.00.010
- EloixClient.jar bzw. EloixClientCS.dll aus Indexserver_Programming 9.00.010

3.2 NTLM für Indexserver konfigurieren

3.2.1 Voraussetzungen

- Der Server muss auf einem Windows-Betriebssystem laufen.
- ELO Server und ELO Client PCs müssen sich in derselben Domäne befinden
- Tomcat muss auf einem lokalen Systemkonto laufen, oder auf einem Systemkonto mit SeTcbPrivilege "Als Teil des Betriebssystems handeln".
- Auf dem Server muss Java Runtime 7.0 oder höher vorhanden sein.
- ELO Java Client-Anwendungen müssen auf Java 7.0 oder höher laufen.

3.2.2 Konfiguration

o Wenn sich nur Benutzer einer bestimmten Domäne einloggen sollen, fügen Sie in der <instdir>/config/ix-Archivname/config.xml des Indexservers diese Zeile ein:

```
<entry key="ntlm.domain">case-insensitive-domain-name</entry
```

In diesem Fall enthält die NT-Username property in der ELO Benutzerverwaltung nur den Benutzernamen, nicht den Domänen Namen.

- Wenn sich Benutzer aus unterschiedlichen Domänen anmelden, fügen Sie diese Zeile ein:

```
<entry key="ntlm.domainUserFormat">format-specification </entry>
```

Die NT-Username property muss in diesem Fall den Domänen Namen und den Benutzernamen enthalten, formatiert in der entsprechenden Formatspezifikation. Die Formatspezifikation ist ein String, in dem der Platzhalter {0} für den Domänennamen steht und der Platzhalter {1} für den Benutzernamen. Beispiel: {0}\{1} ergibt Domäne\Benutzer.

Tomcat neu starten

Für Windows 2003 Clients fügen Sie den Servernamen zu den vertrauenswürdigen Seiten im lokalen Internetbereich hinzu.

Unter ELO 9 kann das Aushandeln der Authentifizierungsmethode (Microsoft Negotiate) ausgeschaltet werden. Dazu ist folgender Eintrag in der Datei *config.xml* des ELOix zu setzen:

```
<entry key="negotiate.protocols">NTLM</entry>
```

Wenn *negotiate.protocols* mit dem Wert *NTLM* konfiguriert ist, wird das NTLM-Protokoll verwendet. Wenn *negotiate.protocols* mit dem Wert *BASIC* konfiguriert ist, wird die BASIC-Authentifizierung verwendet.

3.2.2.1 Konfiguration verifizieren

Öffnen Sie den Internet Explorer und navigieren Sie zur Indexserver Statusseite. Klicken Sie dort den Link „Test SSO Login“.

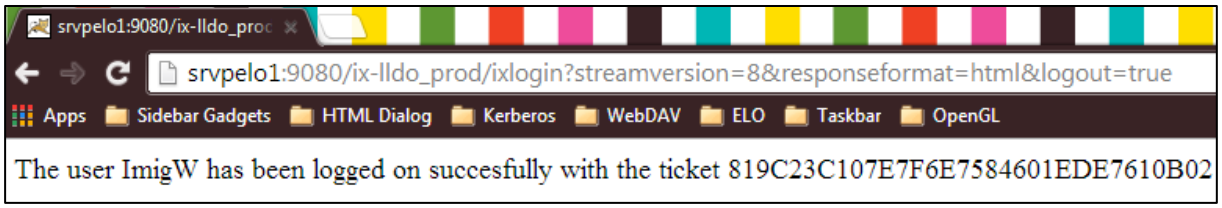


Abb. 5: Testausgabe

Der Browser sollte nicht nach Name und Passwort fragen.

3.3 SPNEGO konfigurieren

SPNEGO verwendet das Kerberos Protokoll, das in den letzten Jahren zu einem Branchenstandard geworden ist.

3.3.1 Voraussetzungen

- Auf dem Server muss Java Runtime 7.0 oder höher vorhanden sein.
- ELO Java Client-Anwendungen müssen auf Java 7.0 oder höher laufen.

Zuerst muss im Key Distribution Center (KDC) ein Kerberos-Prinzipal für den HTTP Service erstellt werden.

In der folgenden Beschreibung wird für den Server, auf dem Indexserver oder Tomcat laufen, der Servername SRVT02 verwendet. Das Passwort für den Benutzer (.z. B.: krb_SRVT02), der mit dem Prinzipal assoziiert wird, ist "elo". Dieser Benutzer muss Rechte für Kerberos Delegation haben: setspn -A HTTP/SRVT02.ELO.LOCAL krb_SRVT02.

Für die Kerberos Umgebung (Domänenname) wird "ELO.LOCAL" verwendet.

3.3.2 SPNEGO für Linux/MacOS konfigurieren

3.3.2.1 Kerberos Dienstprinzipalname und Keytab File java.keytab

- Legen Sie einen Domänen Benutzer an, z. B.: krb_SRVT02
- Mit Hilfe des ktpass command aus dem Windows Resource Kit mappen Sie den Dienstprinzipalnamen auf den Domänenbenutzer und generieren ein Keytab File. Diese Datei enthält einen privaten Schlüssel, um eingehende Anfragen zu verschlüsseln und sollte für niemanden außer für den Administrator und das Tomcat Service Konto sichtbar sein.

```
ktpass -princ HTTP/SRVT02.ELO.LOCAL@ELO.LOCAL
-pass elo -mapuser krb_SRVT02@ELO.LOCAL
-ptype KRB5_NT_PRINCIPAL
-out /etc/java.keytab
-mapOp set
```



Information:

- Benutzen Sie im Domänennamen nur GROSSBUCHSTABEN.
- Die Datei können Sie z. B. hier ablegen:

```
<eloserverinstdir>/config/Kerberos
```

3.3.2.2 Kerberos Konfigurationsdatei für Java bereitstellen: jaas-krb5.conf

Um Kerberos mit Java zu nutzen, müssen Sie für den JAAS der Java Runtime eine Konfigurationsdatei bereitstellen. Der Indexserver benötigt eine Konfigurationsdatei, die folgendermaßen aussieht:

```
IndexServer {
  com.sun.security.auth.module.Krb5LoginModule required
  debug=false
  useKeyTab=true
  storeKey=true
  keyTab="g:/ELOprofessional/config/kerberos/java.keytab"
  principal="http/SRVT02.ELO.LOCAL@ELO.LOCAL";
};
```

Information:

- Benutzen Sie im Domänennamen nur GROSSBUCHSTABEN.
- Die Datei können Sie z. B. hier ablegen:

```
<eloserverinstdir>/config/Kerberos/jaas-krb5.conf
```

- Der Kerberos Layer der Java VM schreibt umfangreiche Informationen in die stdout* log Datei, wenn der debug Parameter auf true eingestellt ist.

3.3.2.3 SPNEGO für Indexserver konfigurieren

Die Indexserver Konfiguration in der cofig.xml Datei muss um ein paar Einträge ergänzt werden. Die config.xml liegt standardmäßig im Verzeichnis

```
<eloserverinstdir>/config/ix-<archivename>.
```

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
<comment>parameters for this web application</comment>
<entry key="jdbcurl">jdbc:sqlserver://srvt02:1433</entry>
<entry key="dbdriver">com.microsoft.sqlserver.jdbc.SQLServerDriver</entry>
<entry key="dbpwd">81-41-181-112-57-23-24-141</entry>
<entry key="schema"/>
<entry key="database">elo70</entry>
<entry key="dbuser">elodb</entry>
<entry key="kerberos.jaas.conf">g:/ELOprofessional/config/jaas-
krb5.conf</entry>
<entry key="kerberos.login">IndexServer</entry>
<entry key="kerberos.kdc">NEGRIL.ELO.LOCAL</entry>
<entry key="kerberos.realm">ELO.LOCAL</entry>
</properties>
```

Property Name	Description
`kerberos.jaas.conf`	Ablageort der jaas-krb5.conf-Datei. Obligatorisch.
`kerberos.login`	Dieser Wert entspricht dem ersten Symbol in der jaas-krb5.conf-Datei. Optional, Voreinstellungen zum Indexserver.
`kerberos.kdc`	Computername des Kerberos Key Distribution Center (Active Directory Server). Optional, siehe 0. *Eine Kerberos Konfigurationsdatei bereitstellen.*
`kerberos.realm`	Kerberos Umgebung (Windows Domänenname). Optional, siehe 0. Eine Kerberos Konfigurationsdatei bereitstellen.

3.3.3 Eine Kerberos Konfigurationsdatei bereitstellen

Normalerweise sind das Kerberos Key Distribution Center und die Umgebung für alle (Java) Anwendungen auf einem Computer die gleichen. Das heißt, es ist praktischer, diese Eigenschaften in einer global verfügbaren Datei zu spezifizieren anstatt für jede einzelne Anwendung.

Java Runtime akzeptiert Kerberos Konfigurationsparameter in einer Datei mit dem Namen krb5.ini im %SystemRoot% directory. (krb5.conf in nicht-Windows Betriebssystemen).

```
The file looks like this:
[libdefaults]
    default_realm = ELO.LOCAL
[realms]
    ELO.LOCAL = {
        kdc = NEGRIL.ELO.LOCAL
    }
```

3.3.4 Integrierte Windows-Authentifizierung auf Windows-PCs

Um Kerberos Authentifizierung für HTTP (SPNEGO) mit Windows Client zu nutzen, folgen Sie den Schritten in folgender Dokumentation: https://blogs.msdn.microsoft.com/friis/2009/12/31/things-to-check-when-kerberos-authentication-fails-using-iisie/

Die Webseite, die hier hinzugefügt wird, sollte so aussehen: http://SRVT02.ELO.LOCAL

3.3.4.1 Die Konfiguration verifizieren

Wie in 3.2.2.1.



Information: Nutzen Sie immer den Kerberos-Dienstprinzipalnamen als Servernamen in der URL: SRVT02.ELO.LOCAL anstelle von SRVT02. Ansonsten misslingt die Authentifizierung und die Debug-Ausgabe im stdout*.log zeigt einen "Checksum failed" Fehler.

3.4 Client Anwendungen schreiben, die SSO über NTLM oder SPNEGO nutzen

3.4.1 Microsoft .NET Plattform

Das folgende Codesegment loggt über SSO ein:

```
  ClientInfo ci = …
IXConnFactory connFact = …
  IXConnection ix = connFact.CreateSso(ci, "mycomputer");
```

3.4.1.1 .NET auf Windows Vista, Windows 7, Windows 2008

Weil .NET 2.0 nicht das aktuellste Authentifizierungsprotokoll unterstützt, setzen Sie den Registry-Schlüssel auf:

```
HKLM\SYSTEM\CurrentControlSet\Control\Lsa\LMCompatibilityLevel=1
```

http://technet.microsoft.com/en-us/library/cc960646.aspx

3.4.2 Java 7.0 Plattform

Das folgende Codesegment loggt über SSO ein:

```
  ClientInfo ci = …
  IXConnFactory connFact = …
  IXConnection ix = connFact.createSso(ci, "mycomputer");
```

4 Indexserver Client Event API

Der Indexserver erlaubt bereits seit der Version 7.00.020 die Behandlung von Ereignissen (z. B. Einchecken eines Dokuments) in JavaScript-Funktionen oder Webservices. Bisher konnte die Ereignisverarbeitung aber nur auf dem Server erfolgen. Ab der Version 8.00.010 ist es möglich, auf bestimmte Ereignisse (z. B. Änderung der Untereinträge eines Ordners) in der Clientanwendung zu reagieren. Darüber hinaus können sich Client-Anwendungen untereinander in einer Art Instant-Messaging selbstdefinierte Ereignisse zusenden.

4.1 Einführungsbeispiel: Untereinträge eines Ordners überwachen

Das folgende Beispiel zeigt, wie ein Ordnerinhalt überwacht werden kann. Werden Untereinträge hinzugefügt, gelöscht oder bearbeitet, erhält die Client-Anwendung ein Ereignis.

Voraussetzung für die Abarbeitung von Ereignissen ist, dass die Client-Anwendung eine Schnittstelle bereitstellt, an die die Ereignisse übergeben werden können. In .NET Anwendungen geschieht dies in Form eines Delegate-Objekts, das an die Event-Eigenschaft „EventBusHandler" angehängt wird.

Anschließend kann durch einen oder mehrere „EventListener" beschrieben werden, welche Ereignisse für die Client-Anwendung von Interesse sind. Die Funktion „AddListener" richtet hier einen „EventListener" für den Ereignistyp „EventBusC.EVENT_TYPE_WATCH_FOLDER" ein. Zu diesem Ereignistyp muss noch angegeben werden, welcher Ordner überwacht werden soll. Dies geschieht durch Übergabe von „folderId".

```
public static void WatchFolder()
{
    IXConnection conn = …

    conn.EventBusApi.EventBusHandler += new
EventBusApi.ProcessEventBusEvents(

EventBusApi_EventBusHandler);

    // This folder is being watched
    String folderId = "1";

    // Add event listener for watching the folder

conn.EventBusApi.BroadcastBus.AddListener(EventBusC.EVENT_TYPE_WATCH_FOLDER
, 0,
                                              folderId);

    // Process events for 1min
    Thread.Sleep(60 * 1000);

    // Delete all attached event listeners
    conn.EventBusApi.BroadcastBus.Release();
```

```
}

static void EventBusApi_EventBusHandler(long subsId, Event[] events)
{
    foreach (Event e in events)
    {
        Object[] args = (Object[])AnyToObject.ToObject(e.any);
        Sord sord = (Sord)args[0];
        SordZ sordZ = (SordZ)args[1];
        int what = (int)args[2];

        Console.WriteLine("Event: " + e.type +
            ", folderId=" + e.param2 +
            ", modified sord=" + sord.name +
            ", what=" + what);
    }
}
```

Wenn das Programm gestartet wird, werden für eine Minute die Änderungsereignisse der obersten Archivstruktur ausgegeben.

4.2 Konzeptioneller Aufbau

Im Zentrum der Ereignisweiterleitung stehen die Ereignisbusse („Event Busses“), an die Ereignisse zur Verteilung gesendet werden (siehe *Abb. 6: Ereignisbus und Teilnehmer*).

An einem Ereignisbus können die Teilnehmer („Subscribers“) Ereignisse abhören. Ein Teilnehmer ist in der Regel gleichbedeutend mit einer Sitzung. Teilnehmer können auch Ereignisse an einen Ereignisbus senden, z. B. um mit anderen Teilnehmern Nachrichten auszutauschen. Des Weiteren werden Ereignisse von Indexserver-Funktionen wie z. B. checkinSord erzeugt.

Ein Teilnehmer kann einen vordefinierten Ereignisbus oder einen selbst erstellten für die Kommunikation nutzen. Zu den vordefinierten Ereignisbussen gehört der „Broadcast Event Bus“, der eine uneingeschränkte Kommunikation der Teilnehmer untereinander erlaubt. Daneben stellt der Indexserver je Benutzer einen Ereignisbus bereit, der wie ein Postfach funktioniert: nur der Eigentümer kann Ereignisse abhören, andere können nur Ereignisse senden.

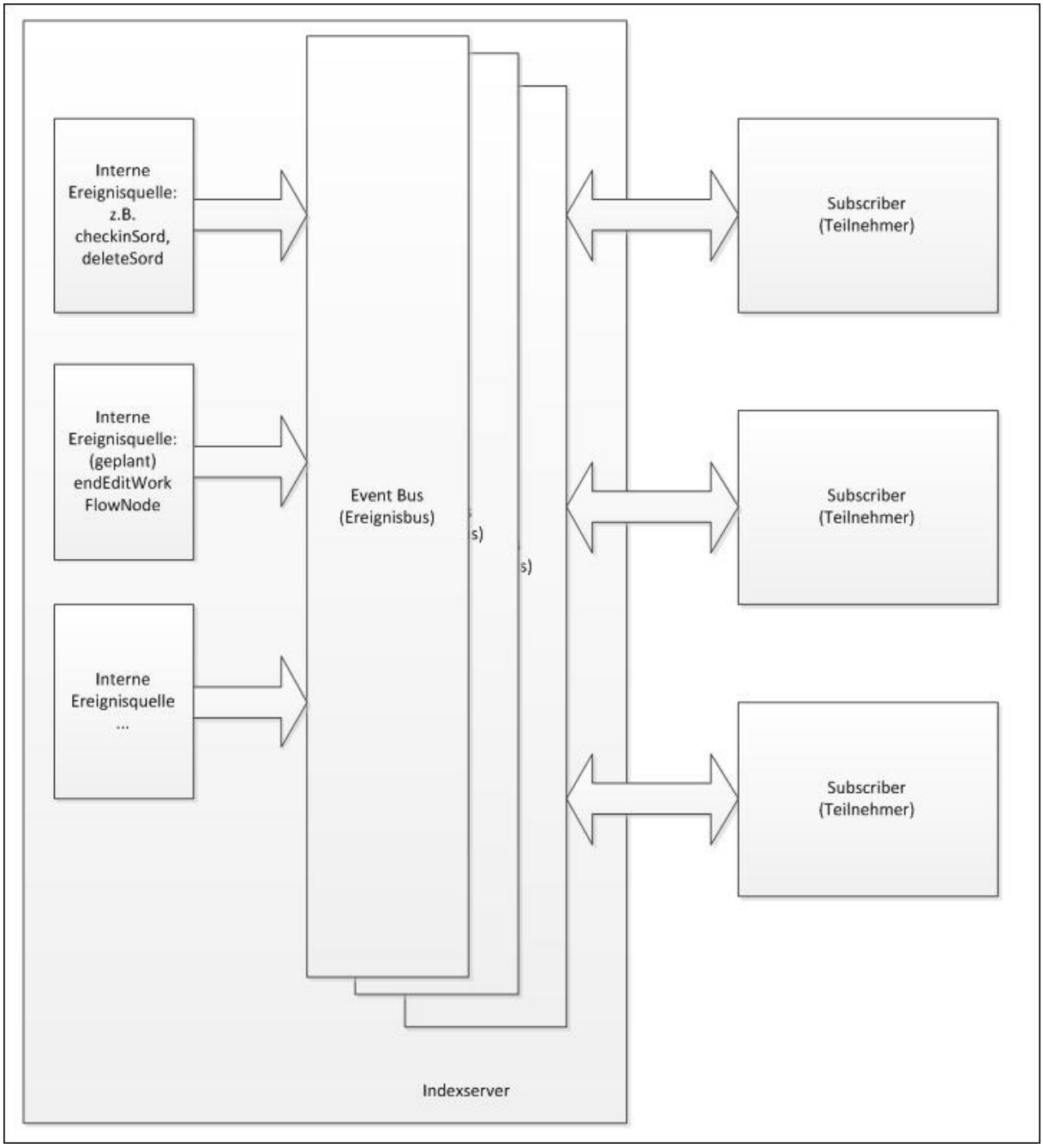


Abb. 6: Ereignisbus und Teilnehmer

Damit die Client-Anwendung Ereignisse behandeln kann, muss sie zunächst einen „EventBusHandler" bereitstellen. In Java-Programmen geschieht dies durch die Übergabe eines Objekts, das die Schnittstelle „IXEventBusHandler" implementiert, an die Funktion „IXConnection.EventBusApi.setHandler". In .NET-Programmen ist der „EventBusHandler" ein Delegate-Objekt, das an das Ereignis „IXConnection.EventBusApi.EventBusHandler" gehängt wird.

Mit dem Zuweisen des „EventBusHandler" wird in der „IXConnection" zugleich auch ein Teilnehmer erstellt.

Anschließend können „EventListener“ eingerichtet werden, deren „EventFilter“ die gewünschten Ereignisse aus den Bussen abhören und an die Client-Anwendung weitergeben. „EventListener“ können zu jeder Zeit eingerichtet und gelöscht werden. In der Abbildung 5 sind die Sachverhalte noch einmal grafisch dargestellt.

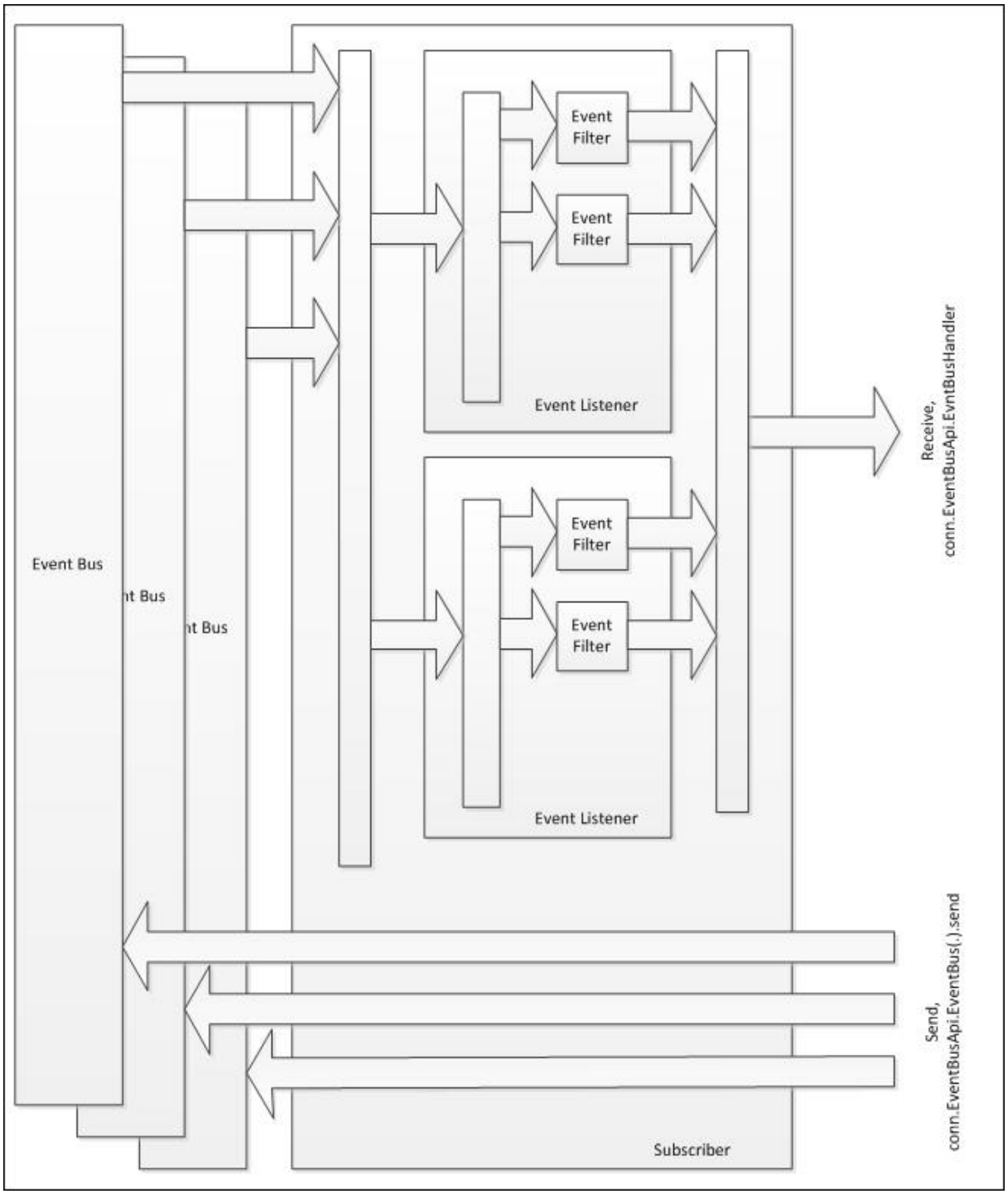


Abb. 7: Teilnehmer, Ereignis-Listener und -Filter

Über die Ereignisse können Nachrichten vom Typ „Any“ übertragen werden. Dieser Typ ist in der Lage, die Standardtypen wie int, long, String usw. aufzunehmen. Darüber hinaus können auch alle API-Klassen der Indexserver-Schnittstelle in ein Any eingepackt werden, z. B. Sord oder DocMask.

4.3 Weitere Informationen

4.3.1 Umgang mit Ereignissen

4.3.1.1 Ereignistypen

Es gibt einige vordefinierte Ereignistypen, deren Werte als Konstanten in EventBusC definiert sind. Darüber hinaus können Anwendungen ihre eigenen Ereignistypen definieren. Ihre Werte müssen größer als EventBusC.EVENT_TYPE_MAX_SYSTEM sein. Ereignistypen mit kleineren Werten sind für ELO reserviert.

Um Überschneidungen der Werte verschiedener Anwendungen zu vermeiden, sollten sie mit einem 64bit Zufallszahlengenerator gebildet werden.

In .NET Programmen kann man wie folgt vorgehen:

