Infopark CMS Fiona

Search Server

Infopark CMS Fiona

Search Server

Inhalt

1

1 Vorbemerkungen

Dieses Dokument wendet sich an Systemadministratoren und Entwickler, die ihr CMS mit der Infopark Search Cartridge mit vielfältigen Suchfunktionen ausstatten möchten. Die Cartridge ist ein optionaler, separat zu lizensierender Bestandteil von Infopark CMS Fiona. Sie kann sowohl im Redaktions- als auch auf dem Livesystem eingesetzt werden.

Leser dieses Dokuments sollten mit der Installation und Konfiguration des von Infopark CMS Fiona vertraut sein sowie Kenntnisse über Suchmaschinenkonzepte haben. Die Search Cartridge in einen Webauftritt zu integrieren, setzt Wissen vor allem über Dateien, den Export und Scripting voraus.

2

2 Konzepte der Infopark Search Cartridge

2.1 Allgemeine Funktionsweise

Die Infopark Search Cartridge besteht aus einer Server-Applikation (Search Engine Server, SES) sowie der Suchmaschine von Autonomy, die Infopark CMS Fiona um vielfältige Suchmöglichkeiten erweitern. Sie kann auf der Seite des Redaktionssystems und auf dem Live-Server eingesetzt werden, um sowohl den Redakteuren als auch den Nutzern Ihres Online-Angebots die Möglichkeit zu geben, Inhalte nach ihren Kriterien zu suchen.

Die Search Cartridge ist durch ihre Architektur sehr flexibel einsetzbar und kann an die individuellen kundenspezifischen Bedürfnisse angepasst werden. So verfügt der Search Engine Server beispielsweise über Schnittstellen, die es ermöglichen, die zu indizierenden Inhalte vorzuverarbeiten oder Suchergebnisse nachzubereiten (siehe Architektur).

Wie auch der Content Manager und die Template Engine hat der SES eine XML-Schnittstelle, über die er den Clients seine Funktionen verfügbar macht. Clients können nicht nur der Content Manager und die Template Engine sein, sondern auch Skripte, die über das XML-Interface mit dem Search Engine Server kommunizieren.

Der SES lässt sich auf dem Live-Server mit und ohne Template Engine einsetzen.

2.1.1 Redaktionssystem

Auf der Seite des Redaktionssystems wird der Search Engine Server vom Content Management Server mit den zu indizierenden Daten versorgt. Der SES bereitet die Daten optional auf und sendet sie zur Indizierung an die Suchmaschine. Während Benutzer mit Dateien und Versionen arbeiten, sorgt der Content Manager dafür, dass die indizierten Daten der Search Cartridge auf dem aktuellen Stand gehalten werden. Die Benutzer merken in der Regel nicht, dass Anlege- oder Löschoperationen im Hintergrund dazu führen, dass die Indizes der Search Cartridge aktualisiert werden.

Aktualisierungen wie auch Suchanfragen, die die Benutzer des Content Managers durchführen, werden vom Content Manager mit XML-Dokumenten zur Search Cartridge kommuniziert. Diese Dokumente sind entsprechend einer XML-DTD (*Document Type Definition*) aufgebaut. Das Protokoll hat den Namen *Method of Interacting with Search Engines* (MISE, siehe MISE als DTD).

Die XML-Dokumente werden mit HTTP-Requests übertragen. Dabei tritt der Content Manager gegenüber der Search Cartridge als Client auf, d. h. er übermittelt der Cartridge Aktualisierungs- oder Suchaufträge, die sie ausführt und beantwortet.

2.1.2 Live-System

Im Gegensatz zum Redaktionssystem werden auf dem Live-System Aktualisierungs- und Suchanfragen nicht vom gleichen Client ausgelöst. Hier hat die Template Engine die Aufgabe, bei Content-Aktualisierungen eine entsprechende Anfrage an die Search Cartridge zu senden, während bei Suchanfragen eine andere Anwendung auf dem Live-Server (beispielsweise ein PHP-Skript) diese Aufgabe übernimmt und aus den Suchergebnissen auch gleich HTML-Seiten produziert.

Um aus Dokumenten heraus direkt mit dem Search Engine Server zu kommunizieren, lassen sich Skripte einsetzen. Ein Skript kann beispielsweise aus dem Suchanfragetext eines Formulars ein XML-Dokument generieren und dieses an den Search Engine Server senden. Anschließend nimmt das Skript die Suchergebnisse von der Search Cartridge als Antwort entgegen und erzeugt daraus die auszuliefernden Ergebnisseiten. Skripte und Suchformulare lassen sich bequem als CMS-Dateien im Content Manager pflegen.

Die Search Cartridge kann auch auf Live-Systemen eingesetzt werden, die auf Ruby on Rails und dem Rails Connector für Infopark CMS Fiona basieren.

2.2 Architektur

Die Infopark Search Cartridge besteht aus dem Search Engine Server und einem Suchmodul von Autonomy (ehemals Verity). Sämtliche Kommunikation zwischen Clients und der Search Cartridge findet über den Search Engine Server statt.

Der Search Engine Server hat die gleiche Architektur wie andere CMS-Applikationen auch. Er wird stets als Master-Server ausgeführt. Der Master erzeugt Slaves, denen er die einzelnen Anfragen in Form von HTTP-Verbindungen übergibt. Die Slaves kommunizieren anschließend selbständig mit den HTTP-Clients, die die Requests initiiert haben. Der Master-Server hat in diesem Zusammenhang eine Überwachungsfunktion. Er sorgt dafür, dass stets genügend Slaves verfügbar sind, an die er hereinkommende Verbindungen verteilen kann. Die Grenzen, d. h. beispielsweise die minimale und maximale Anzahl laufender Slaves, werden ihm durch Konfigurationswerte gesetzt.

Wie der Content Manager und die Template Engine hat der Search Engine Server eine Tcl-Schnittstelle, mit dem administrative Aufgaben erledigt werden können, beispielsweise Collections (zur Aufnahme von Indizes) anlegen und verwalten. Ferner lassen sich die Suchfunktionen des Search Engine Servers mit Tcl-Skripten erweitern. So ist es beispielsweise möglich, Suchanfragen vorzuverarbeiten oder Suchergebnisse nachzubereiten.

Such- und Indizierungsaufgaben werden dem Search Engine Server von Clients wie dem Content Manager oder einem Suchskript über sein XML-Interface übergeben. Das XML-Interface des Servers und die DTD der Dokumente, die darüber übertragen werden, erschließen den Clients die Funktionen der Search Cartridge und vereinheitlichen den Zugriff auf sie.

2.3 Content-Indizierung

2.3.1 Collections

Die Search Cartridge indiziert Dokumente in so genannte Collections. Die Möglichkeit, für die Indizierung eines bestimmten Dokuments eine bestimmte Collection zu wählen, kann später dazu verwendet werden, die Suche zu beschleunigen. So braucht beispielsweise bei einem Webauftritt in zwei oder mehr Sprachen nur der Content in einer Sprache durchsucht zu werden, wenn die Sprache

ein Auswahlkriterium bei der Suche ist. Es ist jedoch auch möglich, mehrere oder alle Collections zu durchsuchen.

Bei der Installation von Infopark CMS Fiona werden zwei Collections angelegt, je eine für die Redaktions- und die Live-Seite. Weitere Collections können mit einem Tcl-Kommando des Search Servers angelegt werden. Hierbei wird eine vorgegebene Konfiguration verwendet, die unter anderem Ländereinstellungen wie den Zeichensatz der indizierten Dokumente und die Dokumentfelder festlegt, deren Werte in Suchergebnissen zurückgegeben werden sollen. Die Struktur einer Collection lässt sich nicht nachträglich ändern.

Wenn auf dem Live-Server die Template Engine eingesetzt wird, wird für die Indizierung und die Suche ein Collection-Paar verwendet. Während Suchanfragen mit Hilfe der gerade live geschaltete Collection beantwortet werden, wird aktualisierter Content in die zweite, offline geschaltete Collection indiziert. Ein solches Collection-Paar wird umschaltbare Collection genannt.