```
long computeMyEventType()
{
    Random rand = new Random();
    long type = 0;

    while (type <= EventBusC.EVENT_TYPE_MAX_SYSTEM)
    {
        byte[] buf = new byte[8];
        rand.NextBytes(buf);
        type = BitConverter.ToInt64(buf, 0);
        if (type < 0) type = -type;
    }
    return type;
}
```

In Java-Programmen wäre der folgende Code geeignet:

```
  public static long createMyEventType()  {
    Random rand = new Random();
    long type = 0;

    while (type <= EventBusC.EVENT_TYPE_MAX_SYSTEM)
    {
      byte[] buf = new byte[8];
      rand.nextBytes( buf );
      BigInteger bint = new BigInteger(buf);
      type = bint.longValue();
      if (type < 0) type = -type;
    }
```

```
    return type;
  }
```

Um eine private Kommunikation zwischen Client-Anwendungen aufzubauen, sollten nicht nur eigene Ereignistypen verwendet werden. Zusätzlich sollten die Ereignisse auch über einen anwendungsdefinierten Ereignisbus statt über den Broadcast-Bus laufen. Dadurch kann der Indexserver schneller die Empfänger der Ereignisse finden.

4.3.1.2 Ereignis-Parameter

Neben dem Ereignistyp können die optionalen Parameter Event.param1 und Event.param2 verwendet werden, um das Ereignis genauer zu beschreiben. Die Ereignisfilter im Indexserver werten neben dem Typ auch die Parameter aus, um zu entscheiden, ob ein Teilnehmer das Ereignis bekommt.

4.3.1.3 Ereignis-ACL

Einem Ereignis kann eine ACL mitgegeben werden, sodass nur leseberechtigte Teilnehmer das Ereignis empfangen.

4.3.1.4 Ereignis-IDs

Wenn Client-Anwendungen über die Ereignis-API kommunizieren wollen, dann wäre möglicherweise eine Art Request-Response-Protokoll hilfreich. Es würde ermöglichen, dass ein Sender auf eine Anfrage eine zugehörige Antwort empfangen könnte. In der Ereignis-API wird jedoch kein Request-Response-Protokoll zwischen Clients angeboten. Dies kann aber leicht selbst entwickelt werden.

Für den Sender ist es dabei wichtig herausfinden zu können, welches gesendete Ereignis zu welchem empfangenen passt. Er kann dies anhand der Ereignis-ID ermitteln, die er frei vergeben kann. Er muss sich dabei darauf verlassen können, dass der Empfänger die ID der Anfrage in die Antwort einträgt.

Die Ereignis-IDs werden vom ELOix nicht ausgewertet und können beliebige Werte annehmen.

4.3.1.5 Ereignisdaten

Die Nutzlast eines Ereignisses wird im Element Event.any transportiert. Any-Objekte können verschiedene Datentypen enthalten. Neben den Standardtypen wie int, long, String sind auch alle Klassen erlaubt, die in der Indexserver-Schnittstelle definiert sind – also z. B. auch Sord. Darüber hinaus können Arrays transportiert werden. Die Any-Klasse wird auch in der API der Registrierten Funktionen verwendet.

4.3.2 Ereignis-Listener

Die Ereignis-Listener fassen eine Liste von Ereignis-Filtern zusammen. Damit können mehrere Filter mit einem Aufruf eingerichtet und wieder gelöscht werden.

Listener werden automatisch gelöscht, wenn die Sitzung abgemeldet oder die Sitzungslebenszeit überschritten wird.

4.3.3 Ereignisbusse

Ereignisse können an vordefinierte Busse gesendet werden, die immer verfügbar sind und nicht explizit geöffnet werden müssen. Daneben können Anwendungen selbst Busse definierten, die sie mit der Funktion openEventBus öffnen und mit closeEventBus schließen müssen.

4.3.3.1 Broadcast Bus

Dieser vordefinierte Bus hat die ID EventBusC.BUSID_BROADCAST und erlaubt es jeder beliebigen Sitzung Ereignisse zu senden und abzuhören.

4.3.3.2 User Bus

Für jeden Benutzer gibt es einen vordefinierten Bus, den nur er selbst abhören kann. Andere Benutzer können nur Ereignisse an den Bus senden. Der Bus funktioniert also ähnlich wie ein Postfach. Die Bus-ID eines Benutzers errechnet sich aus EventBusC.BUSID_USER + <Benutzernummer>.

4.3.3.3 Anwendungsdefinierte Busse

Anwendungen können selbst neue Ereignisbusse öffnen. Dies ist beispielsweise dann interessant, wenn eine Anwendung aus verschiedenen Prozessen besteht, die unter derselben Sitzung (Ticket) arbeiten und miteinander kommunizieren sollen. Hierfür ist ein Ticket-bezogener Ereignisbus hilfreich.

Die Funktion openEventBus der Indexserver API erlaubt es, einen benutzerdefinierten Bus zu öffnen. Sie nimmt als Parameter ein EventBusParams Objekt entgegen. Wenn zwei Anwendungen gleiche Parameterobjekte übergeben, dann erhalten sie von openEventBus dieselbe Bus-ID zurück.

Die mit openEventBus erstellten Busse müssen mit closeEventBus wieder geschlossen werden.

4.3.4 Ereignis-Handler

Der Aufruf der Ereignis-Handler-Funktion erfolgt in einem Hintergrund-Thread. Für .NET-Anwendungen gibt es dafür einen Thread je IXConnection. In Java-Anwendungen stellt HttpAsycClient zwei Threads bereit, die aber für alle IXConnection Objekte einer IXConnFactory verwendet werden. Java spart also Threads durch asynchrones IO.

In beiden Umgebungen sollte die Durchführung der Ereignis-Handler-Funktion möglichst schnell von statten gehen. Solange die Funktion läuft, kann der Hintergrund-Thread keine Ereignisse empfangen. Lang dauernde Verarbeitungen (SQL-Abfragen, WebService-/ELOix-Aufrufe) sollten an einen anderen Thread übergeben werden.

4.4 Anwendungsbeispiele

4.4.1 Ordner in einer Anwendung mit Benutzeroberfläche überwachen

In UI-Anwendungen ist in der Regel darauf zu achten, dass die Ereignis-Handler-Funktion nicht direkt die UI-Elemente aktualisieren kann, weil sie aus einem Hintergrund-Thread aufgerufen wird. Die Ereignisinformationen müssen zuerst an den UI-Thread übergeben werden.

Dafür stellen .NET und Java die nötigen Funktionen bereit. In .NET benutzt man Control.Invoke und übergibt ein Delegate-Objekt. Das Java-AWT definiert EventQueue.invokeLater, und Java-Swing stellt SwingUtilities.invokeLater bereit.

Im Folgenden wird ein .NET Beispieldialog diskutiert, indem die ID des zu überwachenden Ordners eingegeben wird. In einer Liste werden alle Ereignisse zu Änderungen der Untereinträge aufgelistet. (Das Beispiel gehört zu dem .NET-Example-Projekt der IndexServer_Programming.zip).

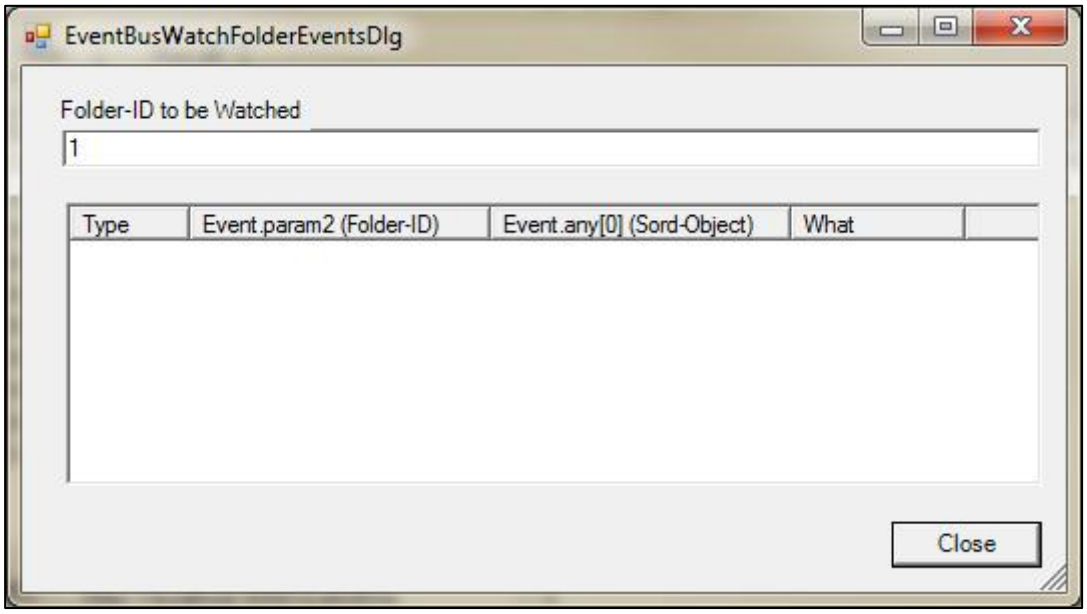


Abb. 8: EventBusWatchFolderEventsDlg

Der Konstruktor EventBusWatchFolderEventsDlg(IXConnection ix) richtet zum einen den Ereignis-Handler im IXConnection-Objekt ein. Zum anderen ruft er initListener auf, um einen Ereignis-Listener für die Ordner-Änderung zu setzen.

Die Ereignis-Handler-Funktion EventBusApi_EventBusHandler erstellt ein Delegate-Objekt vom Typ ShowEventsCallback mit der Funktion ShowEvents und übergibt es zusammen mit den Ereignissen an Control.Invoke. Somit wird ShowEvents mit der Ereignisliste im UI-Thread aufgerufen.

Ändert sich die Ordner-ID, ruft die Funktion edObjId_TextChanged wieder initListener auf, um einen neuen Ereignis-Listener für diesen Ordner zu erstellen. Zuvor wird der Ereignis-Listener auf den vorigen Ordner gelöscht.

Wird der Dialog geschlossen, so stellt EventBusWatchFolderEventsDlg_FormClosed sicher, dass keine weiteren Ereignisse mehr beim Dialog ankommen, indem der Ereignis-Handler abgehängt wird. Anschließend werden mit conn.EventBusApi.BroadcastBus.Release() noch alle Listener gelöscht, die noch mit dem Broadcast-Bus verbunden sind – hier ist es nur ein Listener, nämlich der in initListener eingerichtete.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using EloixClient.IndexServer;

namespace Examples
{
    /// <summary>
    /// This dialog shows the events sent by ELOix, when the
    /// subitems of a folder have changed.
    /// </summary>
    /// <remarks>
    /// The events are received from a background thread.
    /// To show the events in the list view,
    /// the control has to be called in a thread safe way.
    /// See: http://msdn.microsoft.com/en-
us/library/ms171728(v=vs.100).aspx
    /// </remarks>
    public partial class EventBusWatchFolderEventsDlg : Form
    {
        // Indexserver connection
        IXConnection conn;

        // Current listener ID
        long watchFolderLsnId;

        /// <summary>
        /// This delegate type is impemented by the ShowEvents function.
        /// </summary>
        /// <param name="subsId">Subscriber ID</param>
        /// <param name="events">Events</param>
        delegate void ShowEventsCallback(long subsId, Event[] events);

        /// <summary>
        /// Constructor
```

```
        /// </summary>
        /// <param name="conn">Indexserver connection</param>
        public EventBusWatchFolderEventsDlg(IXConnection conn)
        {
            this.conn = conn;

            // Create dialog UI
            InitializeComponent();

            // Add an event bus handler function to the connection.
            conn.EventBusApi.EventBusHandler +=
                new
EventBusApi.ProcessEventBusEvents(EventBusApi_EventBusHandler);

            // Add a listener for the object ID given in TextBox edObjId
            initListener();
        }

        /// <summary>
        /// Delete the current listener, add a new listener.
        /// </summary>
        private void initListener()
        {
            // Delete current listener
            if (watchFolderLsnId != 0L)
            {

conn.EventBusApi.BroadcastBus.DeleteListener(watchFolderLsnId);
                watchFolderLsnId = 0L;
            }

            // Add new listener
            try
            {
                watchFolderLsnId =
conn.EventBusApi.BroadcastBus.AddListener(
                    EventBusC.EVENT_TYPE_WATCH_FOLDER, 0, edObjId.Text);
            }
            catch (Exception)
            {
                // Ignored.
                // initListener() is called on each change in TextBox
edObjId.
                // There might be an invalid object ID during editing.
            }
        }

        /// <summary>
        /// Event bus handler function.
        /// </summary>
        /// <remarks>
```

```
        /// This function is called in a background thread.
        /// Thus, it cannot access the dialog elements of this form
directly.
        /// We have to create a delegate object that contains the call
        /// ShowEvents to update the UI.
        /// This delegate is passed to the Invoke function which executes
        /// ShowEvents in the UI thread.
        /// </remarks>
        /// <param name="subsId">Subscriber ID</param>
        /// <param name="events">Events</param>
        void EventBusApi_EventBusHandler(long subsId, Event[] events)
        {
            ShowEventsCallback d = new ShowEventsCallback(ShowEvents);
            this.Invoke(d, new Object[] { subsId, events });
        }

        /// <summary>
        /// This function adds the events to the list view.
        /// </summary>
        /// <param name="subsId">Subscriber ID</param>
        /// <param name="events">Events</param>
        private void ShowEvents(long subsId, Event[] events)
        {
            foreach (Event e in events)
            {
                ListViewItem item = new ListViewItem();
                item.Text = Convert.ToString(e.type);
                item.SubItems.Add(e.param2);
                Object[] args = (Object[])AnyToObject.ToObject(e.any);
                Sord sord = (Sord)args[0];
                item.SubItems.Add(sord.name);
                int what = (int)args[2];
                switch (what)
                {
                    case EventBusC.WATCH_INSERT:
item.SubItems.Add("inserted"); break;
                    case EventBusC.WATCH_UPDATE:
item.SubItems.Add("updated"); break;
                    case EventBusC.WATCH_DELETE:
item.SubItems.Add("deleted"); break;
                }
                this.lvEvents.Items.Add(item);
            }
        }

        /// <summary>
        /// Update the listener on each change of the TextBox.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void edObjId_TextChanged(object sender, EventArgs e)
```

```
        {
            initListener();
        }

        /// <summary>
        /// Release the broadcast event bus in order to delete attached
        /// listeners and detach the event bus hander.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void EventBusWatchFolderEventsDlg_FormClosed(object sender,
            FormClosedEventArgs e)
        {
            try
            {
                // Detach event bus hander.
                // EventBusApi_EventBusHandler must not be called
                // after the Form has been disposed.
                conn.EventBusApi.EventBusHandler -=
                    new
EventBusApi.ProcessEventBusEvents(EventBusApi_EventBusHandler);

                // Delete all of my attached listeners.
                conn.EventBusApi.BroadcastBus.Release();
            }
            catch (Exception)
            {
            }
        }


    }
}
```

4.4.2 Client-Client-Kommunikation

Es können nicht nur Ereignisse vom Server zur Client-Anwendung versendet werden. Auch können Client-Anwendungen untereinander Informationen austauschen.

Das folgende Beispiel simuliert eine Verbindung zwischen zwei Clients, indem zwei Sitzungen geöffnet werden. Die erste Sitzung, conn1, wartet auf ein Ereignis, die zweite sendet es. Beiden Sitzungen ist der Ereignistyp myEventType und der Kanalname channelName vorab bekannt.

```
public static void SendMessage()
{
    IXConnFactory connFact = …

    // Create a most likely unique event ID for this communication.
    long myEventType = computeMyEventType();
```

```
    // The two connection share the same event bus.
    // Its bus ID is created based on this channel name.
    String channelName = "de.elo.ix.tutorial.SendReceive";

    // Create first connection - the receiver
    IXConnection conn1 = connFact.Create("Administrator", "elo",
computerName, null);

    // Add an event handler to the receiver
    conn1.EventBusApi.EventBusHandler += new
EventBusApi.ProcessEventBusEvents(
        EventBusApi_EventBusHandler2);

    // Open the channel.
    EventBusApi.Bus channelBus1 = conn1.EventBusApi.OpenEventBusChannel(
        channelName);

    // Add a listener to the channel,
    // in order to receive events of type myEventType.
    channelBus1.AddListener(myEventType);

    // Create second connection - the sender
    IXConnection conn2 = connFact.Create("Administrator", "elo",
computerName, null);

    // Open the channel.
    // channelBus2 uses the same bus ID as channelBus1, because it is
opened
    // with the same channel name.
    EventBusApi.Bus channelBus2 =
conn2.EventBusApi.OpenEventBusChannel(channelName);

    // Send message
    String message = "hallo conn2";
    Console.WriteLine("send event: type=" + myEventType + ", message=" +
message);
    channelBus2.Send(myEventType, message);

    // Wait some time.
    // The first connection should receive the message during this time.
    Thread.Sleep(2 * 1000);

    // Delete all listeners from the channel.
    // A bus is deleted inside Indexserver, if the last listener is
deleted.
    channelBus1.Release();

    // Logout connection.
    conn2.Logout();
    conn1.Logout();
}
```

```
static void EventBusApi_EventBusHandler2(long subsId, Event[] events)
{
    foreach (Event e in events)
    {
        Console.WriteLine("received event: type=" + e.type +
                          ", message=" +
AnyToObject.ToObject(e.any).ToString());
    }
}
```

Dieses Beispiel stellt bereits die Basis einer Instant-Messaging-Anwendung dar, über die Benutzer chatten könnten.

Neben Textnachrichten können aber auch andere Informationen versandt werden. Beispielsweise könnte von einer Arbeitsplatzanwendung ein Sord-Objekt an einen Serverprozess übermittelt werden, der damit bestimmte Aktionen durchführt.

Mit etwas mehr Aufwand kann die Kommunikation in ein Request-Response-Schema gefasst werden, sodass der Sender der Nachricht vom Empfänger eine Antwort bekommt.

4.5 Technische Realisierung

Die Kommunikation zum Indexserver erfolgt über das HTTP-Protokoll, das prinzipiell Verbindungen nur in einer Richtung – vom Client zum Server – zulässt. Für die Ereignisse, die im Server ausgelöst und im Client behandelt werden sollen, wird jedoch eigentlich die umgekehrte Richtung benötigt.

Eine Lösung könnte sein, die Client-Anwendung in kurzen Abständen nach neuen Ereignissen fragen zu lassen. Der Nachteil davon ist aber, dass bei vielen Client-Anwendungen auch sehr viele Aufrufe übers Netzwerk gehen und vom Indexserver verarbeitet werden müssen.

Weniger Anfragen verursacht eine Technik, die oft als „Long-Poll“ bezeichnet wird. Hierbei sendet der Client eine Anfrage, die vom Server solange nicht beantwortet wird, bis die gewünschten Daten vorliegen. Das gravierendste Problem dieser Vorgehensweise war in der Vergangenheit, dass in der Server- und Client-Anwendung je ein Thread für das Warten verbraucht wurde. Threads sind aber recht teure Objekte des Betriebssystems, mit denen man sparsam haushalten sollte.

Mit der Servlet Spezifikation 3.0, die im Tomcat 7.0 umgesetzt ist, steht die Möglichkeit der asynchronen Verarbeitung von Anfragen zur Verfügung. Sie erlaubt es, die Kommunikations-Streams im Server noch offen zu halten, nachdem die Servicemethode des Servlets zurückgekehrt ist. So kann zu einem quasi beliebigen Zeitpunkt aus einem beliebigen Thread heraus eine Antwort an den Client gesendet werden.

Auch auf der Client-Seite stellt sich das Problem des wartenden Threads. Sofern die Anwendung aber nicht selbst eine Serveranwendung ist, die viele Verbindungen zum Indexserver aufmacht (z. B. Web-Client), ist der zusätzliche Thread nicht tragisch.

5 Indexserver JSON API

Ab der Version 8.00.014 können Indexserver Anwendungen auch in JavaScript für Webbrowser geschrieben werden. Die Kommunikation erfolgt synchron oder asynchron mithilfe des XmlHTTPRequest-Objekts, das von allen gängigen Webbrowsern bereitgestellt wird. Die Anfragen und Ergebnisse werden im JSON-Format übertragen. Es stehen alle ELO Indexserver-Aufrufe zur Verfügung, bis auf das lokale Speichern von Dokumenten. Das Hochladen von Dokumenten ist mit freien JavaScript-Bibliotheken, die auf dem HTML-Input-File-Element basieren, möglich.

5.1 Einführungsbeispiel: Anzeige eines Dokuments nach Objekt-ID

Das folgende Beispiel zeigt die Anmeldung am ELO Indexserver, das Lesen der Daten eines Objekts und die Anzeige der Dokumentendatei in einem zweiten Browserfenster oder –Tab.

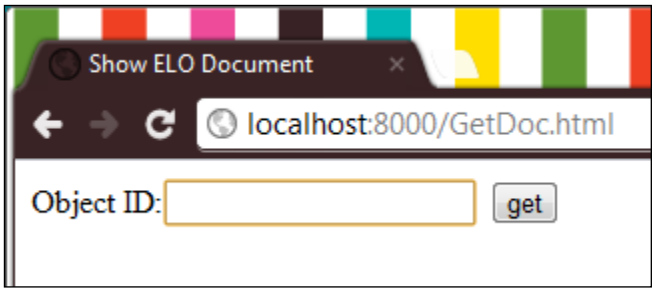


Abb. 9: Formular des Einführungsbeispiels im Browser

Es besteht aus einer HTML Datei, die ein einfaches Formular, wie in der Abbildung gezeigt, darstellt. Im Eingabefeld wird eine Objekt-ID erwartet, und beim Klick auf die Schaltfläche *get* wird das zugehörige Dokument angezeigt.

Der HTML-Code sieht wie folgt aus.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Show ELO Document</title>

<!-- Include Indexserver JavaScript Library -->
<script type="text/javascript" src="EloixClient.js"></script>

<!-- Example code -->
<script type="text/javascript">

// Simplify namespace access
// (all ELOix symbols are member of de.elo.ix.client)
var IX = de.elo.ix.client;

// Indexserver URL
```

```
// Since cross domain requests are not supported, Indexserver must be
reachable
// via the same server from where this example HTML file is loaded.
var ixUrl = "/ix-elo80/ix";

/**
 * This function reads the object given by its ID in the element
"searchTerm".
 * If the object exists, it navigates the browser to its URL.
 * If the object does not exsit, it opens an alert box and displays the
error message created by Indexserver.
 */
function Example_getDoc() {
    try {
      // Get Indexserver connection
      var connFact = new IX.IXConnFactory(ixUrl, "GetDoc", "1.0");
      var conn = connFact.create("Administrator", "elo");

      // Read API constants
      var CONST = conn.getCONST();

      // Get object ID from HTML element
      var objId = document.getElementById("searchTerm").value;

      // Read object
      var ed = conn.ix().checkoutDoc(objId, "", CONST.EDIT_INFO.mbDocument,
                                     CONST.LOCK.NO);

      // Assing URL to download link
      document.getElementById("downloadLink").href =
                  ed.document.docs[0].url + "&acode=attachment";

      // Assign URL to the link that shows the document in the browser
      document.getElementById("viewLink").href =
                 ed.document.docs[0].url; // + "&acode=inline" is default
    }
    catch (ex) {
       // Show message box with error text.
      alert(ex.toString());
    }
    finally {
     // Skip logout in this example.
     // Otherwise the document URL might not be valid when the browser
loads it.
    }
};
</script>
</head>
<body>
  Object ID:<input id="searchTerm" type="text" value=""/>
```

```
  <input id="bnSearch" type="button" value="get" onclick="Example_getDoc()"
  <a href="about:blank" id="downloadLink">Download</a><br/>
  <a href="about:blank" id="viewLink">View</a><br/>
</body>
</html>
```

Die erste interessante Zeile des HTML-Codes ist

```
<script type="text/javascript" src="EloixClient.js"></script>
```

Sie hat die Aufgabe, die JavaScript-Klassen und –Funktionen einzubinden, die für die Ansteuerung des Indexservers benötigt werden. Die Datei kann von der Statusseite des Indexservers heruntergeladen werden. Es gibt sie als ausführliche Version mit Kommentaren (EloixClient.js) und als komprimierte Version ohne Kommentare und Formatierungen (EloixClient-min.js). Letztere ist nur ein Drittel so groß und sollte wegen der kürzeren Ladezeit in produktiven Anwendungen benutzt werden.

Im darauffolgenden Script-Block ist die Programmlogik für das Beispiel definiert. Er startet damit, einen einfacheren Zugriff auf die Indexserver-Klassen bereitzustellen.

```
var IX = de.elo.ix.client;
```

Denn die ELO Indexserver-Symbole sind im Package „de.elo.ix.client“ enthalten und dieses muss jedem Klassennamen vorangestellt werden. Um Tipparbeit im Beispielcode zu sparen, wird zum langen Package-Namen der Alias „IX“ definiert. Statt „de.elo.ix.client.IXConnFactory“ kann somit kürzer „IX.IXConnFactory“ geschrieben werden.

Die Zeile

```
var ixUrl = "/ix-elo80/ix";
```

Legt die URL zum ELO Indexserver fest. Weil die Webbrowser im JavaScript nur erlauben, Anfragen an denjenigen Server zu stellen, von dem auch die HTML-Seite stammt, ist die URL hier relativ zum Server angegeben. Entweder muss also die HTML-Seite vom Server kommen, auf dem auch der ELO Indexserver läuft. Oder bestimmte Maßnahmen müssen dies dem Browser glauben machen. Der Abschnitt *5.2 Cross-Domain Anfragen* beschäftigt sich ausführlich mit diesem Thema.

Beim Klick auf die Schaltfläche „get“ wird die Funktion „Example_getDoc“ ausgeführt. Sie erstellt im ersten Schritt eine Verbindungs-Factory, von der aus mit der „create“ Funktion eine Anmeldung am Indexserver erfolgt.

```
      var connFact = new IX.IXConnFactory(ixUrl, "GetDoc", "1.0");
      var conn = connFact.create("Administrator", "elo");
```

Lesern, die bereits mit der Java- oder .NET-Schnittstelle gearbeitet haben, wird dieses Vorgehen bekannt vorkommen. Überhaupt ist die JSON-API sehr eng an die bestehenden Indexserver-APIs angelehnt.

In einem Punkt gibt es aber einen deutlichen Unterschied: die Konstanten sind nicht über statische Klassenelemente zugreifbar. Sie müssen wie bei der SOAP-Schnittstelle aus dem Konstanten-Objekt geholt werden. Im Beispiel wird das Konstantenobjekt für den einfacheren Zugriff in einer Hilfsvariablen gespeichert:

```
var CONST = conn.getCONST();
```

Die darauffolgende Zeile

```
var objId = document.getElementById("searchTerm").value;
```

liest die Objekt-ID aus dem HTML-Eingabefeld.

Sie wird der Funktion „checkoutDoc“ übergeben, die die Dokumentdaten liest.

```
var ed = conn.ix().checkoutDoc(objId, "", CONST.EDIT_INFO.mbDocument,
                               CONST.LOCK.NO);
```

Ist das Dokument vorhanden, enthält die Rückgabe von „checkoutDoc“ in „ed.document.docs[0]“ die Daten zur aktuellen Arbeitsversion des Dokuments. Insbesondere findet sich in „ed.document.docs[0].url“ die URL, unter der das Dokument vom Browser geladen werden kann.

Die URL kann direkt einem Anker-Element zugewiesen werden. Wenn der String „acode=attachment“ angehängt wird, dann erhält der Browser die Anweisung die Datei herunterzuladen, statt sie im Browserfenster anzuzeigen.

```
… ed.document.docs[0].url + "&acode=attachment";
```

In diesem Beispiel wird auf eine Abmeldung des Benutzers bewusst verzichtet. Denn üblicherweise lädt der Browser das Dokument in einem parallelen Prozess (oder Thread). So lange muss die Sitzung auf jeden Fall gültig sein.

In der Praxis könnte man die Abmeldung in eine Timer-Funktion packen. Oder man sorgt zumindest dafür, dass nicht bei jedem Klick auf „get“ eine neue Sitzung erstellt wird und speichert sie in einer globalen Variablen.

5.2 Cross-Domain Anfragen

Wer mit dem Browser-Objekt XmlHttpRequest bereits gearbeitet hat, ist bestimmt schon auf die Einschränkung gestoßen, dass es aus Sicherheitsgründen nur HTTP-Anfragen zu demjenigen Server erlaubt, von dem auch die HTML Seite kommt. Dies wird häufig als „Same Origin Policy“ bezeichnet.

Diese Einschränkung ist insbesondere während der Entwicklung einer Anwendung störend. Sie kann aufgehoben werden, wenn die HTML-Datei mit dem HTTP-Header „Access-Control-Allow-Origin : <server>“ ausgeliefert wird. Dies ist allerdings oft nicht einfach möglich, weil die HTTP-Header nicht in der HTML-Datei enthalten sind sondern vom Server vergeben werden. (s. http://en.wikipedia.org/wiki/XMLHttpRequest#Cross-domain_requests)

Programmierer von JavaScript-Bibliotheken, wie z. B. jQuery, umgehen die Sicherheitsprüfung, indem sie in den DOM-Baum des Dokuments ein <script> Tag einfügen und dies mit der gewünschten URL laden. Das funktioniert, weil dieses Tag nicht der „Same Origin Policy" unterliegt und Daten von beliebigen Servern laden kann. Nachteil dieser Lösung ist, dass der Aufruf nicht synchron erfolgen kann. Mit asynchronen Aufrufen wird aber die Programmlogik schnell sehr unübersichtlich.

Die einfachste Lösung für das Cross-Domain-Problem ist der Einsatz eines Reverse-Proxys. Dieser verteilt die Anfragen an beliebig viele Server und stellt sich dem Browser gegenüber als ein einziger Server dar. So kann man seine HTML Seiten auf dem einen Server entwickeln und den Indexserver auf einem anderen Server ansprechen.

Auch in der Praxis ist dies eine günstige Konfiguration, wenn man über das Internet auf ELO Daten zugreifen will. Der Reverse-Proxy läuft dafür in der DMZ und leitet die Anfragen an die Server im geschützten Intranet weiter.

Im Abschnitt *5.8 Reverse-Proxy mit dem Apache2 Webserver* wird gezeigt, wie mit wenigen Schritten ein Reverse-Proxy aufgesetzt werden kann.

5.3 Synchrone und asynchrone Aufrufe

Bis auf die Anmeldung über die create-Funktionen der IXConnFactory können alle Funktionen der API auch asynchron aufgerufen werden. Hierzu ist im letzten Parameter ein de.elo.ix.client.AsyncCallback-Objekt mit Callback-Funktionen zu übergeben.

Das folgende Beispiel zeigt den asynchronen Aufruf der findFirstSords-Funktion. Dem Callback-Objekt werden neben den Rückruffunktionen weitere Elemente hinzugefügt, um später ggf. findNextSords aufrufen zu können. Insbesondere werden auch die Icons für die Dokumente bereitgestellt (s. checkoutSordTypes), sodass die Funktion „Example_printSord" das zum Dokument gehörende Icon in einem HTML-Image-Element anzeigen kann.

```
/**
 * This function perfoms a fulltext search asynchronously.
 * The search term is read from the HTML element with id="searchTerm".
 * The result list is printed into the HTML table with id="searchResults".
 */
function Example_doSearchAsync() {

    Example_clearResults();

    try {

        // Get Indexserver connection
        var conn = getConn();

        // Get Indexserver API constants
        var CONST = conn.getCONST();

         // Setup FindInfo
    var findInfo = new IX.FindInfo();
    findInfo.findByFulltext = new IX.FindByFulltext();
    findInfo.findByFulltext.term =
document.getElementById("searchTerm").value;

    // Setup Callback object
    var cb = new IX.AsyncCallback(Example_receivedResult,
                                        Example_receivedException);
    cb.conn = conn;
    cb.idx = 0;
    cb.max = 10;
    cb.sordZ = new IX.SordZ(CONST.SORD.mbMin, CONST.SORD.mbDocVersion);
    cb.sordTypes = conn.ix().checkoutSordTypes(null,
CONST.SORD_TYPE.mbAllJPG,
                                                    CONST.LOCK.NO);

         // Start serach.
    conn.ix().findFirstSords(findInfo, cb.max, cb.sordZ, cb);
    }
    catch (ex) {
      alert(ex.toString());
    }
    finally {
    }
};

/**
 * This function is called with the results returned by findFirstSords or
findNextSords.
 * It prints the results into the HTML table "searchResults" and calls
findNestSords,
 * if more results are available.
 * @param findResult de.elo.ix.client.FindResult Object
```

```
 */
function Example_receivedResult(findResult) {

    // "this" refers to the AsyncCallback Object passed to findFirstSords.

    // Write results into HTML table
    for (var i = 0; i < findResult.sords.length; i++) {
      Example_printSord(findResult.sords[i], this.sordTypes);
    }

    // If there are more results...
    if (findResult.moreResults) {

        // Invoke findNextSords asynchonously.
        this.idx += findResult.sords.length;
        findResult = this.conn.ix().findNextSords(findResult.searchId,
this.idx,
                                                  this.max, this.sordZ,
this);
    }
    else {

        // No more results: close the find handle.
        this.conn.ix().findClose(findResult.searchId);
    }
}

/**
 * This function is called, if findFirstSords or findNextSords returns an
exception.
 */
function Example_receivedException(ex) {

     // Show message box with error.
     alert(ex.toString());
};

/**
 * This function appends the given sord to the result table.
 * @param sord Sord object
 * @param sordTypes Array of SordType objects
 */
function Example_printSord(sord, sordTypes) {

    // Create a row in the result table
    var tr = document.createElement("TR");
    document.getElementById("searchResults").appendChild(tr);

    // Create the first column
    var td = document.createElement("TD");
    tr.appendChild(td);
```

```
    // The column content
    var item = "";

    // Find the icon related to the given sord
    for (var i = 0; i < sordTypes.length; i++) {
        var sordType = sordTypes[i];
        if (sordType.id == sord.type) {

            // Create an HTML img element
            // and assign the Base64 encoded JPEG icon data to its src
element.
              item += "<img src=\"" +
sordType.icon.data + "\"/>";
            break;
        }
    }

    // Create an anchor for the document.
    // If the anchor is clicked, it shows the document in the iframe next
to the table.
    // The browser cannot display all documents this way. If it cannot do
so, it prompts
    // for downloading the file.
    item += "<a href=\"javascript:Example_showDocument('" +
sord.docVersion.url +
                                                          "')\">" + sord.name
+ "</a>";

    // Set column content
    td.innerHTML = item;

    // Second column with document size or text "Folder"
    td = document.createElement("TD");
    tr.appendChild(td);
    td.innerHTML = sord.docVersion ? sord.docVersion.size : "Folder";
};

/**
 * This function displays the given URL in the iframe next to the result
table.
 */
function Example_showDocument(url) {
  document.getElementById("searchPreview").src = url;
};
<body>
<table>
  <tr>

    <!-- Search form -->
    <td valign="top">
```

```
    <div id="searchForm">
      Search Term:<input id="searchTerm"  type="text" value="ELO Update"/>
      <input id="bnSearch" type="button" value="Search"
                  onclick="Example_doSearchAsync()" />
    </div>

    <!-- Result table -->
    <div style="overflow: scroll; width: 700px; height: 700px;">
    <table border="1">
      <thead>
            <tr>
                  <th>Short description</th>
                  <th>File Size</th>
            </tr>
      </thead>

      <!-- This element will be filled with search results -->
      <tbody id="searchResults">
      </tbody>
    </table>
    </div>
    </td>

    <!-- Preview frame shows the selected document -->
    <td valign="top"><iframe id="searchPreview" src="about:blank"
      width="500" height="700" frameborder="1"></iframe></td>
  </tr>
</table>
</body>
```

Seit ELO 9 kann statt eines Objekts vom Typ de.elo.ix.client.AsyncResult auch ein Funktionsobjekt übergeben werden, das wie folgt definiert ist:

```
            function(result, exception) {
                  if (exception) {
                        // Fehler ausgeben
                  }
                  else {
                        // result verarbeiten;
                  }
            });
```

5.4 Datei Upload

Mithilfe ausgefeilter JavaScript Programmierung auf der Basis des HTML-File-Input Elements ist es möglich, auch Dateien als Dokumente oder Attachments ins Archiv einzuchecken. Derartige Lösungen gibt es in kommerziellen und freien Bibliotheken und manche setzten die Bibliothek jQuery voraus. Der Indexserver JSON-API ist aktuell keine File-Upload-Funktionalität beigelegt, weil sie bisher hauptsächlich im ELO Web Client eingesetzt wird und dieser die Upload-Funktionen von ExtJS, einer kommerziellen JavaScript-Bibliothek, benutzen kann.

Eine einfache freie Bibliothek stellt Andris Valums zur Verfügung. Sie setzt keine anderen Bibliotheken voraus und eignet sich deshalb besonders gut für ein Beispiel.

In die demo.htm von Valums File-Upload-Beispiel ist die JSON-Lib des Indexservers im HTML-head einzubinden:

```
<head>
…
<!-- Include Indexserver JavaScript Library -->
<script type="text/javascript" src="/ix-elo80/EloixClient.js"></script>
```

Daran anschließend folgt der Beispielcode mit dem Ereignishandler Example_onComplete, der nach erfolgreichem Upload aufgerufen wird.

```
<!-- Example code -->
<script type="text/javascript">

// Simplify namespace access
// (all ELOix symbols are member of de.elo.ix.client)
var IX = de.elo.ix.client;

// Indexserver URL
var ixUrl = "/ix-elo80/ix";

// Login credentials
var userName = "Administrator";
var userPwd = "elo";

// Indexserver connection
var _conn;

/**
 * Returns the Indexserver connection.
 * Creates a new connection, when called the first time.
 */
function getConn() {
  if (!_conn) {
    var connFact = new IX.IXConnFactory(ixUrl, "SimpleSearch", "1.0");
    _conn = connFact.create(userName, userPwd);
```

```
  }
  return _conn;
};

function Example_onComplete(id, fileName, responseJSON) {
  try {
    var conn = getConn();
    var CONST = conn.getCONST();

    var ed = conn.ix().createDoc("1", "", "",
CONST.EDIT_INFO.mbSordDocAtt);
    ed.sord.name = fileName;

    ed.document.docs = [ new IX.DocVersion() ];
    ed.document.docs[0].ext = fileName;

         // Indexserver has created an element “uploadResult” in the
response text.
         // It has to be sent back in order to attach the uploaded file
with the
         // document version.
    ed.document.docs[0].uploadResult = responseJSON.uploadResult;

    ed.document = conn.ix().checkinDocEnd(ed.sord, CONST.SORD.mbAll,
                                                ed.document, CONST.LOCK.NO);

    return true;
  }
  catch (ex) {
    alert(ex);
    return false;
  }
}

</script>

</head>
```

Die Funktion createUploader der demo.htm ist wie folgt zu erweitern:

```
…
        function createUploader(){

            var conn = getConn();

         // Create an URL used as the destination for the file upload.
         // Append a parameter in order to distinguish different upload
solutions,
         // because each solution expects its own JSON response format.
            var actionUrl = conn.makeUploadUrl("&solution=valums");
```

```
            var uploader = new qq.FileUploader({
                element: document.getElementById('file-uploader-demo1'),
                action: actionUrl,
                debug: true,
                onComplete: Example_onComplete
            });
        }
…
```

Mit dem Aufruf conn.makeUploadResult wird eine URL zusammengestellt, zu der die Datei hochgeladen wird. Beim Hochladen wird noch kein Dokument oder Attachment erzeugt. Es wird die Datei lediglich, mit einem eindeutigen Namen versehen, ins Temp-Verzeichnis des Servers gelegt. Der Indexservers gibt eine Art ID auf die Datei zurück: das uploadResult. Es ist im obigen Beispielcode im Ereignishandler Example_onComplete im Parameterobjekt responseJSON enthalten und wird dem DocVersion-Objekt vor checkinDocEnd zugewiesen. Auf diese Art wird die hochgeladene Datei mit einer Dokumentversion verbunden.

Weil jede File-Upload-Lösung ihr eigenes Rückgabeformat vom Server erwartet, kann die Antwort in einem JavaScript-Ereignis zusammengestellt werden. Dazu muss eine JavaScript-Datei in den Archivordner „Administration\Indexserver Scripting Base_ALL“ (bzw. …\Servername) eingestellt werden, die eine Funktion namens onFileUploadBuildResponse enthält. Sie wird mit einigen Informationen zur Datei in Form eines DocVersion-Objekts versorgt (Parameter dv). Außerdem werden die wichtigsten Informationen aus der Anfrage im Parameter requestInfo bereitgestellt: HTTP-Header, Cookies und URL-Parameter. Die Funktion erwartet, dass die Ergebnisse in den Parameter responseInfo eingetragen werden. Es sind der Content-Typ und der Ergebnistext einzutragen. Gewöhnlich ist er eine String-Repräsentation eines Java-Objekts (JSON) und muss das vom Indexserver gebildete uploadResult enthalten.

```
importPackage(Packages.de.elo.ix.client);
importPackage(Packages.java.util);

/**
 * Indexserver calls this event after a file has been uploaded via the
JSON-API.
 * https://github.com/blueimp/jQuery-File-Upload/wiki
 * https://github.com/blueimp/jQuery-File-Upload/wiki/Setup
 * @param ec Execution context
 * @param dv DocVersion object with the following valid members:
 * ext, size, guid, uploadResult, physPath.
 * DocVersion.uploadResult has to be returned to the client application.
 * It must be set into its DocVersion.uploadResult before checkinDocEnd can
be called.
 * DocVersion.physPath is set to the location of the uploaded file in the
 * servers temporary directory.
 * @param fileName Name of the uploaded file.
 * @param requestInfo Information of the underlying HttpServletRequest
object.
```

```
 * @param responseInfo  Information for the underlying HttpServletResponse
object.
 */
function onFileUploadBuildResponse(ec, dv, fileName, requestInfo,
responseInfo) {
  log.info("onFileUploadBuildResponse(");
  log.info("dv=" + dv);
  log.info("dv.ext=" + dv.ext);
  log.info("dv.size=" + dv.size);
  log.info("dv.physPath=" + dv.physPath);

  log.info("fileName=" + fileName);

  log.info("requestInfo=" + requestInfo);
  log.info("requestInfo.headers=" + Arrays.toString(requestInfo.headers));
  log.info("requestInfo.cookies=" + Arrays.toString(requestInfo.cookies));
  log.info("requestInfo.requestParams=" +
Arrays.toString(requestInfo.requestParams));

  log.info("responseInfo=" + responseInfo);

  // Find request parameter named “solution”.
  // The function conn.makeUploadUrl("&solution=valums") called in the
browsers
  // JavaScript code has appended this parameter to the action URL.
  var solution = getValueFromKey(requestInfo.requestParams, "solution",
"valums");

  if (solution != "valums") {
    throw new "Unknown File-Upload solution: " + solution;
  }

  // Set the Content-Type header of the response.
  responseInfo.contentType = "text/html";

  // Assign the response text: build a JavaScript object with
  // members “uploadResult” and “success” and convert it into
  // a JSON string.
  var response = { uploadResult : ""+dv.uploadResult, success : true };
  responseInfo.responseString = JSON.stringify( response );

  log.info("responseInfo=" + responseInfo);
  log.info(")onFileUploadBuildResponse");
}

// Helper function to find a value by a given key in an array of KeyValue
objects.
function getValueFromKey(keyValues, key, defaultValue) {
  var value = defaultValue;
  for (var i = 0; i < keyValues.length; i++) {
    if (keyValues[i].key == key) {
```

```
      value = keyValues[i].value;
      break;
    }
  }
  return value;
}
```

5.5 Anmeldung

Zur Anmeldung stehen drei Verfahren zur Auswahl.

5.5.1 Anmeldung mit Benutzername und -kennwort

```
var conn = de.elo.ix.client.IXConnFactory.create(userName, userPassword);
```

Das Kennwort wird hierbei RSA-verschlüsselt versendet.

5.5.2 SSO-Anmeldung: IXConnFactory.createSSO

```
var conn = de.elo.ix.client.IXConnFactory.createSSO();
```

Der Browser übernimmt die Authentifizierung über HTTP-Header zu dem Server, auf dem der Indexserver läuft. Die Prüfung der Benutzerdaten erfolgt unter Windows gegen das ActiveDirectory. Wenn der Benutzer sich dort anmelden kann, erstellt der Indexserver für ihn eine ELO Sitztung. Es kann auch ein vorgeschalteter Apache2 oder IIS (Reverse-Proxy) die Authentifizierung übernehmen. Dann muss der Tomcat, auf dem der Indexserver läuft, mit dem Modul „mod_jk“ angebunden werden.

5.5.3 Übernahme eines bestehenden Tickets

```
var conn = de.elo.ix.client.IXConnFactory.createFromTicket(ticket);
```

Das Ticket kann z. B. von einer Webanwendung in die dynamisch generierte HTML-Seite oder in den JavaScript-Code eingetragen werden.

5.6 Elementselektoren

Elementselektoren geben an, welche Datenelemente einer API-Klasse bei einem Funktionsaufruf gültig sind. In checkinSord gibt der Parameter sordZ beispielsweise an, welche Datenelemente des übergebenen Sord-Objekts geschrieben werden sollen.

In der Java- und .NET-API sind die Elementselektoren 64bit-Integer-Werte. Jedes Bit steht für ein bestimmtes Datenelement. JavaScript-Integer sind aber auf höchstens 32bit beschränkt. Aus diesem Grund wurde eine andere Darstellung in der JSON API gewählt: Elementselektoren werden durch Strings repräsentiert. Sie enthalten eine oder eine Kette von Dezimaldarstellungen von 64bit Werten. Jeder Wert endet dabei mit einem Punkt.

Beispiel: Die Indexwerte eines Dokuments speichert das Element „objKeys“ der Klasse „Sord“. Der zugehörige Elementselektor heißt „mbObjKeys“ und befindet sich in der Konstantenklasse „SordC“. In der Java- und .NET-API ist der Elementselektor definiert als 64bit Integer, in dem das Bit 43 gesetzt ist:

```
public final static long mbObjKeys = (1L << 43);
```

Wenn der Elementselektor über die JSON API abgefragt wird (CONST.SORD.mbObjKeys), erhält man den String „8796093022208.".

Eine Kombination von Elementselektoren kann durch String-Addition (Verketten) erfolgen.

Beispiel: Zusammenstellen eines Elementselektors für das Lesen der Kurzbezeichnung und der Indexwerte eines Objekts.