Ist auf dem Live-Server die Template Engine nicht verfügbar, so lassen sich die Live-Server-Collections auch mit dem Content Manager erzeugen. Sofern dies in der Systemkonfiguration eingeschaltet wurde, werden die mit dem Befehl `exportSubtree` exportierten Dokumente automatisch indiziert.

2.3.2 Indizierte Daten

Auf dem Redaktionssystem indiziert die Search Cartridge die Versionen von Dateien sowie einige wichtige Dateifelder. Es ist konfigurierbar, welche Arten von Versionen indiziert werden sollen, wobei eine beliebige Kombination aus Arbeitsversionen, freigegebenen und archivierten Versionen wählbar ist.

Auf dem Live-Server dagegen werden von der Template Engine die UTF-8-kodierten exportierten Dokumente mit ihren Metadaten indiziert, bevor die konfigurierte Export-Kodierung auf sie angewendet wird. Bei Dateien, zu deren Export auch Daten anderer Dateien herangezogen werden (Framesets, Layoutdateien für den Hauptinhalt), werden nur die Metadaten der Hauptdatei indiziert.

Framesets und ihre Frames werden gemeinsam als ein einziges Dokument indiziert. Daher wird im Suchergebnis das Frameset geliefert, wenn ein dazu gehörendes Frame auf eine Suchanfrage passt. Steht die Template Engine auf dem Live-Server nicht zur Verfügung, so kann der Content Manager die Indizes beim statischen Export erzeugen.

Da die Search Cartridge nicht nur die Versionsfelder, sondern auch die wichtigsten Dateifelder indiziert, erhält man auch dann Suchergebnisse, wenn man nach Feldern wie dem Dateinamen oder dem Dateiformat sucht.

In den Werten von Feldern, die HTML-Text enthalten (wie `body`), bewirken die SGML-Kommentare `<!-- noindex -->` und `<!-- /noindex -->`, dass der Text zwischen diesen beiden Kommentaren nicht indiziert wird. Kommentare werden grundsätzlich nicht indiziert.

Der Content Manager indiziert eine Version für die Suche im Redaktionssystem, wenn sich der Wert eines indizierten Datei- oder Versionsfeldes oder der Dateistatus durch Workflowaktionen wie *Freigeben* oder *Zurückziehen* ändert. Für die Suche auf dem Live-System indiziert wahlweise die Template Engine oder der Content Manager die Webdokumente während sie exportiert werden.

2.3.3 Dokumentzonen und -felder

Der Search Engine Server erhält die zu indizierenden Daten einer Version oder eines Webdokuments als XML-Dokument in einem Request. In einem solchen Dokument entspricht jedes Feld einem XML-

Element. So wird beispielsweise das zusätzliche Versionsfeld `abstract` folgendermaßen in der erzeugten XML-Datei abgelegt:

```
<abstract>Zusammenfassung des Dokuments</abstract>
```

Sobald der Search Engine Server dem Autonomy-Such-Modul das zu indizierende Dokument übergeben und das Such-Modul es indiziert hat, entsprechen die indizierten Felder der CMS-Datei so genannten Zonen. Zonen sind benannte durchsuchbare Dokumentbereiche.

Man kann Suchanfragen explizit auf eine oder mehrere Zonen beschränken, um nach Dokumenten zu suchen, die den Suchbegriff in diesen Zonen enthalten. Eine solche Suche wird "attributiert" genannt, weil sie nicht das gesamte Dokument erfasst, sondern nur ausgewählte Bereiche. Bei jeder Suchanfrage, die sich nicht explizit auf bestimmte Zonen bezieht, werden stets alle Zonen durchsucht.

Während Dokumentzonen es ermöglichen, Dokumentbestandteile gezielt zu durchsuchen, dienen Dokumentfelder dazu, das Suchergebnis je gefundenem Dokument mit den Informationen anzureichern, die man auf den Ergebnisseiten anzeigen möchte. So wird in der Standardkonfiguration das Versionsfeld `title` nicht nur zu einer Zone, sondern sein Inhalt wird ebenfalls im Dokumentfeld `title` gespeichert. Dadurch können die Titel der Dokumente von der Search Cartridge in die Suchergebnisse aufgenommen und von den Clients zu beliebigen Zwecken verwendet werden.