```
var conn = … // IXConnFactory.create…
var objId = …
var CONST = conn.getCONST();

// Create element selector.
// Constructor computes the bitwise OR operation of the given arguments.
var sordZ = new de.elo.ix.client.SordZ(CONST.SORD.mbName +
CONST.SORD.mbObjKeys);

// Call checkoutSord to read the data.
var sord = conn.ix().checkoutSord(objId, sordZ, CONST.LOCK.NO);
```

Die Verkettung der beiden Elementselektoren hat den Wert „32.8796093022208.".

Die Funktion checkoutSord kann sowohl mit einem Elementselektor vom Typ SordZ als auch mit EditInfoZ aufgerufen werden. Je nach dem gibt sie ein Sord oder ein EditInfo zurück. Auch für die Funktion createSord gibt es eine solche Sonderbehandlung.

Für checkoutDoc oder createDoc gibt es keine Sonderbehandlung und es muss ein EditInfoZ übergeben werden. Dieser Elementselektor nimmt eine Sonderstellung ein, weil er nicht nur ein Bitset sondern auch ein SordZ enthält. Das nächste Beispiel zeigt, wie man einen EditInfoZ zusammenstellt, der die Kurzbezeichnung des Dokuments, die Haftnotizen und die Maskendefinition selektiert.

```
var conn = … // IXConnFactory.create…
var objId = …
var CONST = conn.getCONST();

// Create element selector for EditInfo object
var editZ = new de.elo.ix.client.EditInfoZ(CONST.EDIT_INFO.mbNotes +
                                           CONST.EDIT_INFO.mbMask);
editZ.sordZ = new de.elo.ix.client.SordZ(CONST.SORD.mbName);

// Read the data.
var editInfo = conn.ix().checkoutDoc(objId, "", editZ, CONST.LOCK.NO);
```



Information: Die interne Darstellung der Elementselektoren wurde von ELO 2011 auf ELO 9 geändert. In ELO 2011 besteht ein Elementselektor aus einem Array von Integer-Werten. Daher müssen in den JSON API Programmen alle Stellen geändert werden, wo Elementselektoren kombiniert werden.

5.7 Anwendungsentwicklung, Debugging

Für die Entwicklung von Indexserver Anwendungen mit der JSON API eignen sich sowohl Java-Entwicklungsumgebungen wie bspw. „Eclipse IDE for Java EE Developers „ als auch das „Visual Studio“ von Microsoft.

Das Debugging kann mit den Werkzeugen erfolgen, die in den Webbrowsern integriert sind. Sowohl Chrome, Internet Explorer als auch Firefox bieten komfortable Debugger an. Visual Studio kann im Zusammenspiel mit dem Internet Explorer die Skripte auch in der Entwicklungsumgebung debuggen.

Den Debugger im Chrome findet man unter dem Schraubenschlüssel-Icon über „Tools – Entwicklertools“. Eine Anleitung für den Debugger findet man hier: http://code.google.com/chrome/extensions/tut_debugging.html

Insbesondere während der Entwicklung ist das Cross-Domain-Problem (s. Abschnitt 5.2) von Bedeutung. Es ist also neben der Entwicklungsumgebung auch ein Reverse-Proxy nötig, der wie in Abschnitt 5.8 aufgesetzt werden kann.

5.7.1 Entwicklung mit Eclipse

Für die Entwicklung mit Eclipse unter Windows wird benötigt:

- Aktuelle Java Runtime: www.java.com
- „Eclipse IDE for Java EE Developers“: http://www.eclipse.org/downloads/
  Entpacken in C:\Programme
  (Beim ersten Start nach der Wahl des Workspace auf “Workbench“ klicken, um in die Entwicklungsumgebung zu gelangen.)
- Tomcat 7.0: http://tomcat.apache.org/download-70.cgi, “Binary Distribution”, “x-bit Windows zip”, nicht “Windows Service Installer”
  Entpacken in C:\Programme
- Apache2 Web-Server: http://httpd.apache.org/download.cgi
  “Win32 Binary without crypto (no mod_ssl) (MSI Installer)”
  (Installation erfolgt später wie in Abschnitt 5.8 beschrieben)

Der Tomcat wird im Eclipse wie folgt eingerichtet:

- Window – Preferences – Server – Runtime Environments – Add…
- Apache Tomcat v7.0

- Installationsverzeichnis des Tomcat eingeben
- Finish

Tomcat läuft standardmäßig auf Port 8080. Wenn er geändert werden muss, dann kann dies geschehen über:

- Ansicht „Servers" öffnen: Window – Show View – Other … - Server – Servers
- Doppelklick auf „Apache Tomcat 7" in der Ansicht „Servers"
- Im Abschnitt Ports die Portnummer(n) anpassen
- Speichern: „File – Save"

Als Entwicklungsprojekt eignet sich ein "Dynamic Web Project" besonders gut, weil sich die damit erstellte Anwendung in eine WAR-Datei exportieren lässt, die man einfach auf einem beliebigen, anderen Tomcat einspielen kann.

Ein solches Projekt erstellt man über „File – New … - Project … - Web – Dynamic Web Project". Der Projektname muss gewählt werden. Für weitere Eingaben können die Standardwerte akzeptiert werden.

Im neuen Projekt befindet sich der Ordner „WebContent". In diesem Ordner und seinen Unterordnern können die HTML-Dateien und JavaScripte angelegt werden. Hier sollte als erstes eine Datei namens „index.html" erstellt werden, die standardmäßig als Einstiegspunkt für die Anwendung dient.

Zum Ausführen der Anwendung öffnet man das Kontextmenü auf dem Projekt und startet „Run As… - Run on Server". Daraufhin erscheint ein Auswahlfenster, indem man den Server angegeben kann, auf den man die Anwendung deployen möchte. Wir haben oben nur einen Tomcat-Server installiert und haken die Checkbox „Always use this server…" an, damit der Dialog nicht wieder kommt. Mit „Finish" startet der Server, und im Eclipse wird die Seite index.html angezeigt.

Eclipse verwendet für die Anzeige von Web-Seiten den Internet Explorer als Plugin. Die angesteuerte URL verweist auf den oben eingerichteten Tomcat. Wegen des Cross-Domain-Problems wird im JavaScript aber keine Verbindung zum Indexserver möglich sein.

Hierfür muss der Apache2 noch installiert und als Reverse Proxy konfiguriert werden, wie es in Abschnitt 5.8 beschrieben ist.

Anschließend ruft man die Web-Seite unter der Adresse des Apache2 im Browser auf.

5.8 Reverse-Proxy mit dem Apache2 Webserver

Nur wenige Handgriffe sind nötig, um den Apache2 Webserver als Reverse-Proxy zu konfigurieren, damit wir in der Lage sind, Cross-Domain-Anfragen auszuführen. Sicher ist auch der IIS dafür geeignet. Die Konfiguration des Apache2 macht aber einen einfacheren Eindruck.

Auf der Seite http://httpd.apache.org/download.cgi kann man den aktuellen Apache2 als Anwendung (Binary) herunterladen, beispielsweise das MSI-Paket für Windows.

> **Information:** Bei der Installation erscheint zweimal eine Eingabeaufforderung, die nicht automatisch verschwindet, wenn der Port 80 bereits von einer anderen Anwendung belegt wird. Die Fenster können dann mit der ESC-Taste geschlossen werden. Unabhängig davon gelingt die Installation. Man muss nur vor dem ersten Start in der Konfigurationsdatei die Portnummer ändern („Listen <port>“) oder den Port frei machen.

Die Konfiguration des Webservers erfolgt in der Datei <instdir>/conf/httpd.conf. Es ist eine einfache Textdatei, die mit dem Notepad bearbeitet werden kann. Zeilen, die mit einem „#“ beginnen, sind Kommentare.

Als erstes muss der Apache2 angewiesen werden, die Module „proxy_module“ und „proxy_http_module“ zu laden. Dazu sind die entsprechenden Zeilen „#LoadModule …“ zu aktivieren, d. h. das Kommentarzeichen „#“ muss am Zeilenanfang entfernt werden.

```
…
LoadModule proxy_module modules/mod_proxy.so
…
LoadModule proxy_http_module modules/mod_proxy_http.so
…
```

Nun müssen die Module noch parametrisiert werden. Am Ende der Konfigurationsdatei erfolgt dies mit den folgenden Zeilen, die man noch an die konkreten Gegebenheiten anpassen muss.

```
ProxyRequests Off

<Proxy *>
Order deny,allow
Allow from all
</Proxy>
ProxyPass /ix-elo80 http://srvt02:8084/ix-elo80
ProxyPassReverse /ix-elo80 http://srvt02:8084/ix-elo80
```

Die Optionen „ProxyPass“ und „ProxyPassReverse“ haben jeweils gleiche Parameter. Der erste legt fest, welche URI umgeleitet werden soll. Mit dem zweiten wird angegeben, welcher Server den Aufruf bekommt.

Beispiel: Angenommen, der Apache2 wird mit http://myserver:8000 angesprochen. Dann werden mit obiger Konfiguration Anfragen beginnend mit „/ix-elo80“ an den Server „srvt02“ weitergeleitet. Z. B. wird die Anfrage http://myserver:8000/ix-elo80/ix. weitergeleitet an http://srvt02:8084/ix-elo80/ix. Enthält der Request nicht „/ix-elo80“, erfolgt keine Weiterleitung und der Apache2 sucht den Inhalt in seinem Verzeichnis „<instdir>/htdocs“. Hier könnte beispielsweise die HTML-Datei mit dem JavaScript-Programm liegen.



Information: Wenn der Apache2 zu einem Apache Tomcat weiterleiten soll, kann dies auch über das Modul „mod_jk“ geschehen. Hierbei kommt ein spezielles und optimiertes Protokoll zum Einsatz. Ferner bietet es auch die Möglichkeit, mehrere Tomcats in einer Lastverteilung zusammenzuschalten. Die Konfiguration ist nur geringfügig aufwendiger und wird in vielen Anleitungen im Internet beschrieben.

5.9 Beispiele

In der Indexserver_Programming.zip befinden sich einige Beispiele zur Verwendung der JSON-API im Ordner /doc/JSON API/. Zum Ausprobieren auf einem Tomcat, kopiert man das komplette Verzeichnis in ein Unterverzeichnis unter webapps. Das Unterverzeichnis sollte der Einfachheit halber kein Leerzeichen enthalten, bspw. „JSONAPI“. Die Beispielseite GetDoc.html kann dann aufgerufen werden über: http://server:port/JSONAPI/GetDoc.html.

In den Beispielen ist die Indexserver URL fest verdrahtet in der Variablen „ixUrl“ und muss an die Umgebung angepasst werden. Ebenso sind Benutzername und Kennwort vorgegeben. Die Variablen hierzu lauten „userName“ und „userPwd“.

6 ELOix Lastverteilung

Dieses Kapitel beschreibt den Betrieb mehrerer Indexserver für dasselbe Archiv und die Lastverteilung mittels Apache2 Webserver.

6.1 Überblick

In ELOenterprise-Installationen mit großen Benutzerzahlen kann es sinnvoll sein, mehrere Indexserver auf verschiedene Server verteilt zu betreiben. Damit für alle Client-Anwendungen dieselbe Indexserver-URL gilt, schaltet man einen Proxy-Dienst zwischen Client und Server, der die Anfragen gleichmäßig verteilt.

Der Apache2 eignet sich besonders gut für diese Aufgabe wegen seiner einfachen Handhabung und der Möglichkeit, die Kommunikation zu den Servern über ein optimiertes Protokoll abzuhandeln.

6.2 Zusätzlichen Indexserver installieren

Voraussetzung ist eine Java 7.0 Installation und ein Tomcat 7.0.30 oder neuer.

Ein neuer Indexserver kann zwar auch im laufenden Betrieb ergänzt werden. Unter hoher Last kann aber das nötige „Reload" zu Out-Of-Memory-Fehlern in den Anwendungen führen. Deshalb sollte ein Zeitraum gewählt werden, in dem möglichst wenige Indexserver-Anfragen zu erwarten sind.

	Benennung: das Verzeichnis der ELOenterprise Installation wird im Folgenden **instdir1** genannt.	ELOenterprise Verzeichnis: _____
	Benennung: das Verzeichnis der Tomcat-Installation für die ELOenterprise Installation wird im Folgenden **tomdir1** genannt.	tomdir1 Verzeichnis: _____
	Benennung: der Indexserver der ELOenterprise Installation benötigt einen (unter allen Indexservern des Archivs) eindeutigen Namen. Wenn nicht explizit ein Name angegeben wurde, ist es der Computername.	ixid1: _____
	Benennung: das Verzeichnis der Tomcat-Installation auf dem Zielserver wird im Folgenden **tomdir2** genannt.	tomdir2 Verzeichnis: _____
	Tomcat auf dem Zielserver **stoppen.**	
	Neues Verzeichnis für die benötigten Dateien auf dem Zielserver erstellen. Z. B. „C:\ELOenterprise". Es wird im Folgenden **instdir2** genannt.	Zielverzeichnis: _____
	Verzeichnis **instdir2\webapps** anlegen	
	Verzeichnis **instdir2\config\ix-Archiv** anlegen	

	Verzeichnis **instdir2\data\ix-Archiv** anlegen	
	Verzeichnis **instdir2\logs** anlegen	
	Download der **Indexserver.zip** aus dem Supportweb.	
	Datei **eloix.war** aus der Indexserver.zip in das Verzeichnis **instdir2\webapps** kopieren	
	Alle Dateien aus dem Verzeichnis der ELOenterprise Installation **instdir1\config\ix-Archiv** kopieren nach: instdir2\config\ix-Archiv	
	Datei **instdir2\config\ix-Archiv\config.xml** editieren: Neuen entry-Eintrag hinzufügen: **<entry key="ixid">ixid2</entry>** entry-Eintrag **„jdbcurl"** kontrollieren (localhost wird i.a. nicht funktionieren) „ixid2" ist ein eindeutiger Name unter allen Indexserver-Instanzen des Archivs.	**ixid2**: _____
	Datei **instdir2\config\ix-Archiv\log4j.properties** editieren: **log4j.appender.FI.File=instdir2\logs\ix-Archiv.log**	
	Kopieren (einer) der folgenden Dateien (JDBC-Treiber) aus dem Verzeichnis **tomdir1\lib** ins Verzeichnis **tomdir2\lib** **sqljdbc4.jar (für Microsoft SQL Server)** **ojdbc6.jar (für ORACLE)** **db2jcc.jar (für DB2)**	
	Connector-Einstellungen der server.xml auf den neuen Tomcat übertragen.	

	Kopieren der Benutzerdefinitionen für die Tomcat-Administration:
	Kopiere
	`tomdir1\conf\tomcat-users.xml`
	nach
	`tomdir2\conf\tomcat-users.xml`	

	Erstellen der XML-Datei
	tomdir2\conf\Catalina\localhost\ix-Archiv.xml
	mit dem folgenden Inhalt:
	`<?xml version='1.0' encoding='UTF-8'?>`
	`<Context`
	`  path="/ix-Archiv"`
	`docBase="instdir1\webapps\eloix.war"`
	`  unpackWAR="false">`
	
	`  <Environment name="webappconfigdir"`
	`  value="instdir1\config\ix-Archiv"`
	`  type="java.lang.String"`
	`  override="false"/>`
	
	`</Context>`	

	Tomcat auf dem Zielserver noch **nicht** starten.	

6.2.1 Konfiguration

	Statusseite des Indexservers der ELOenterprise-Installation aufrufen (**nicht** die Statusseite des gerade installierten)	
	Link „Configure Options“ klicken und mit dem Tomcat Konto anmelden.	
	In der letzten Zeile der Optionentabelle nacheinander die folgenden Einträge hinzufügen (Klick auf „Add“): **Instance** / **Name** / **Value** **ixid2** / `configDir` / **instdir2\data\ix-Archiv** **ixid2** / `privateUrlBase` / **http://tomcat2:port/ix-Archiv/ix** Wenn für den Indexserver der ELOenterprise Installation (auf tomdir1 mit ixid1) noch keine privateUrlBase angegeben wurde, dann muss dies jetzt geschehen: **Instance** / **Name** / **Value** **ixid1** / `privateUrlBase` / **http://tomcat1:port/ix-Archiv/ix**	

	Information: Über die privateUrlBase nehmen die Indexserver gegenseitig Kontakt auf und gleichen Ihre Cache-Daten ab. Die Einstellung darf also nicht auf den Load-Balancer verweisen.	
	Tomcat auf dem Zielserver starten.	
	Alle Indexserver des Archivs jeweils auf ihrer Seite „Configure Options“ mit der Schaltfläche „Reload“ neu starten (Offene Sitzungen gehen nicht verloren).	

6.3 Apache2 als Lastverteilung

Während der Installation und Konfiguration wird es nötig sein, den ELOenterprise Server zu stoppen.

6.3.1 Apache2 installieren

	Laden Sie dazu den Apache Webserver unter folgender Adresse herunter: http://httpd.apache.org/download.cgi	
	Installieren Sie den Server. Das Zielverzeichnis wird im Folgenden **apachedir** genannt.	**apachedir:** _____
	Laden Sie den Apache Tomcat Connector (mod_jk) unter folgender Adresse herunter: http://tomcat.apache.org/download-connectors.cgi **Information:** Nutzen Sie den Link Binary Releases auf der Downloadseite, um die zum Apache2 passende Versionen zu finden: z. B. „/windows/tomcat-connectors….-httpd-2.4.x.zip“	
	Kopieren Sie die „**mod_jk.so**“ aus der „tomcat-connectors…zip“ in das Verzeichnis **apachedir\modules**.	

6.3.2 Apache2 konfigurieren

	Bearbeiten Sie die Datei **httpd.conf** im Verzeichnis **apachedir\conf** des Apache2. Ergänzen Sie am Ende: **Include conf/tomcat_jk.conf**	
	Erstellen Sie die Datei **tomcat_jk.conf** mit folgendem Inhalt: **#load the JK modul** LoadModule jk_module modules/mod_jk.so # include Tomcat-Workers file JkWorkersFile conf/workers.properties # Logfile, level und format JkLogFile logs/mod_jk.log # Log level [debug/error/info] JkLogLevel info JkLogStampFormat "[%a %b %d %H:%M:%S %Y] " # jk shared memory location JkShmFile logs/mod_jk.shm # redirect /webapp to the tomcat loadbalancer JkMount /ix-Archiv loadbalancer JkMount /ix-Archiv/* loadbalancer	
	Erstellen Sie die Datei **workers.properties** im Verzeichnis **apachedir\conf** mit folgendem Inhalt: **worker.list=loadbalancer** worker.tomcat1.port=8009 worker.tomcat1.host=server1 worker.tomcat1.type=ajp13 worker.tomcat1.lbfactor=1 worker.tomcat2.port=8009 worker.tomcat2.host=server2 worker.tomcat2.type=ajp13 worker.tomcat2.lbfactor=1	

	worker.loadbalancer.type=lb worker.loadbalancer.balance_workers=tomcat1,tomcat2 worker.loadbalancer.method=Busyness worker.loadbalancer.sticky_session=true **Information:** Die Portnummer für worker.*.port entnehmen Sie der server.xml des entsprechenden Tomcat im Connector-Tag. Beispielsweise: : **<Connector port="8009" protocol="AJP/1.3"…>** Die Bezeichnung **tomcat1** (oder **tomcat2**) benennt einen Tomcat-Server. Sie muss später in der **server.xml** des Tomcat eingetragen werden, s. 6.3.3. Für einen weiteren Tomcat-Server ergänzen Sie einen Block **worker.tomcat3.*** und die Liste bei **worker.loadbalancer.balance_workers=tomcat1,tomcat2,tomcat3**	
	Starten Sie den Apache2 Server neu.	

6.3.3 Tomcat Konfiguration ergänzen

Die Lastverteilung muss dafür sorgen, dass Anfragen eines Clients immer zum selben Server gelangen. Den richtigen Server ermittelt sie aus dem HTTP Cookie JSESSIONID, dass beim ersten Aufruf des Clients vom Server vergeben wird und vom Client bei folgenden Anfragen immer mitgesendet wird. Die Lastverteilung erwartet, dass der Servername des Tomcat (hier also tomcat1 oder tomcat2) an die JSESSIONID angehängt ist. Dazu muss der Tomcat seinen Namen kennen. Die folgenden Schritte beschreiben die hierzu nötige Erweiterung der Tomcat-Konfiguration.

	Editieren Sie die Datei <tomdir1>\conf\server.xml	
	Suchen Sie das XML-Tag **<Engine name="Catalina" …>** Ergänzen Sie im Tag das Attribut jvmRoute und weisen Sie ihm den Tomcat-Namen tomcat1 zu. <Engine name="Catalina" jvmRoute="tomcat1" …> **Information:** Es gibt möglicherweise mehrere dieser Tags. Achten Sie darauf, dass sie das **nicht-auskommentierte** verändern.	
	Starten Sie den Tomcat neu.	

	Führen Sie die Schritte auch mit dem zweiten Tomcat in **tomdir2** durch. Verwenden Sie als Tomcat-Namen **tomcat2**.	

6.3.4 Indexserver Option ixUrlBase setzen

	Statusseite eines Indexservers aufrufen.	
	Link „Configure Options“ klicken und mit dem Tomcat Konto anmelden.	
	In der Optionentabelle die Optionen **ixUrlBase** wie folgt ergänzen: **Instance** / **Name** / **Value** **ixid1** / ixUrlBase / **http://apache:port/ix-Archiv/ix** **Ixid2** / ixUrlBase / **http://apache:port/ix-Archiv/ix** Hierbei ist **apache:port** der Servername und Port unter dem der Apache2 extern erreichbar ist. Über diese URL werden die Client-Anwendungen auf Dokumente zugreifen.	

6.3.5 Weitere Informationen

Weitere Informationen zum Apache2 Server und Tomcat Connector finden Sie auf den Web-Seiten der Apache Group.

http://httpd.apache.org/

http://tomcat.apache.org/connectors-doc/index.html

7 Zugriff auf ein ELO Archiv über WebDAV

ELO WebDAV (IxWebDav) ist die Implementierung der Standardschnittstelle WebDAV, um die Struktur eines ELO Archivbaums als Netzlaufwerk darzustellen.

Wie ELO FS beinhaltet ELO WebDAV Funktionen zum Auschecken von Dokumenten, zum Einchecken von neuen Versionen und zur Bearbeitung von Verschlagwortungsinformationen (nur bei einigen WebDAV-Clients möglich). Ergänzend können Ordner und Dokumente gelöscht, kopiert und verschoben werden. ELO WebDAV bietet die Funktion zum Sperren von Dateien und Ordner (wenn einen Ordner gesperrt ist, kann man keine Datei von diesem Ordner löschen und keine neue Dateien in diesem Ordner hinzufügen). Allerdings unterstützen wenige WebDAV-Clients (wie BitKinex, WebDrive, CrossFTP) diese Features.

Im Gegensatz zu ELO FS ist der Zugriff über ELO WebDAV auf das ELO Archiv komplett von Betriebssystemen unabhängig. WebDAV ist in Windows, Mac OS X und Linux fest in das Betriebssystem integriert. Außerdem können mobile Geräte wie iPhone, iPad, Android-Geräte, Blackberries oder Windows-Mobiltelefone über WebDAV auf ELO Archive zugreifen.

Ein weiterer Unterschied zu ELO FS besteht darin, dass z. B. Microsoft-Office-Dokumente nicht explizit aus- und wieder eingecheckt werden müssen. Dies geschieht automatisch durch die WebDAV-Client-Implementierung in den Office-Programmen bzw. in Windows.

7.1 ELO WebDAV –Server-Adresse

ELO WebDAV ist bereits in den ELO Indexserver integriert. Sie erreichen das ELO WebDAV-Verzeichnis über den Browser. Geben Sie dazu die ELO WebDAV-Adresse nach folgendem Schema ein:

```
http://<Servername>:<Port>/ix-<Archivname>/ixwebdav
```

7.1.1 Verbindung mit vollqualifiziertem Domänennamen

Wenn Sie den WebDAV-Server über den vollqualifizierten Domänennamen (Fully-Qualified Domain Name, FQDN) verbinden wollen, müssen Sie den entsprechende Server in die Zone *Lokales Intranet* aufnehmen:

Öffnen Sie den Konfigurationsdialog *Internetoptionen*.

Wechseln Sie auf die Registerkarte *Sicherheit*.

Klicken Sie auf die Schaltfläche *Sites*.

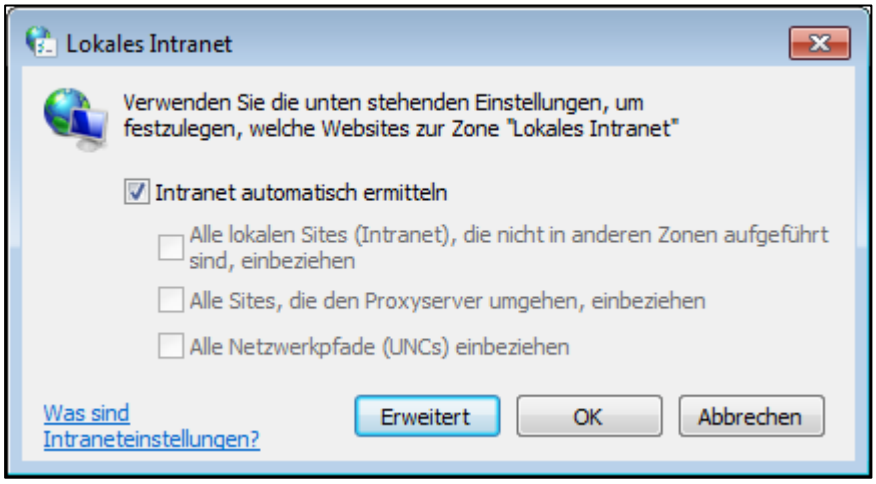


Abb. 10: Zone ‚Lokales Intranet' bearbeiten

Der Dialog *Lokales Intranet* erscheint.

Die Einstellung *Intranet automatisch ermitteln* muss aktiviert sein.

Klicken Sie auf *Erweitert*.

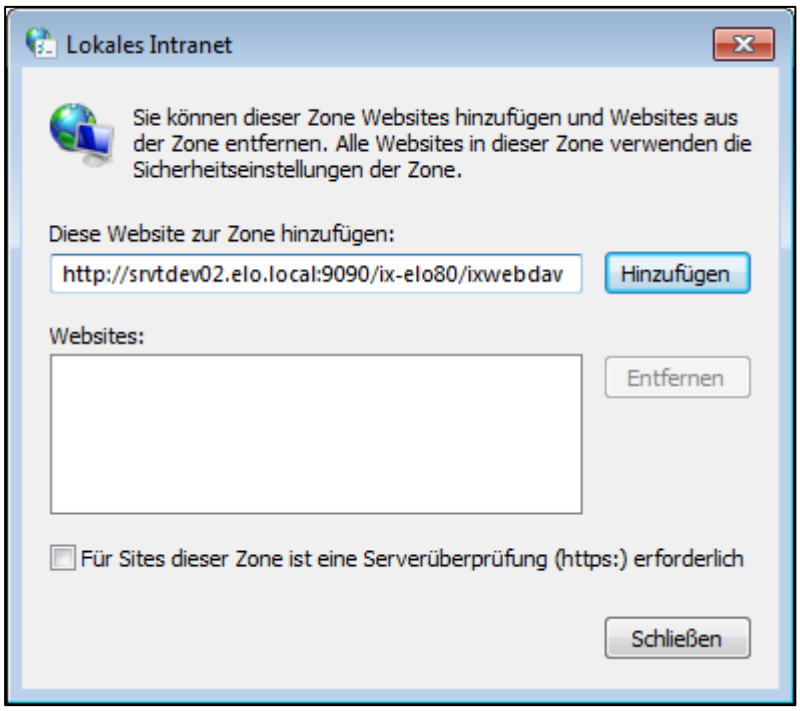


Abb. 11: Vollqualifizierte URL hinzufügen

Ein weiterer Dialog *Lokales Intranet* erscheint.

Tragen Sie die vollqualifizierte URL des Servers ein.

Folgende Beispiele verdeutlichen den Unterschied zwischen einer normalen URL und einer vollqualifizierten URL:

- Normale URL: http://srvtdev02:9090/ix-elo80/ixwebdav
- Vollqualifizierte URL: http://srvtdev02.elo.local:9090/ix-elo80/ixwebdav

Klicken Sie auf *Hinzufügen*.

Schließen Sie den Dialog.

Schließen Sie die weiteren Dialoge mit *OK*.

7.2 ELO WebDAV unter Windows verwenden

Unter Windows wird die NTLM-Authentifizierung verwendet. D. h. Passwörter werden nur verschlüsselt an den Server übertragen.

7.2.1 Allgemeiner Hinweis zur Verwendung unter Windows

Mit Standardeinstellungen lassen sich über WebDAV maximal Dateien bis 47 MB übertragen.

Um den eine größere Übertragungsrate zu ermöglichen, passen Sie in der Registry den Schlüssel *FileSizeLimitInBytes* an.

Sie finden den Schlüssel unter:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\WebClient\Parameters
```

Sie können hier Werte bis zu 4 GB (4294967295 Byte) eintragen.

Starten Sie anschließend den WebClient-Dienst neu, damit die Änderungen wirksam werden.

7.2.2 Mit Windows Explorer als Netzlaufwerk verbinden (ab Windows Vista)

Um ELO WebDAV als Netzlaufwerk unter Windows (Vista oder höher) mit dem ELO WebDAV zu verbinden, rufen Sie zunächst den Explorer auf und klicken dann mit der rechten Maustaste auf *Computer* oder *Netzwerk*.

Im Kontextmenü klicken Sie auf den Menübefehl *Netzlaufwerk verbinden*.

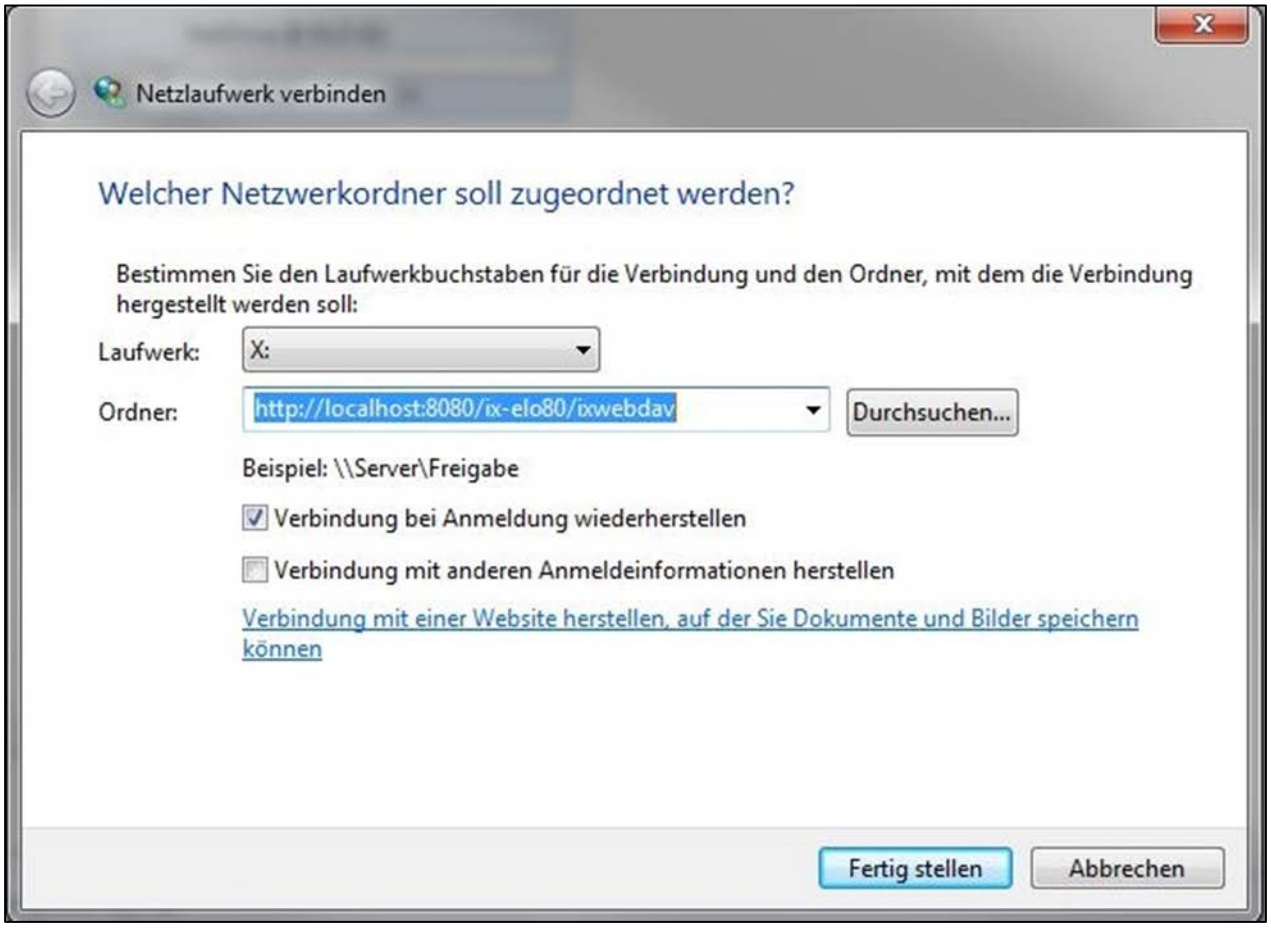


Abb. 12: Netzwerklaufwerk verbinden unter Windows

Wählen Sie einen entsprechend freien Laufwerksbuchstaben und geben unter *Ordner* die Adresse zu Ihren ELO WebDAV ein.

Klicken Sie auf *Fertig stellen*.

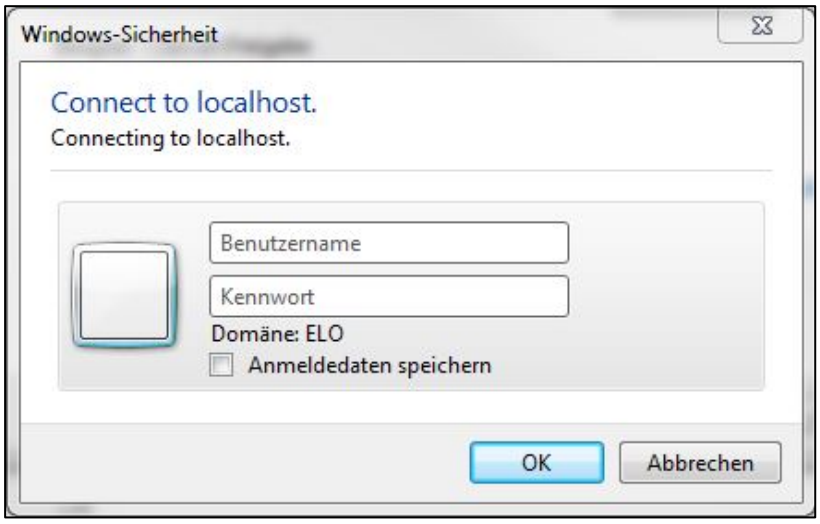


Abb. 13: Login ohne Active Directory

Gehört der angemeldete Windows Benutzer zum dem Active Directory und ist in ELO vorhanden, wird die NTLM-Authentifizierung automatisch angewendet und keine Anmeldeinformationen sind einzugeben. Sonst wird man Dialog Windows-Sicherheit nach Username und Password gefragt. Autorisieren Sie sich mit den ELO Benutzer-Daten.



Beachten Sie: In diesem Fall wird das Basic-Authentifizierungsverfahren verwendet, und das Password wird als Klartext an den Server übertragen. Sie können die Datenübertragung durch SSL sichern.

Entsprechend sollten Sie in der Registry den Wert des Schlüssels *BasicAuthLevel* auf "2" (dezimal) ändern.

Sie finden den Schlüssel unter:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\WebClient\Parameters
```



Beachten Sie: Beim Kopieren von Dokumenten geht die Verschlagwortung verloren.

Der Grund dafür ist, dass der Explorer den PUT-Befehl mit dem Zielpfad und dem Inhalt zum ELOix sendet. Es ist daher unmöglich festzustellen, von welchem Sord der Inhalt kommt, da die Sord-ID fehlt.

7.2.2.1 Langsame Verbindung zum WebDAV-Laufwerk

Wenn die Verbindung zum WebDAV-Laufwerk und Windows Vista oder höher nur mit niedriger Geschwindigkeit erfolgt, beachten Sie folgende Schritte:

Öffnen Sie den Konfigurationsdialog *Internetoptionen*.

Wechseln Sie auf die Registerkarte *Verbindung*.

Klicken Sie auf die Schaltfläche *LAN-Einstellungen*.

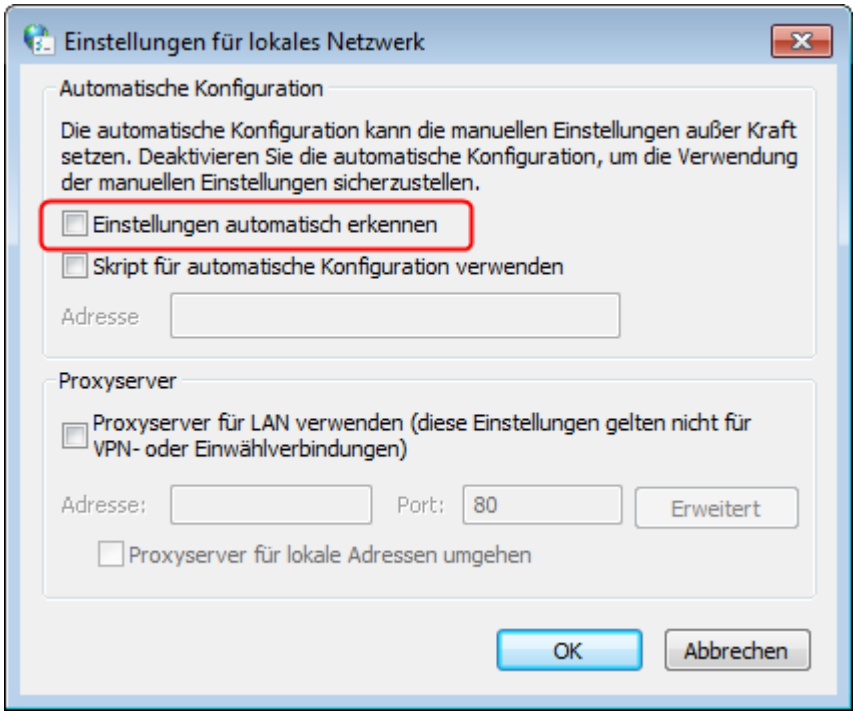


Abb. 14: Option für die lokalen Netzwerkeinstellungen ändern

Der Dialog Einstellungen für lokales Netzwerk erscheint.

Deaktivieren Sie die Option *Einstellungen automatisch erkennen.*

Bestätigen Sie mit *OK*.

Schließen Sie die *Internetoptionen* mit *OK*.

7.2.3 Mit Windows XP

Öffnen Sie den Arbeitsplatz, klicken Sie auf *Extras* > *Netzlaufwerk verbinden*.

Wählen Sie den gewünschten Laufwerksbuchstaben aus und klicken Sie auf den Link *Onlinespeicherplatz anfordern oder mit einem Netzwerkserver verbinden*.

Der Windows-Assistent zum Hinzufügen von Netzwerkressourcen startet. Klicken Sie auf *Weiter*.

Wählen *Sie Eine andere Netzwerkressource auswählen* aus und klicken Sie auf *Weiter*.

Geben Sie die Adresse des ELO WebDAVs ein und klicken Sie auf *Weiter*.

Geben Sie einen Namen für die Netzwerkadresse ein.

Klicken Sie auf *Fertig stellen*.

7.2.4 Verwendung des WebDAV Redirectors

Unter folgendem Link finden Sie die Informationen über die Verwendung von WebDAV über den WebDAV Redirector (Unter Windows 7, Windows 8, Windows Server 2008 oder Windows Server 2012):

http://www.iis.net/learn/publish/using-webdav/using-the-webdav-redirector



Information:

- Für Windows Server ab Version 2008 muss Desktop Experience installiert werden
- Wenn Sie die Fehlermeldung "System Fehler 67 aufgetreten. Der Netzwerkname wurde nicht gefunden" beim Einbinden des ELO WebDAV als Netzlaufwerk bekommen, bitte prüfen Sie ob der Windows-Dienst WebClient läuft und ob Desktop Experience installiert ist.

7.2.5 Mit der Kommandozeile

Um ELO WebDAV zu einem Netzlaufwerk zu mappen, nutzen Sie in der Windows-Kommandozeile den Befehl:

```
net use Z: \\<Host-Name>:<Port>/ix-<Archivname>/ixwebdav/persistent:yes
```

Über den Parameter `/persistent` können Sie festlegen, ob das Netzlaufwerk auch nach einem Neustart wieder verfügbar sein soll. Setzt man hier den Wert `no`, wird die Verbindung nicht mehr wiederhergestellt.

7.2.6 Mit NetDrive

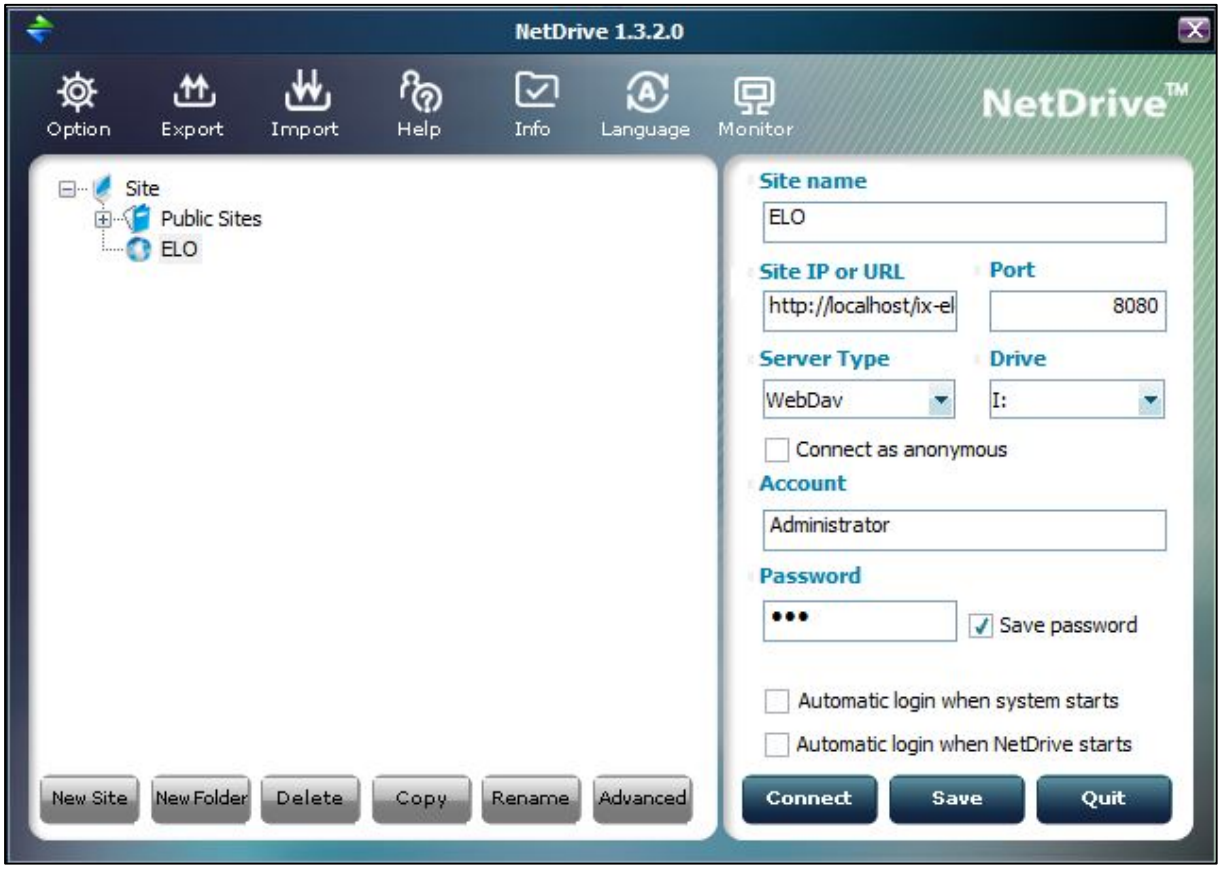


Abb. 15: Zugang über NetDrive

Klicken Sie auf *New Site* und geben Sie einen Namen für die Verbindung ein.

Geben Sie bei *Site IP or URL* die Adresse zum ELO WebDAV ein.

Wählen Sie im Drop-Down-Menü *Server Type* den Eintrag *WebDav* aus.

Tragen Sie in das Feld *Port* den Port des Servers ein

Wählen Sie den Laufwerksbuchstaben aus, den Sie verwenden möchten.

Geben Sie Ihre ELO Zugangsdaten bei *Account* und *Password* ein.

Um SSL-Verschlüsselung zu aktivieren, klicken Sie auf *Advanced* und aktivieren Sie *Use HTTPS*.

Um SSL-Verschlüsselung zu benutzen, wechseln Sie den Port von `8080` auf `443`

Um die Verbindung zu speichern, klicken Sie auf *Save*.

Um die Verbindung herzustellen, klicken Sie auf *Connect*.

7.2.7 Mit BitKinex

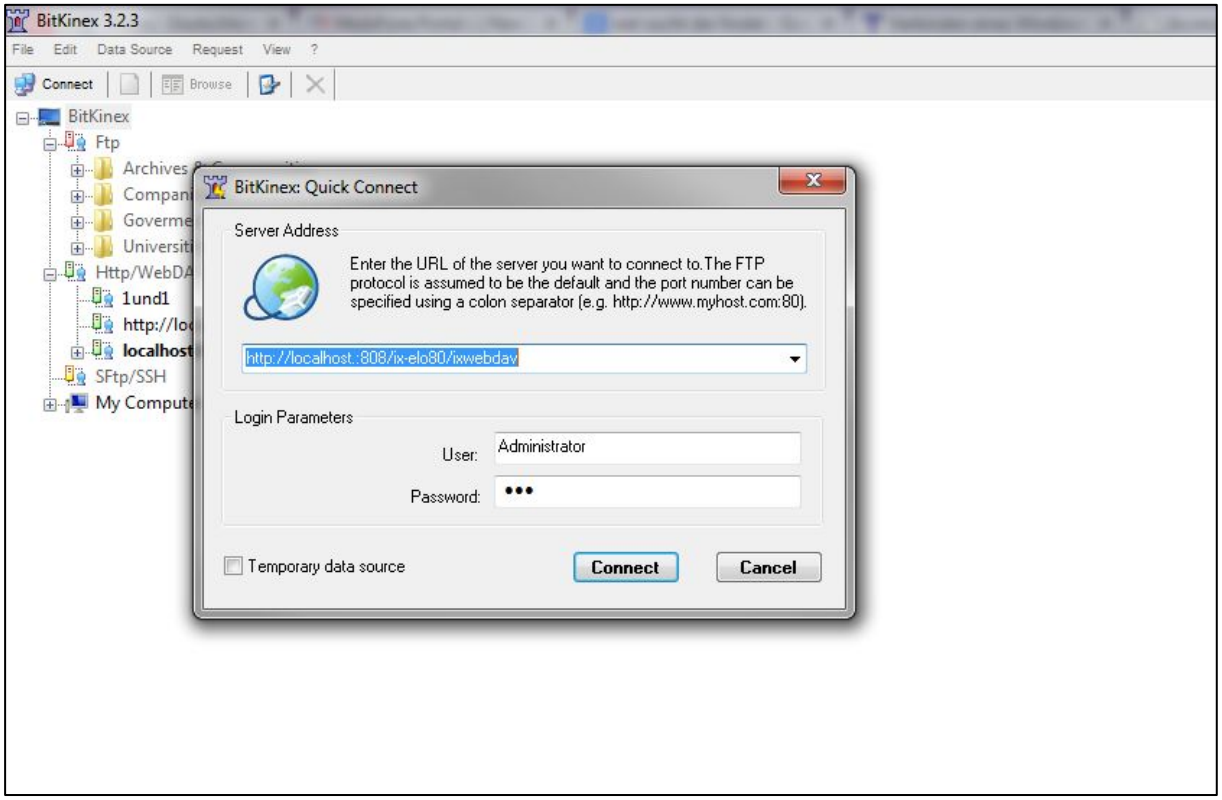


Abb. 16: Zugang über BitKinex

Im Gegensatz zum Windows Explorer und NetDrive unterstützt BitKinex Verbindung mit WebDAV als Netzlaufwerk nicht, sondern verwendet eine eigene Oberfläche. Allerdings bietet BitKinex umfangreiche Features wie das Sperren von Ordner und Dateien, Verschiebung und Kopieren von Dateien im Server ohne Herunterladen von Dateien.

Um BitKinex mit ELO WebDAV zu verbinden, klicken Sie auf *Connect* oder *Quick Connect* im Menü *File*. Geben Sie die Internetadresse zum ELO WebDAV und die ELO Benutzerdaten ein.

Dann klicken Sie auf *Connect*.

7.2.8 WebDAV unter Microsoft Office Anwendungen verwenden

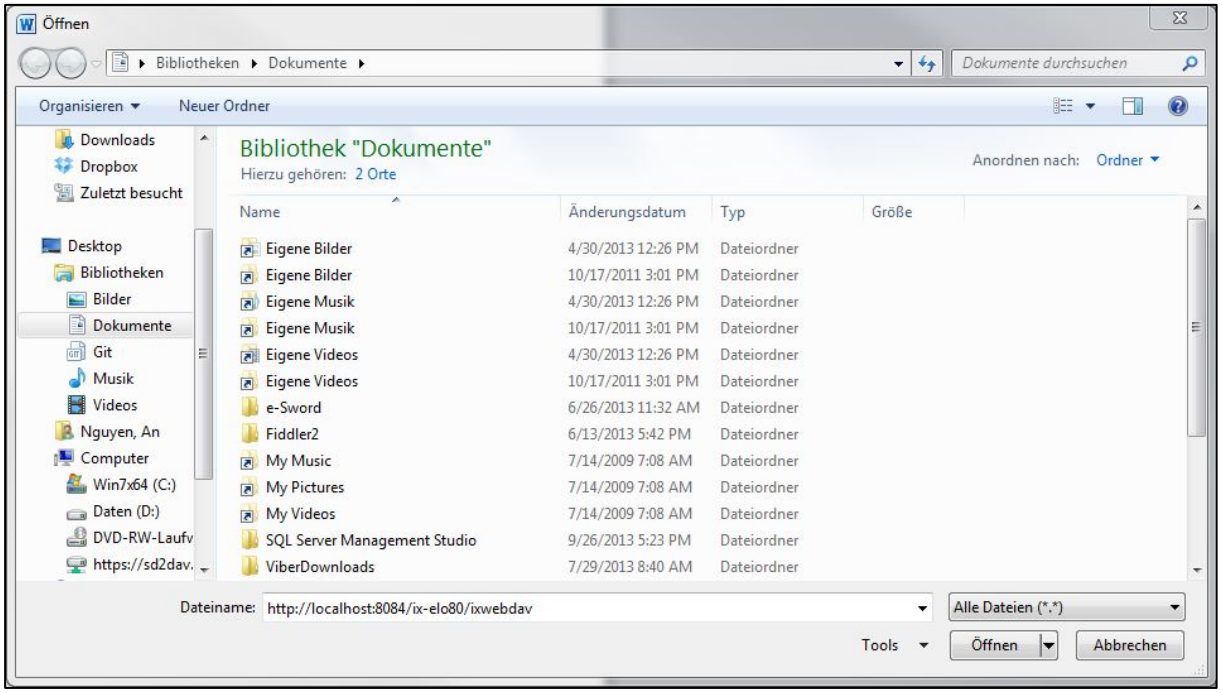


Abb. 17: WebDAV-Anbindung über Microsoft Word

Microsoft Office verwendet einen eigenen WebDAV-Client. Deshalb können Sie ein ELO Archiv in Office-Anwendungen wie Microsoft Word, Microsoft Excel oder Microsoft PowerPoint öffnen, ohne ELO WebDAV mit einem Netzlaufwerk verbinden zu müssen.

Geben Sie im Dialog *Öffnen* die Internetadresse als Dateiname ein und bestätigen Sie die Eingabe mit der Eingabe-Taste.

Beachten Sie: Klicken Sie nicht auf die Schaltfläche *OK*.

7.2.8.1 Allgemeiner Hinweis zur Verwendung unter Microsoft Office Anwendungen

Mit Standardeinstellungen öffnen einige Microsoft-Office-Programme Dokumente, die per Hyperlink aufgerufen werden, nur mit Lesezugriff. Entsprechend sollten Sie in der Registry den Wert des Schlüssels OpenDocumentsReadWriteWhileBrowsing auf "1" (DWORD) ändern.

Sie finden den Schlüssel unter:

HKEY_CURRENT_USER\Software\Microsoft\Office\<version>\Common\Internet

Mehr dazu biete folgende Webseite eine Anleitung: http://www.webdavsystem.com/server/documentation/ms_office_read_only.

Falls beim Öffnen von Office-Dokumenten jedes Mal ein Anmeldedialog angezeigt wird, sind folgende Einstellungen zu ändern.

- aktivieren Sie die Option *Automatische Anmeldung mit aktuellem Benutzernamen und Kennwort* (zu finden über: *Internetoptionen > Sicherheit > Lokales Intranet > Stufe anpassen > Benutzerauthentifizierung > Anmeldung*)
- der ELOix-Server muss in der Intranet-Zone stehen. Sie benötigen 2 Einträge dafür:
    - Einen mit FQN. Z. B.: (http://srvtdev.elo.local)
    - Einen ohne FQN (http://srvtdev)
- Erstellen Sie in der Windows-Registry den Eintrag (REG_MULTI_SZ) mit dem Namen *AuthForwardServerList* unter:

```
HKEY_LOCAL_MASCHINE\SYSTEM\CurrentControlSet\Services\WebClient\Parameters
```

    - Als Wert verwenden Sie den Hostnamen des IX-Servers als Wert setzen.
    - Sie benötigen 2 Einträge dafür,
        - Einen mit FQN. Z. B.: (http://srvtdev.elo.local)
        - Einen ohne FQN (http://srvtdev)

7.2.8.2 Objekte aus Archiv in MS Office Anwendungen einbetten

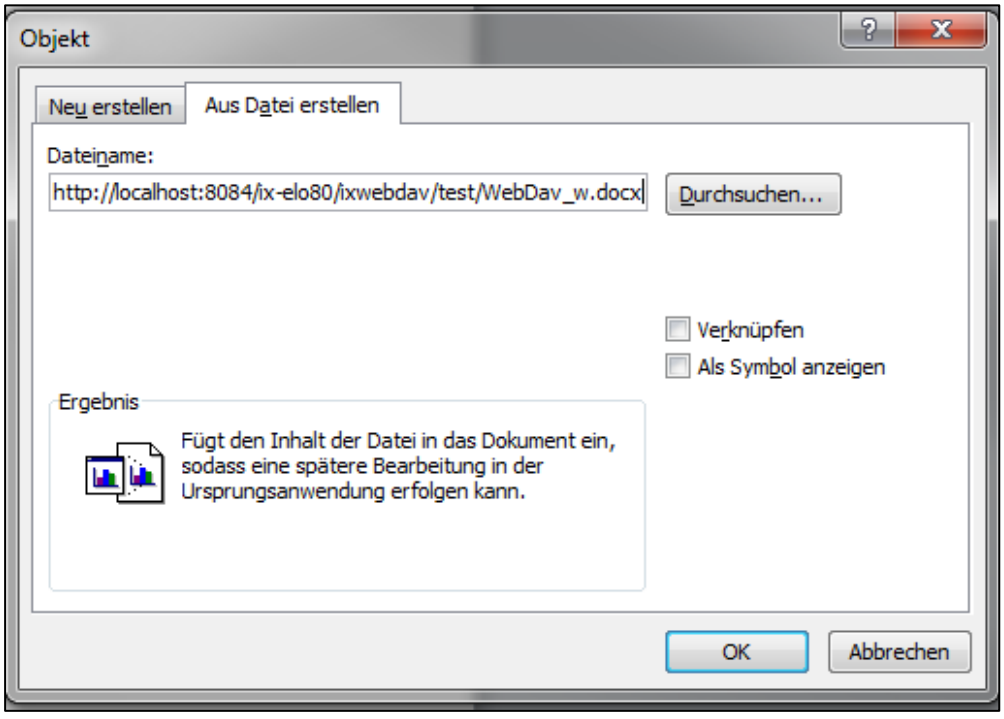


Abb. 18: WebDAV-Objekt einfügen in einer Microsoft-Office-Anwendung

Über ELO WebDAV lassen sich Dateien aus dem ELO Archiv auch als Objekt einbetten.

7.3 WebDAV unter Linux verwenden

In Linux wird das Authentifizierungsverfahren NTLM nicht verwendet. Stattdessen kommt das Basic-Authentication-Verfahren zum Einsatz. Deshalb ist es sehr empfehlenswert, dass Sie die Datenübertragung durch SSL sichern.

7.3.1 Davfs2 (Kommandozeile)

Nutzen Sie folgenden Befehl:

```
sudo mount.davfs -o userid=<LINUX-USER> http://<Servername>:<Port>/ix-
<Archivname>/ixwebdav/mnt/<Mount-Point>
```

Ersetzen Sie die Platzhalter.

7.3.2 Gnome/Nautilus

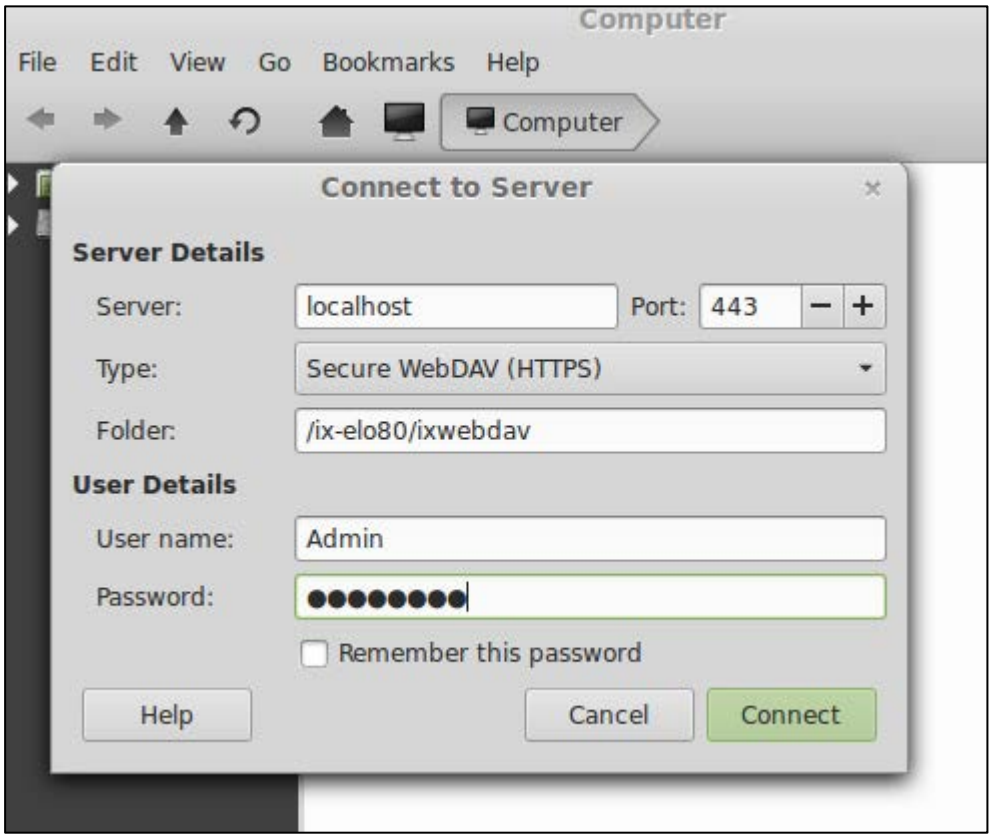


Abb. 19: Einrichtung über Nautilus

In Nautilus klicken Sie auf *Orte* > *Mit Server verbinden* oder auf im Menü *File* auf *Mit dem Server verbinden*.

7.3.3 KDE/Dolphin

Öffnen Sie die Adresse zum ELO WebDAV mit Dolphin oder Konqueror.

7.4 WebDAV unter MAC OS X verwenden

Unter MAC OS X wird das Authentifizierungsverfahren NTLM nicht verwendet. Stattdessen kommt das Basic-Authentication-Verfahren zum Einsatz. Deshalb ist es sehr empfehlenswert, dass Sie die Datenübertragung durch SSL sichern.

7.4.1 Kommandozeile

Nutzen Sie folgenden Befehl:

```
mount -t webdav http://<Servername>:<Port>/ix-<Archivname>/ixwebdav/<Mount-
Point >
```

7.4.2 Finder

Klicken Sie auf im Finder auf *Gehe zu* > *Mit Server verbinden...*

Geben Sie bei *Serveradresse* die Internetadresse zum ELO WebDAV ein.

Klicken Sie auf *Verbinden.*



Information: Während der Bearbeitung unter Finder werden viele temporäre Dateien erzeugt, im Archiv hochgeladen und danach wieder gelöscht. Außerdem werden die Dateien auch im Root-Ordner von ELO erzeugt. Deshalb muss der angemeldete Benutzer Schreibrechte für den Root-Ordner besitzen.

7.4.3 Cyberduck

Der Gratis Tool Cyberduck erzeugt, anders als der Finder, keine temporäre Dateien.

Nach der Installation von Cyberduck, starten Sie Cyberduck von Ihrem Dock oder vom Programmordner. Klicken Sie auf *Neue Verbindung*.

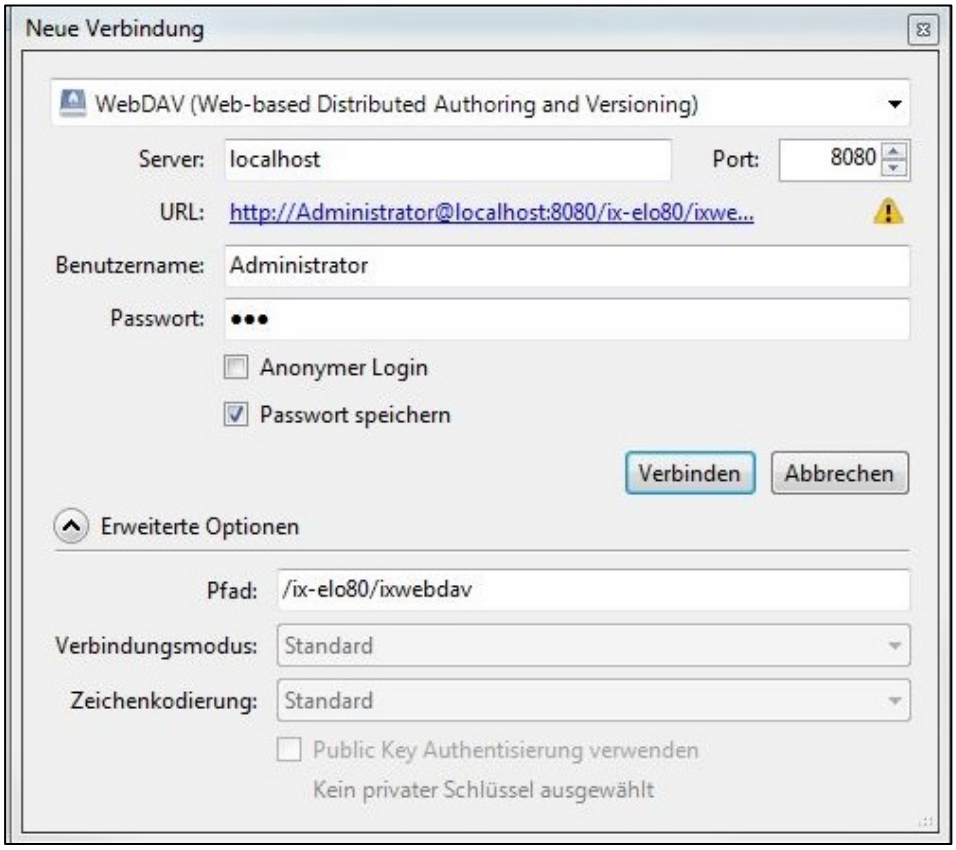


Abb. 20: Einrichtung in Cyperduck

7.5 WebDAV unter Android verwenden

7.5.1 WebDAV File Manager

Für die Verbindung mit ELO WebDAV muss zuerst ein neuer Server konfiguriert werden (Menüeintrag *Server Add*).

- Name: Name der Verbindung (z. B.: „ELO“)
- URL: Internetadresse zum ELO WebDAV.
-

> **Beachten Sie:** Hier darf keine Port-Nummer eingegeben werden.

- User Name: ELO Benutzername
- Password: Passwort für das ELO Konto.

Wählen Sie die Einstellung *Optionally specify a port number* und geben Sie den Port ein.

Klicken Sie *Registration*.

7.5.2 WebDAV Nav Lite

Klicken Sie auf das Plus-Zeichen zum Einfügen eines neuen WebDAV-Servers.

- Name: Name der Verbindung (z. B.: „ELO“)
- Server URL: Internetadresse zum ELO WebDAV.
- Benutzername: ELO Benutzername
- Passwort: Passwort für das ELO Konto.

7.6 Webdav unter iOS verwenden

7.6.1 WebDAV Navigator

Klicken Sie auf das Plus-Zeichen zum Einfügen eines neuen WebDAV-Servers.

- Name: Name der Verbindung (z. B.: „ELO“)
- Server-Adresse: Internetadresse zum ELO WebDAV.
- Benutzername: ELO Benutzername
- Passwort: Passwort für das ELO Konto.

7.6.2 Over The Air

Klicken Sie auf *Add Server*.

- Display Name: Name der Verbindung (z. B.: „ELO“)
- URL: Internetadresse zum ELO WebDAV.
- User ID: ELO Benutzername
- Password: Passwort für das ELO Konto.

7.7 Übersicht: WebDAV-Clients

Mit folgenden WebDAV-Clients wurde ELO WebDAV getestet

Client	Betriebs-systemen	Als Laufwerk verbinden	LOCK	PROPPATCH[1]	MOVE / COPY[2]	Kostenlos	Bemerkungen
Windows Explorer	Windows	Ja	Nein	Nein		N/A	
BitKinex	Windows	Nein	Ja	Nein	Ja	Ja	
WebDrive	Windows, Mac	Ja	Ja	Nein	Nein	Nein	Erzeugt Temp-Dateien
NetDrive	Windows	Ja	Nein	Nein	Nein	Ja, für nicht kommerzielle Nutzung	
Cyberduck	Windows, Mac	Nein	Nein	Ja	Nein	Ja	
Cross FTP Pro	Windows, Mac, Linux	Nein	Ja	Nein	Ja	Nein	
CarotDAV	Windows	Nein		Ja	Ja	Ja	
WebDav-plugin von TotalCommander	Windows, Android	Nein	Nein	Nein		Kostenlos für Android	
Transmit	Mac	Laufwerk und eigene UI	Nein	Nein	Ja	Nein	Erzeugt Temp-Dateien
Finder	Mac	Ja	Nein	Nein	Nein	N/A	Temp-Dateien im Root-Ordner und im aktuellen

[1] Änderung der Metadaten wie Verschlagwortung
[2] MOVE/COPY im Server ohne Herunterladen von Dateien

							Order speichern
Cadaver	Linux	Nein	Ja	Ja	Ja	Ja	Kommandozeile
Dolphin	Linux	Nein	Nein	Nein	Ja	Ja	
Nautilus	Linux	Nein	Nein	Nein	Nein	Ja	
davfs2	Linux	Ja	Nein	Nein	Nein	Ja	Erzeugt Temp-Dateien
Microsoft Office	Windows, Mac	Nein	Nein	Nein	Nein	Nein	
WebDAV File Manager	Android	Nein	Nein	Nein	Ja	Ja	
WebDAV Nav Lite	Android	Nein	Nein	Nein	Ja	Ja	
FolderSync Lite	Android	Nein	Nein	Nein	Ja	Ja	
WebDAV Navigator	iOS	Nein	Nein	Nein	Ja	Ja	
OverTheAir	iOS	Nein	Nein	Nein		Ja	Nicht zum Bearbeiten. Nur zum Lesen und Hochladen der Dateien.

8 Dynamische Stichwortlisten

Bei dynamischen Stichwortlisten handelt es sich um Stichwortlisten, deren Inhalt zur Laufzeit erzeugt wird. Der Inhalt von dynamischen Stichwortlisten kann auch auf bereits eingegeben Daten im Verschlagwortungsdialog reagieren und die Stichworte kontextabhängig filtern.

8.1 Beispiel

Man hat in der Verschlagwortungsmaske die Felder *Abteilung* und *Mitarbeiter*. Die Definition der Abteilungen und ihrer Mitarbeiter wird in einer Datenbank außerhalb von ELO gepflegt. Um die aktuellen Abteilungen abzufragen, wird ein Skript ausgeführt, welches sich mit der externen Datenbank verbindet, vorhandene Abteilungen abfragt und als Stichwortliste zurückliefert. Aus dieser Liste kann dann die gewünschte Abteilung ausgewählt werden.

Das Feld für den Namen des Mitarbeiters sollte nun mit einem Mitarbeiter aus der zuvor ausgewählten Abteilung gefüllt werden. Hierzu sendet die Anwendung die ausgewählte Abteilung an das Skript mit der Information, jetzt die Mitarbeiter abzufragen. Der weitere Verlauf ist wie gehabt.

8.2 Administration

Ein Indexfeld erhält die neue Option *ServerScript*. Hier ist der Name des Skriptes einzutragen, welches zum Laden der Daten für die dynamische Stichwortliste ausgeführt werden soll.

Das Skript müssen Sie im Ordner *Administration* unter *IndexServer Scripting Base* hinterlegen.

8.3 Client-Anwendung

Es wird ein Objekt verschlagwortet. Im Dialog gibt der Benutzer Daten in ein Indexfeld mit hinterlegter dynamischer Stichwortliste ein. Die Client-Anwendung erkennt ein solches Indexfeld daran, dass das Feld *DocMaskLine.serverScriptName* gesetzt ist.

Sobald die Client-Anwendung eine Eingabeverzögerung erkennt, ruft diese die IX-API-Funktion

```
checkoutKeywordsDynamic(ClientInfo,KeywordsDynamicInfo)
```

auf. Im IX wird sodann das zu den gegebenen Parametern entsprechende Skript aufgerufen und das Ergebnis als

```
KeywordsDynamicResult
```

zurückgegeben. Das Ergebnis wird in der Client-Anwendung als Liste oder als Tabelle unterhalb des Indexfeldes angezeigt. Der Benutzer kann daraus die für das Objekt richtige Zeile auswählen.

Mit den Informationen aus diesem Ergebnis kann die Client-Anwendung das in Bearbeitung befindliche Indexfeld vervollständigen. Sind im Ergebnis weitere Spalten mit der Information darüber gegeben, zu welchem Schlüssel die Daten aus dieser weiteren Spalte gehören, besteht für die Client-Anwendung sogar die Möglichkeit weitere Indexfelder zu vervollständigen.

8.4 Beispiel-Skript

Im Folgenden ist ein funktionierendes Beispiel-Skript gelistet, welches zum einen die benötigten Funktionen zeigen und zum anderen als Vorlage für Skripte dienen soll. Rudimentäre Beschreibungen der Funktionen sind im Listing dokumentiert.

8.4.1 SampleScript.js

```
importPackage(Packages.de.elo.ix.jscript);

/**
 * Diese Funktion muss im Skript implementiert werden. Sie wird
 * ohne Argumente aufgerufen. Als Rückgabe wird ein neues Objekt
 * erwartet, welches die abstrakten Funktionen der Klasse
 * DynamicKeyword implementiert.
 */
function getDataIterator() {
  return new DynamicKeyword(new SimpleDatabaseQuery());
}

/**
 * Implementierung der Klasse DynamicKeyword
 * var index: Laufvariable für das ResultSet
 * var resultset: Ergebnis der DB-Abfrage
 */
function SimpleDatabaseQuery() {
  this.index = 0;
  this.resultset = undefined;
}

/**
 * Öffnet die Datenquelle. Diese Funktion wird als erstes aufgerufen.
 * @param ec: Ausführungskontext des Skripts.
 *   Enthält ec.user: UserInfo, ec.ci: ClientInfo,
 * @param sord: Das derzeit in Bearbeitung befindliche Sord. Belegte
 *   Indexfelder können als Filter für die Datenquelle dienen.
 * @param key: Schlüssel der aktuell im Client ausgewählten Zeile.
 */
SimpleDatabaseQuery.prototype.open = function(ec, sord, key) {

  // this.getObjKeyValue erleichtert den Zugriff auf Werte in Indexfeldern
  // Rückgabe ist ein Array von Strings.
  var filter = this.getObjKeyValue(sord, key);

  // SQL-Statement mit Platzhaltern. Diese werden dann von params belegt
```

```
  var statement = "SELECT userid, data_1, data_2, data_3"
      + " FROM testkeywordsdynamic"
      + " WHERE userid LIKE ? AND data_1 LIKE ? ORDER BY id;";

  // params[0] ersetzt das erste ? in statement, params[1] das zweite ?
  var params = [];

  if (ec.user.id == 0) {
    // Der Administrator soll alle Daten geliefert bekommen
    params.push("%");
  } else {
    // Ein normaler User soll nur seine Daten geliefert bekommen
    params.push(ec.user.guid);
  }

  if (filter && filter[0]) {
    // Falls das Sord Daten für das Indexfeld ‘key’ bereitstellt,
    // eine Präfixsuche in der DB durchführen.
    params.push(filter[0] + "%");
  } else {
    // Ansonsten nichts filtern.
    params.push("%");
  }

  var database = new DBConnection("jdbc/TestDataSource", "elo90");
  this.resultset = database.query(statement, params);
};

/* Gibt jeweils die erste noch nicht zurückgegeben Zeile zurück */
SimpleDatabaseQuery.prototype.getNextRow = function() {
  return this.resultset[index++];
};


/* Wird aufgerufen, nachdem getNextRow() keine neuen Ergebnisse mehr
 * zurückgibt oder bereits 1000 Ergebniszeilen abgeholt wurden. */
SimpleDatabaseQuery.prototype.close = function() {
  // DB Verbindung ist bereits geschlossen, sobald database.query(…)
  // zurückkommt. Daher in diesem Fall leer.
};

/* Gibt Überschriften für Spalten als Array zurück. Die Anzahl an Spalten
 * muss mit der Anzahl an Spalten von getNextRow() übereinstimmen. Der IX
 * prüft die Anzahl nicht. */
SimpleDatabaseQuery.prototype.getHeader = function() {
  return ["Nothing", "Something", "Special", "Important"];
};

/* Gibt die Schlüssel für Spalten als Array zurück. Die Anzahl an Spalten
 * muss mit der Anzahl an Spalten von getNextRow() übereinstimmen. Der IX
 * prüft die Anzahl nicht. */
```

```
SimpleDatabaseQuery.prototype.getKeyNames = function() {
  return ["", "testKeywordScriptUser", "", ""];
};

/* Wird aufgerufen, nachdem der IX alle Zeilen abgeholt hat. Sollten aber
 * noch Zeilen übrig sein (also insgesamt >1000), gibt hasMoreRows() true
 * zurück, sonst false. */
SimpleDatabaseQuery.prototype.hasMoreRows = function() {
  return (this.index < (this.resultset.length));
};

/* Gibt eine Nachricht zurück, die der Client anzeigen soll. Können
 * Fehlermeldungen oder Informationsmeldungen sein (z.B. „Bitte zunächst
 * Feld XYZ ausfüllen“) */
SimpleDatabaseQuery.prototype.getMessage = function() {
  return "";
};

/* Gibt einen aussagekräftigen Namen für die Stichwortliste zurück
 */
SimpleDatabaseQuery.prototype.getTitle = function() {
  return "Auftragsdaten";
};
```



Beachten Sie: Die Verwendung von Variablennamen, für die im Skript eine get…()-Funktion existiert, kann zu einer unendlichen Rekursion führen.

Die Java-Script-Engine wandelt Zugriffe auf Properties in entsprechende Getter-Funktionen um. Dadurch entsteht im gekürzten Beispiel unten eine endlose Rekursion sobald die Funktion `getTitle()` aufgerufen wird:

```
function BadKeywordList() {
  this.title = "My Title";
}

BadKeywordList.prototype.getTitle = function() {
  // wird in this.getTitle() umgewandelt.
  return this.title;
};
```

8.5 Konfiguration in der Verschlagwortungsmaske

Das Indexserver Skript muss in der Indexzeile der Maskendefinition angegeben werden. Der Name ergibt sich über die Kurzbezeichnung des Skriptes in der *Indexserver Scripting Base*. Tragen sie in der ELO Administration Console den Skriptnamen in das Feld *Dynamische Stichwortliste* ein.

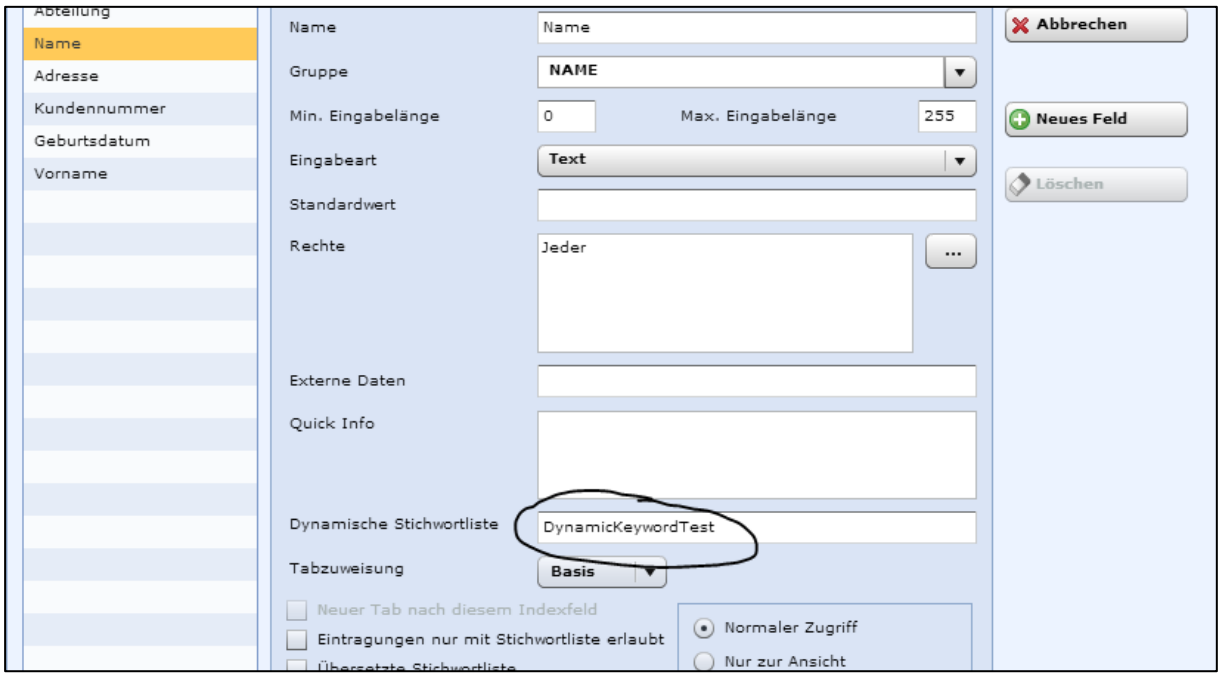


Abb. 21: Eintrag im Feld 'Dynamische Stichwortliste'

8.6 Dynamische Stichwortlisten im ELOwf

8.6.1 Verwendung aus Maskendefinition wie in anderen Clients

Im ELOwf können dynamische Stichwortlisten für die Verschlagwortung wie in anderen Clients verwendet werden. Dazu muss das Eingabefeld wie mit einer normalen Stichwortliste konfiguriert und der Gruppenname als Listenname verwendet werden. Voraussetzung hierfür ist, wie bei den anderen Clients, die Konfiguration des Skriptes in der Maskendefinition (siehe oben).

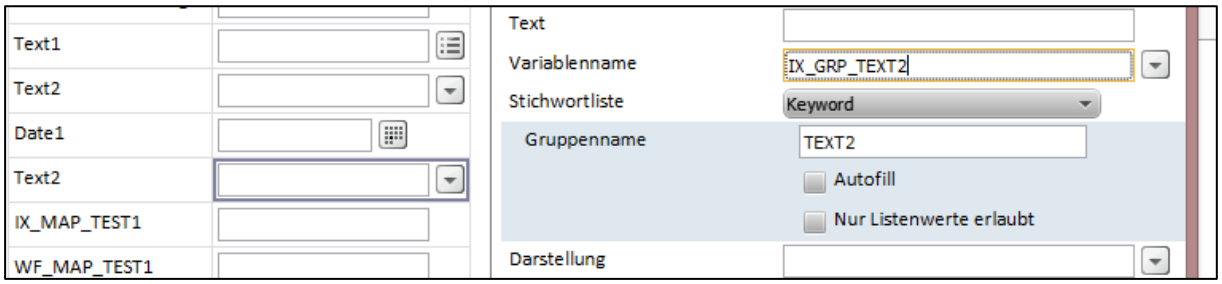


Abb. 22: Konfiguration im Formulardesigner

Wird solch ein Feld im Formular verwendet, sendet dieses wie andere Clients die aktuelle Verschlagwortung aus dem Formular als Sord an den Server und bekommt eine Tabelle mit Werten für einzelne Indexfelder.

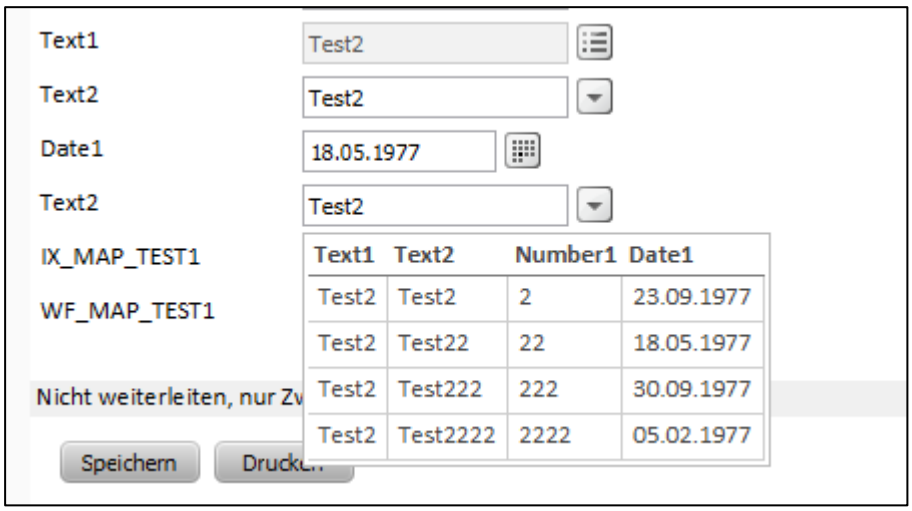


Abb. 23: Beispiel einer dynamischen Stichwortliste im Formular

8.6.2 Verwendung von dynamischen Stichwortlisten für Map-Felder

Im ELOwf können zusätzlich zur herkömmlichen Anwendung dynamische Stichwortlisten auch für Map-Felder definiert werden.

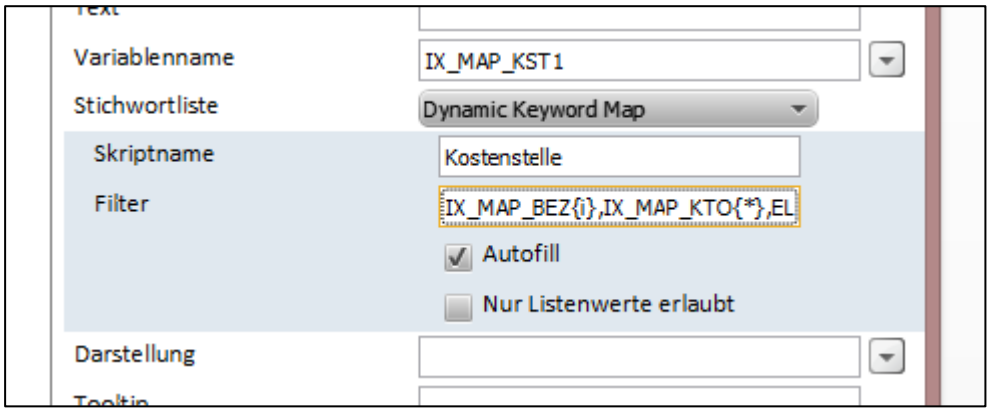


Abb. 24: Einrichtung einer dynamischen Stichwortliste mit einem Map-Feld

Der Variablenname wird als `focusName` an das Indexserver Script übergeben, sobald die dynamische Stichwortliste für dieses Feld aufgerufen wird. Als Stichwortliste muss *Dynamic Keyword Map* verwendet werden. Es erscheinen die Felder *Skriptname* und *Filter*. In *Skriptname* wird der Name (Kurzbezeichnung des Indexserver Skriptes eingesetzt (bitte beachten Sie dass in diesem Fall die Methode `openMap` im Skript aufgerufen wird. Unter *Filter* können alle Felder eingetragen werden, die beim Aufrufen der dynamischen Stichwortliste in die Map übernommen und an das Script übergeben werden sollen. Auch hier gilt das Namensschema mit *IX_MAP_* und *WF_MAP_* für die Map Felder, sowie ELO_PARAMS_ um Werte aus dem statischen ELO_PARAMS Objekt zu übertragen (z.B: ELO_PARAMS_ELO_OBJID überträgt die Objekt-Id des Eintrages). Die Map wird ohne Änderungen als Objekt `map` an das Skript übergeben.

Beim Aufrufen der dynamischen Stichwortliste wird das fokussierte Eingabefeld als `focusName` und die Map mit allen in Filter definierten Feldern inklusive dem fokussierten Feld als `map` übergeben.

8.6.2.1 Besonderheiten

Für die Funktionalität *JS_ADDLINE* können im Filterfeld Platzhalter `{i}` für den Index eingetragen werden. In diesem Fall wird beim Aufruf der dynamischen Indexzeile der Index des Fokusfeldes ausgelesen und anstelle des Platzhalters gesetzt. Wird im obigen Bild zum Beispiel für die 4. Spalte die dynamische Stichwortliste im Feld *IX_MAP_KST4* aufgerufen, wird *IX_MAP_KST4* als `focusName` an das Skript übergeben. In der Map werden die Werte für *IX_MAP_BEZ4* und *IX_MAP_KTO4* sowie der Wert des fokussierten Feldes (*IX_MAP_KST4*) übergeben.

Ebenfalls kann über den Platzhalter `{*}` alle Indexfelder an die Map übergeben werden. Als Beispiel wird im *Filter IX_MAP_KTO{*}* definiert und die dynamische Indexzeile von Feld *IX_MAP_KST4* aufgerufen werden für alle existierenden JS_ADDLINE Zeilen das Feld *IX_MAP_KTO* übertragen, also *IX_MAP_BEZ1*, *IX_MAP_BEZ2*, usw.

9 OCR über den Indexserver

Seit Version 8.00.020 kann mithilfe der Funktion „processOcr“ der „ELO OCR Service“ angesteuert werden. Voraussetzung dafür ist, dass der Service entsprechend konfiguriert ist.

9.1 Konfiguration

Das Serversetup von ELO 9 trägt soweit möglich die nötigen Konfigurationswerte ein. Allerdings richtet es nur einen OCR Worker Prozess ein, der für den ELOtr vorgesehen ist. Dies ist nicht ausreichend, wenn auch die Java-Clients die OCR über den Indexserver ausführen sollen - was automatisch dann der Fall ist, wenn auf den Client-PCs keine OCR installiert ist.

Unter dem Registrierungsschlüssel

HKEY_LOCAL_MACHINE\Software\Wow6432Node\ELO Digital\OCR\Service

kann die Anzahl der Worker im Wert “WorkerCount” festgelegt werden. Der „ELO OCR Service“ startet aber höchstens so viele Prozesse, wie der Computer CPUs anbietet. Die Prozesse laufen standardmäßig auf niedrigster Priorität, damit sie den übrigen Betrieb möglichst wenig stören.

Die Verbindung zwischen OCR und Indexserver ist eine HTTP Verbindung. Die OCR meldet sich beim Indexserver an und wartet auf Aufgaben (Server-Push-Technik). Somit erfolgt die Festlegung, welcher „ELO OCR Service“ mit dem Indexserver arbeitet in der Konfiguration des „ELO OCR Service“ – nicht beim Indexserver.

Die Indexserver URL und das Anmeldekonto wird unter dem Schlüssel

…ELO Digital\OCR\Worker\OcrForIx\<Server-Port-Archiv>

angegeben. Wobei „<Server-Port-Archiv>“ nur als Konvention vorgeschlagen wird. Wie der Sub-Schlüssel unter „OcrForIx“ heißt, spielt technisch keine Rolle.

Unter diesem Schlüssel ist im Wert „url“ die Indexserver URL und im Wert „credentials“ das Anmeldekonto plus Semikolon plus Kennwort einzutragen.

Beispiel:

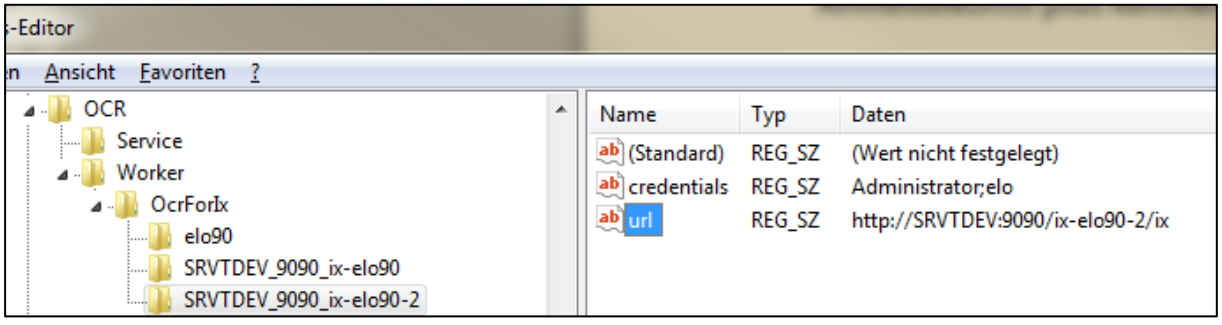


Abb. 25: Registrierungsschlüssel für OCR erstellen

Der Registry-Zweig wird vom „ELO OCR Service“ alle 10s geprüft und Änderungen werden ohne Neustart übernommen. Die Klartexteingabe des Anmeldekontos wird automatisch verschlüsselt.

Ein „ELO OCR Service“ kann mit mehreren Indexservern Verbindung aufnehmen. Andersherum kann ein Indexserver mit mehreren „ELO OCR Service“ Instanzen zusammenarbeiten. Beispielsweise könnte man für eine Anwendung, die intensiv die OCR verwendet, alle „ELO OCR Service“ Dienste, die mit dem ELO Java Client auf den Client-PCs installiert sind, auf den Indexserver konfigurieren.

9.2 Aufruf der processOcr Funktion

9.2.1 Feststellen, ob OCR möglich ist

Mit diesen Zeilen kann festgestellt werden, ob die OCR über den Indexserver möglich ist:

```
private bool checkOcrAvail(IXConnection conn)
{
    bool ocrAvail = true;
    try
    {
        conn.Ix.processOcr(new OcrInfo());
    }
    catch (Exception e)
    {
        if (conn.Ix.parseException(e.ToString()).exceptionType !=
            IXExceptionC.UNSUPPORTED_FUNCTION)
        {
            throw e;
        }
        ocrAvail = false;
    }
    return ocrAvail;
}
```

Wenn die OCR nicht verfügbar ist, sich also keine OCR Worker beim Indexserver angemeldet haben, wird die Exception mit dem Typ „IXExceptionC.UNSUPPORTED_FUNCTION“ geworfen.

9.3 Landessprachen ausgeben

Die von der OCR unterstützten Landessprachen können wie folgt ermittelt werden.

```
OcrInfo ocrInfo = new OcrInfo();
ocrInfo.queryLanguages = new OcrInfoQueryLanguages();

// Extenal language names should be returned in this language
ocrInfo.messagesLanguage = OcrInfoC.MESSAGES_LANGUAGE_GERMAN;

OcrResult ocrResult = conn.Ix.processOcr(ocrInfo);
for (int i = 0; i < ocrResult.queryLanguages.externalLangs.Length; i++)
```

```
{
    Logger.instance().log("external=" +
                          ocrResult.queryLanguages.externalLangs[i]);
    Logger.instance().log("internal=" +
                          ocrResult.queryLanguages.internalLangs[i]);
}
```

Die externen Sprachnamen können in einer Benutzeroberfläche ausgegeben werden. Als Parameter für die Analyse müssen die internen Sprachnamen gesetzt werden.

9.4 OCR-Analyse synchron durchführen

Die OCR Analyse kann synchron auf die folgende Art erfolgen. Für die Erkennung eines Archivdokuments setzt man „ocrInfo.recognizeFile.objId“ auf die Objekt-ID des Dokuments. Soll eine beliebige Datei analysiert werden, muss sie zur Übergabe komplett in den Speicher gelesen und dem Element „ocrInfo.recognizeFile.imageData“ zugewiesen werden.

```
OcrInfo ocrInfo = new OcrInfo();
ocrInfo.recognizeFile = new OcrInfoRecognizeFile();
ocrInfo.recognizeFile.imageData = new FileData();
ocrInfo.recognizeFile.imageData.data = File.ReadAllBytes(fileName);
ocrInfo.recognizeFile.imageData.contentType = ".TIF";
ocrInfo.recognizeFile.recognizeLangs = new String[] { "German" };
ocrInfo.recognizeFile.timeoutSeconds = 30;
ocrInfo.recognizeFile.outputFormat = OcrInfoC.TEXT;

OcrResult ocrResult = conn.Ix.processOcr(ocrInfo);
String text = ocrResult.recognizeFile.text;
Logger.instance().log("text=" + text);
```

Solange die OCR weniger als 2min für die Analyse benötigt, ist dieser einfache Weg zu empfehlen. Dauert die Verarbeitung aber länger, dann sollte die asynchrone Ausführung gewählt werden.

9.5 OCR-Analyse asynchron ausführen

Bei der asynchronen Verarbeitung wird das OCR-Ergebnis über den Ereignisbus versendet. Dadurch hat die OCR beliebig lang Zeit, den Text zu ermitteln.

```
// Synchronisation object
private Object syncObject = ("syncObj").Clone();

// OCR Result
private OcrResult result;

private String recognizeFileAsync(IXConnection conn, String fileName,
                                  int timeoutSeconds)
{
    // Open an event bus private to the session ticket and add a listener.
```

```
    // OCR results are send over this bus.
    EventBusApi.Bus eventBus = conn.EventBusApi.OpenEventBusChannel(
                                    conn.LoginResult.clientInfo.ticket);
    eventBus.AddListener(EventBusC.EVENT_OCR_RESULT);

    // Add a handler function for events.
    conn.EventBusApi.EventBusHandler +=
                           new EventBusApi.ProcessEventBusEvents(
                                           EventBusApi_EventBusHandler);

    OcrInfo ocrInfo = new OcrInfo();
    ocrInfo.recognizeFile = new OcrInfoRecognizeFile();
    ocrInfo.recognizeFile.imageData = new FileData();
    ocrInfo.recognizeFile.imageData.data = File.ReadAllBytes(fileName);
    ocrInfo.recognizeFile.imageData.contentType = ".TIF";
    ocrInfo.recognizeFile.recognizeLangs = new String[] { "English" };
    ocrInfo.recognizeFile.timeoutSeconds = timeoutSeconds;
    ocrInfo.recognizeFile.outputFormat = OcrInfoC.TEXT;
    ocrInfo.recognizeFile.pageNo = -1; // All pages

    // Set this properties for asynchronous processing:

    // Event bus that receives the result
    ocrInfo.busId = eventBus.Id;
    // ID for the request, unique over open requests on ocrInfo.busId
    ocrInfo.eventId = 123;

    // Process OCR
    conn.Ix.processOcr(ocrInfo);

    // Wait for result
    lock (syncObject)
    {
        long t1 = DateTime.Now.Ticks / 10000;
        int dt = timeoutSeconds * 1000;
        while (result == null)
        {
            Monitor.Wait(syncObject, dt);
            long t2 = DateTime.Now.Ticks / 10000;
            dt -= (int)(t2 - t1);
            if (dt <= 0) throw new Exception("Timeout");
            t1 = t2;
        }
    }

    String text = result.recognizeFile.text;
    return text;
}

// Event handler function.
private void EventBusApi_EventBusHandler(long subsId, Event[] events)
```

```
{
    foreach (Event ev in events)
    {
        if (ev.type == EventBusC.EVENT_OCR_RESULT)
        {
            lock (syncObject)
            {
                this.result = (OcrResult)AnyToObject.ToObject(ev.any);

                // Wakeup thread in recognizeFileAsync()
                Monitor.PulseAll(syncObject);
            }
        }
    }
}
```

9.6 OCR-Analyse mit Rückgabe der Zeichenpositionen

Neben der reinen Textinformation kann die OCR auch die Positionen der erkannten Zeichen zurückgeben. Hierzu setzt man als Ausgabeformat „ocrInfo.recognizeFile.outputFormat = OcrInfoC.CHAR_AND_RECT_EX“. Das Ergebnis befindet sich in „ocrResult.recognizeFile. textData.data“.

Diese Daten beginnen mit dem folgenden Vorspann:

Header:

Pos	Purpose	Value	Bytes
0	File ID	0x52434F45	4
4	Version	1	4
8	Skew Angle	Double IEEE 754	8
16	Width in Points	Integer	4
20	Height in Points	Integer	4

In den Vorspann schließen sich unmittelbar je Zeichen die folgenden Daten an:

Zeichenpositionen:

Pos	Purpose	Value	Bytes
0	UTF-16 Character	Wide Character	2
2	X-Pos. Upper Left Corner	Integer	4
6	Y-Pos. Upper Left Corner	Integer	4
10	X-Pos. Bottom Right Corner	Integer	4
14	Y-Pos. Bottom Right Corner	Integer	4
18	Confidence in Percent	Integer	4
22	Original UTF-16 Character (might be replaced due to low confidence)	Wide Character	2

Die Werte sind in der Byte-Order Little Endian ausgerichtet. Die Y-Achse zeigt in positiver Richtung nach unten. Die Koordinatenwerte sind gemessen in Bildpunkten.

Beispiel:

```
OcrInfo ocrInfo = new OcrInfo();
ocrInfo.recognizeFile = new OcrInfoRecognizeFile();
ocrInfo.recognizeFile.imageData = new FileData();
ocrInfo.recognizeFile.imageData.data = File.ReadAllBytes(fileName);
ocrInfo.recognizeFile.imageData.contentType = ".TIF";
ocrInfo.recognizeFile.recognizeLangs = new String[] { "English" };
ocrInfo.recognizeFile.timeoutSeconds = 30;
ocrInfo.recognizeFile.outputFormat = OcrInfoC.CHAR_AND_RECT_EX;
ocrInfo.recognizeFile.pageNo = 0; // First pages

// Recognize document
OcrResult ocrResult = conn.Ix.processOcr(ocrInfo);

// Put result bytes into BinaryReader
byte[] data = ocrResult.recognizeFile.textData.data;
BinaryReader rd = new BinaryReader(new MemoryStream(data));

// Read header
int magic = rd.ReadInt32();
if (magic != 0x52434F45) throw new Exception("Unrecognized format");
int version = rd.ReadInt32();
if (version != 1) throw new Exception("Unsupported version");
double skew = rd.ReadDouble();
int width = rd.ReadInt32();
int height = rd.ReadInt32();
```

```
    Logger.instance().log("magic=0x" + Convert.ToString(magic, 16));
    Logger.instance().log("version=" + version);
    Logger.instance().log("skew=" + skew);
    Logger.instance().log("width=" + width);
    Logger.instance().log("height=" + height);

    // Read characters
    int headerSize = 4 + 4 + 8 + 4 + 4;
    int charSize = 2 + 4 + 4 + 4 + 4 + 4 + 2;
    int nbOfChars = (data.Length - headerSize) / charSize;
    for (int i = 0; i < Math.Min(10, nbOfChars); i++)
    {
        char ch = (char)rd.ReadInt16();
        int left = rd.ReadInt32();
        int top = rd.ReadInt32();
        int right = rd.ReadInt32();
        int bottom = rd.ReadInt32();
        int percent = rd.ReadInt32();
        char chOrig = (char)rd.ReadInt16();

        Logger.instance().log("char " + i + ": '" + ch + "'");
        Logger.instance().log("  rect=(" + left + "," + top + "," +
                              right + "," + bottom + ")");
        Logger.instance().log("  confidence=" + percent +
                              ", orig='" + chOrig + "'");
    }
```

9.7 OCR-Analyse eines Rechtecks

Die OCR Analyse kann auf ein Rechteck eingeschränkt werden. Das Rechteck wird im Element „ocrInfo.recognizeFile.recognizeRects“ übergeben.

```
    OcrInfo ocrInfo = new OcrInfo();
    ocrInfo.recognizeFile = new OcrInfoRecognizeFile();
    ocrInfo.recognizeFile.imageData = new FileData();
    ocrInfo.recognizeFile.imageData.data = File.ReadAllBytes(fileName);
    ocrInfo.recognizeFile.imageData.contentType = ".TIF";
    ocrInfo.recognizeFile.recognizeLangs = new String[] { "English" };
    ocrInfo.recognizeFile.timeoutSeconds = 30;
    ocrInfo.recognizeFile.outputFormat = OcrInfoC.TEXT;
    ocrInfo.recognizeFile.pageNo = 0;

    ocrInfo.recognizeFile.rectUnit = OcrInfoC.UNIT_PER_THOUSAND;

    int pmLeft = 50, pmTop = 100, pmRight = 200, pmBottom = 1000;
    ocrInfo.recognizeFile.recognizeRects = new OcrRect[] {
                    new OcrRect(pmLeft, pmTop, pmRight, pmBottom) };

    OcrResult ocrResult = conn.Ix.processOcr(ocrInfo);
    String text = ocrResult.recognizeFile.text;
```

Wenn die Größe des Dokuments in Bildpunkten nicht vorab bekannt ist, bietet es sich an, die Koordinaten des Rechtecks relativ zu den Dokumentabmessungen anzugeben. Dafür gibt man „ocrInfo.recognizeFile.rectUnit = OcrInfoC.UNIT_PER_THOUSAND“ an und übergibt die Koordinaten als Promille-Werte relativ zur Breite bzw. Höhe.

10 Preview-Generierung

Die Ansicht von Dokumenten in einem Web-Browser oder auf Mobilgeräten ist ohne zusätzliche Erweiterungen meist nicht möglich. Um PDF- oder TIFF-Dateien dennoch anzeigen zu können, gibt es seit Version 9.00.000 einen Modul im Indexserver, welches verschiedene Eingabeformate in Vorschaubilder konvertiert, auf eine gewünschte Größe skaliert und bei Bedarf vorhandene Annotationen zeichnet.

Die Funktionen zum Skalieren sowie Zeichnen von Annotationen können auch auf im Archiv hinterlegte Bilddokumente angewendet werden. Die unterstützten Formate sind BMP, JPEG, PNG und PSD (Adobe Photoshop).

10.1 Technischer Hintergrund

Um eine Vorschau für ein Dokument als Bilddatei zu erzeugen, sind folgende zwei Schritte notwendig:

URL-Erzeugung für die Seiten eines Dokumentes

Es wird das Dokument gelesen, die erste Seite gerendert und dem Aufrufer dessen Dimensionen, die Anzahl an Seiten des Dokumentes und für jede Seite eine generierte URL zurückgegeben.

Herunterladen der Vorschau

Sobald man die URL für eine Seite des Dokumentes erzeugt lassen hat, kann man über Sie die Vorschau herunterladen. Das verwendete Protokoll hierfür ist http.

Zum Einlesen und Rendern verschiedener Dokumenttypen werden 3rd-Party-Bibliotheken verwendet. Für PDF ist dies ICEpdf, für TIFF zum einen JAI (Java Advanced Imaging) und zum anderen TwelveMonkeys (https://github.com/haraldk/TwelveMonkeys), welches auch zur Verarbeitung von Photoshop-Dateien verwendet wird. Abgesehen von ICEpdf werden die Bibliotheken nicht direkt angesprochen, sondern zur Laufzeit als Plug-In für die ImageIO-API geladen.

Da es vorkommen kann, dass spezielle Dokumente in den 3rd-Party-Bibliotheken Instabilität zum Beispiel in Form von hoher Prozessor- oder Speicherlast verursachen, wurde die Konvertierung von PDF- und TIFF-Dateien in einen externen Prozess ausgelagert. Dies hat den Vorteil, den Konvertierungsprozess sicher beenden und dessen Ressourcen wieder vollständig zur Verfügung stellen zu können, ohne den Indexserver selbst zu beeinträchtigen oder neu starten zu müssen. Hierzu wird beim Start des Indexservers aus dem War-Archiv die Jar-Datei elopp.jar in ein temporäres Verzeichnis entpackt und gestartet, sobald erstmalig die Preview-Generierung eines TIFF- oder PDF-Dokumentes angefragt wird. Das Stoppen des IX beendet ebenso den externen Prozess. Zusätzlich schickt der Prozess alle 10 Sekunden ein ping-Signal an den Indexserver und beendet sich selbst, falls er nicht erreicht wird.

Auf dem Server erzeugte Bilder werden mit ihren Metadaten in einem Cache zwischengespeichert. Dieser Cache hält Objekte eine konfigurierte Zeitspanne vor und räumt selbstständig veraltete Objekte auf. Sollte die Vorschau eines Dokumentes angefragt werden, welche sich bereits im Cache befindet, so wird das eigentliche Dokument nicht mehr geladen, sondern ausschließlich die Vorschau verwendet. Eine Ausnahme bilden hier PDF-Dokumente, wenn sie in einer noch nicht zwischengespeicherten Größe angefragt werden. Hier wird dann das bereits erzeugte Bild nicht kleiner skaliert, sondern das Vorschaubild in der entsprechenden Größe neu generiert, was in einer sichtbar besseren Qualität bei geringerem Speicherverbrauch des Vorschaubildes mündet.

Die Kodierung der zu erzeugenden Bilder wählt serverseitig eine Heuristik zwischen JPEG und PNG aus. Erstere Kodierung wird bei farbigen Bildern bevorzugt, da hier die Dateigröße deutlich geringer ausfällt und letztere für Text und Schwarzweißbilder. PNGs bieten bei Text den Vorteil fehlender Kodierungs-Artefakte, bei Schwarzweißbildern unterstützt die verwendete Bibliothek pngj das Erstellen indizierter Farben, wodurch die Dateigröße stärker als bei JPEG reduziert werden kann.

Zum Skalieren von Bildern wird die Bibliothek imgscalr verwendet. Erzeugte Bilder werden ausschließlich kleiner skaliert.

10.2 Preview erzeugen

Für ein Dokument soll eine Vorschau erzeugt werden. Hierzu ruft die Client-Anwendung folgende Funktion auf:

```
checkoutPreviewImageURLs(PreviewImageInfo);
```

In den Parametern werden ID des Sords oder Dokuments angegeben, sowie die gewünschte Größe der Vorschaubilder, Seitenbereich und ob Annotationen gerendert werden sollen:

```
private String getSordPreviewUrl(IXConnection conn, Sord sord) {
    PreviewImageInfo imageInfo;
    imageInfo = new PreviewImageInfo();
    imageInfo.previewSize = PreviewImageInfoC.SIZE_ORIGINAL;
    imageInfo.objectId = sord.guid;
    imageInfo.startPage = 1;
    imageInfo.endPage = 1;
    imageInfo.renderAnnotations = true;
    imageInfo.renderAnnotationsData = true;
    return conn.ix().checkoutPreviewImageURLs(imageInfo).url[0];
}
```

Als Ergebnis erhält man ein

```
PreviewImageResult
```

welches neben den URLs für die einzelnen Seiten auch Meta-Informationen über die erste angefragt Seite enthält. Hierbei ist zu beachten, dass es sich bei PDF-Dokumenten um die Abmessungen in Pixel des Dokumentes in 300dpi und bei TIFF-Dokumenten um die des Originaldokumentes handelt. Da bei TIFF-Dokumenten die horizontale und vertikale Auflösung voneinander abweichen können, werden solche im Server normiert. Um Client-seitig in der Lage zu sein, eine akkurate OCR anbieten zu können, werden daher die originalen Abmessungen zurückgegeben.

Zum Herunterladen der Vorschau kann man beliebige Software verwenden, die das http-Protokoll unterstützt. Im Client erreicht man dies zum Beispiel über:

```
IXConnection.download(String url, File file);
```

10.3 Administration

Über die Konfigurationsseite des Indexservers können für die Erzeugung von Vorschaubildern verschiedene Parameter angepasst werden. Die folgenden Abschnitte erläutern deren Funktion und Standardwerte.

10.3.1 Kontrolle des Externen Prozesses

Damit Client-Anwendungen zum Beispiel im Falle Verarbeitungsfehlers nicht unbegrenzt auf eine Antwort des Servers warten, findet die Erzeugung nebenläufig statt. Der Parameter

```
imagingTimeoutProcessPageSeconds
```

gibt an, wie lange auf die Erzeugung eines Vorschaubildes gewartet werden soll. Wird die Grenze überschritten, so überträgt der Indexserver ein lokalisiertes Ausweichbild:



Abb. 26: Ausweichbild

Die Verarbeitung findet allerdings weiterhin statt, sodass eine spätere Anfrage mit denselben Parametern durchaus das gewünschte Ergebnis bringen kann. Der Grund für das Überschreiten des Timeouts wird meist hohe Rechenlast auf dem Server sein. Häuft sich die Übertragung der Ausweichbilder, so sollte dieser Parameter erhöht werden.

Sollte es ein Dokument geben, welches zum Beispiel eine Endlosschleife verursachen sollte, wird zum einen dessen Verarbeitung nie beendet und zum anderen entsteht durch den Prozess eine hohe Last auf dem Server. Der Indexserver erkennt solche Situationen und sendet dann dem externen Prozess ein Signal sich selbst zu beenden. Wird dies nicht umgesetzt, wird er von außen terminiert. Die Zeitspanne, bevor der Indexserver den Prozess beenden soll wird über den Parameter

```
imagingTimeoutKillProcessSeconds
```

eingestellt. Im Standard ist sein Wert auf 60 Sekunden gesetzt. Im Gegensatz zum ersten Timeout-Parameter findet eine weitere Verarbeitung sowohl des Dokuments, welches zur Auslösung der Terminierung führt, als auch alle weiteren Dokumente, die parallel in der Warteschleife waren, zu diesem Zeitpunkt nicht mehr statt. Befinden sich im Archiv Dokumente, die eine längere Verarbeitungszeit benötigen - zum Beispiel aufgrund ihrer Größe - ist dieser Wert zu erhöhen.

Dem Verarbeitungsprozess können individuelle JVM-Parameter übergeben werden. Der Wert des Parameters

```
imagingProcessJvmArgs
```

wird ungefiltert als JVM-Parameter gesetzt. Hier kann man zum Beispiel dem Programm mehr oder weniger Arbeitsspeicher zuweisen. Ohne manuelle Anpassung wird hier bei 32-bit JVMs 1500 MB, bei 64-bit 2GB gesetzt. Bitte beachten Sie, dass dies bei 32-bit JVMs von Oracle dem Maximalwert entspricht. Der Versuch einen höheren Wert anzugeben quittiert die JVM mit sofortigem Programmabbruch.

10.3.2 Cache

Im Zwischenspeicher werden erzeugte Bilder auf dem Massenspeicher persistiert. Soweit nicht anders spezifiziert wird das Verzeichnis „cache_previews“ im Konfigurationsordner des Indexservers verwendet. Geändert wird diese Einstellung über den Parameter

```
imagingImageCacheDir
```

Die folgenden Parameter schränken die Lebenszeit von Objekten im Cache nach verschiedenen Kriterien ein. Sobald entsprechende Werte überschritten werden, wird innerhalb des Caches durch Löschen von veralteten oder überdurchschnittlich großen Einträgen aufgeräumt, bis die Grenzen wieder eingehalten werden.

Die Parameter lauten:

```
imagingImageCacheMaxEntries
imagingImageCacheLifetimeSeconds
imagingImageCacheSpace
```

Der Reihenfolge nach sind die Standardwerte auf 250000 Einträge, 2419200 Sekunden (28 Tage) und 21474836470 Bytes (20 GiB) gesetzt.

11 Skriptentwicklung

Für die ELO 9 Version des Indexservers kann die Entwicklungsumgebung Eclipse genutzt werden, um Indexserver-Ereignisskripte zu entwickeln. Auch das Debugging ist in dieser Umgebung möglich, sodass zum Schreiben von Code und zur Fehlersuche nicht zwischen verschiedenen Werkzeugen gewechselt werden muss.

Dies wird durch das „ELO Scripting Plug-In for Eclipse“ möglich. Es wird im Eclipse über „Help – Install New Software…“ von der URL http://forum.elo.com/elo-scripting-plugin-update/ installiert.

Neben Indexserver-Skripten können auch Skripte für den ELO Java Client entwickelt werden.

Das Plugin ersetzt den Rhino Debugger, der nicht mehr verwendet werden kann.

11.1 IXConnection Klasse in Ereignisskripten

Bis zur Version 9.00.012 steht für Ereignisskripte mit dem „ix“ Objekt nur der Zugriff auf die Schnittstelle „IXServicePortIF“ bereit, und es muss in jedem Aufruf die „ClientInfo“ übergeben werden. Seit Version 9.00.014 befinden sich implizit im Kontext der Skripte auch die IXConnection-Objekte „ixConnect“ und „ixConnectAdmin“. Sie enthalten die aus dem ELOas-Skripting und der Client-Programmierung gewohnten Eigenschaften und Funktionen. Im Rechtekontext des aktuellen Benutzers werden die Funktionen von „ixConnect“ ausgeführt. Benötigt das Skript Administratorrechte, kann stattdessen „ixConnectAdmin“ verwendet werden.

Beispiel:

```
function RF_testConnectionCallAPI() {
  log.info("RF_testConnectionCallAPI(");

  var ed = ixConnect.ix().createDoc("1", "", "", EditInfoC.mbSord);
  ed.sord.name = "sord.name";

  log.info(")RF_testConnectionCallAPI=" + ed);
  return ed;
}
```

Über die IXConnection Objekte erhält man auch Zugriff auf weitere Schnittstellen, z. B. der Dokument-Feed-Schnittstelle:

```
function RF_testConnectionFeed(ec, arg) {
  log.info("RF_testConnectionFeed(");

  var objId = arg;
  var act = ixConnect.getFeedService().createAction(
                                  EActionType.UserComment, objId);
  act.text = "Feedaction text";
  var actionGuid = ixConnect.getFeedService().checkinAction(act,
                                  ActionC.mbAll);

  log.info(")RF_testConnectionFeed=" + actionGuid);
  return actionGuid;
}
```

Daneben können, wie aus der Client-Programmierung gewohnt, Dokumente hoch- und heruntergeladen werden:

```
function RF_testConnectionUpload(ec, arg) {
  log.info("RF_testConnectionUpload(");

  var tempFile = File.createTempFile("ixconn", ".txt");
  log.info("create tempFile=" + tempFile);
  var fos = new FileOutputStream(tempFile);
  fos.write(tempFile.getAbsolutePath().getBytes("UTF-8"));
  fos.close();

  var guid = arg;
  log.info("createDoc, guid=" + guid);
  var ed = ixConnect.ix().createDoc("1", "", "", EditInfoC.mbAll);
  ed.sord.guid = guid;
  ed.sord.name = "RF_testConnectionUpload-doc";

  ed.document = new Document();
  ed.document.docs = [new DocVersion()];
  ed.document.docs[0].ext = ixConnect.getFileExt(tempFile);
  log.info("checkinDocBegin, ext=" + ed.document.docs[0].ext);
  ed.document = ixConnect.ix().checkinDocBegin(ed.document);

  log.info("upload, url=" + ed.document.docs[0].url);
  ed.document.docs[0].uploadResult = ixConnect.upload(
                                  ed.document.docs[0].url, tempFile);

  log.info("checkinDocEnd");
  ed.document = ixConnect.ix().checkinDocEnd(ed.sord, SordC.mbAll,
                                  ed.document, LockC.NO);
  var objId = ed.document.objId;
  log.info("objId=" + objId);
```

```
  tempFile["delete"]();

  log.info(")RF_testConnectionUpload=" + objId);
  return objId;
}


function RF_testConnectionDownload(ec, arg) {
  log.info("RF_testConnectionDownload(");

  var objId = arg;
  var sord = ixConnect.ix().checkoutDoc(objId, "", EditInfoC.mbAll,
                                        LockC.NO).getSord();
  log.info("sord=" + sord);

  var tempFile = File.createTempFile("ixconn", "." +
                                     sord.docVersion.ext);
  ixConnect.download(sord.docVersion.url, tempFile);

  var textFile = new Packages.de.elo.ix.jscript.DiscFile(
                                     tempFile.getAbsolutePath());
  var content = textFile.readAllText();

  tempFile["delete"]();

  log.info(")RF_testConnectionDownload=" + content);
  return content;
}
```

12 Feed-Einträge erstellen

In den Dokument-Feed können Einträge programmatisch eingefügt werden, um beispielsweise einen wichtigen Workflow-Status im Feed festzuhalten.

Automatisch generierte Feed-Beiträge sollten immer mit dem Typ „EActionType. AutoComment“ geschrieben werden.

Das folgende Beispiel zeigt das Erstellen eines Feed-Beitrags:

```
private void createFeedAction(IXConnection conn,
        Sord sord, String text) throws RemoteException {
    Action action = conn.getFeedService().createAction(
                              EActionType.AutoComment, sord.guid);
    action.text = text;
    action.guid = conn.getFeedService().checkinAction(
                              action, ActionC.mbAll);
}
```