Dokumentzonen und -felder sind frei konfigurierbar (siehe Collections konfigurieren). Eine Version einer CMS-Datei kann beliebig viele Versionsfelder haben, die bei der Indizierung zu ebenso vielen Dokumentzonen werden, sofern durch die Konfiguration nicht Zonen von der Indizierung ausgeschlossen oder nur bestimmte Zonen zugelassen werden.

Alle indizierten Dokumente haben dagegen immer alle Dokumentfelder. Wenn die Konfiguration beispielsweise vorsieht, dass der Inhalt einer Zone in einem bestimmten Feld gespeichert werden soll, so hat das indizierte Dokument dieses Feld auch dann, wenn die betreffende Zone nicht in dem Dokument vorkommt. Das Feld bleibt in diesem Fall leer.

Ein Client, der eine Suchanfrage an den Search Engine Server sendet, kann in seinem Request explizit die Felder angeben, deren jeweiligen Inhalt er im Suchergebnis erhalten möchte (siehe Suchanfragen). Die Zonen und Felder, die es in der Standardkonfiguration gibt, sind im Abschnitt Content-Suche aufgeführt.

2.3.4 Vorverarbeitung bei der Indizierung

Vor der eigentlichen Indizierung eines Dokuments können Sie den Search Engine Server jedes Dokument von einem Skript oder enem Programm, dem so genannten Präprozessor, vorverarbeiten lassen. Mit einem Präprozessor lassen sich die zu indizierenden Dokumente beispielsweise um Informationen erweitern, die nicht in den Versionen oder den Dateien gespeichert sind. Die Vorverarbeitung ist auch hilfreich, wenn Dokumente umkodiert werden müssen, weil ihr Format nicht UTF-8 ist.

Der Search Engine Server sendet die zu indizierenden Dokumente unmodifiziert zum Präprozessor, d.h. das Skript oder Programm erhält den ursprünglichen Indizierungs-Request. Nachdem der Präprozessor die Daten verarbeitet hat, sendet er sie zurück zum Search Engine Server, der sie an das Autonomy-Suchmodul zum Zwecke der Indizierung weitergibt.

2.3.5 Identifizierung indizierter Versionen

Das Suchmaschinenmodul ordnet einem Dokument bei der Indizierung einen eindeutigen Bezeichner – eine Dokument-ID – zu. Diese ID wird als Feld im Suchindex gespeichert und bei einer Suche im Suchergebnis zurückgegeben. Die IDs werden dem Such-Modul vom Search Engine Server übergeben, der sie wiederum vom jeweiligen Client erhält. Es ist also der Client, der darüber entscheidet, welches CMS-Datei- oder Versionsfeld als Dokument-ID verwendet werden soll.

Während der Content Management Server als Dokument-Identifikatoren Versions-IDs verwendet, handelt es sich bei der Template Engine um Datei-IDs. Daher entspricht auf dem Redaktionssystem die ID eines indizierten Dokuments einer Versions-ID, während auf dem Live-System die Dokument-ID eine Datei-ID ist.

Die Dokument-ID ermöglicht es einem Client (etwa der Benutzeroberfäche des Redaktionssystems, oder der Template Engine), zusätzliche Informationen über das Dokument zu ermitteln. Zusätzlich zu dieser ID werden auch andere wichtige Inhalte voreingestellt als Dokumentfelder indiziert, die Pfade der CMS-Dateien und die Titel der betreffenden Dateiversionen. Diese Informationen kann ein Client aus den Suchergebnissen extrahieren, um auf den Suchergebnisseiten die Titel der gefundenen Dokumente mit den entsprechenden Webseiten zu verlinken.

2.4 Content-Suche

Der Search Engine Server indiziert auf dem Redaktionssystem die Versionen von CMS-Dateien. Auf dem Live-System werden die exportierten Webdokumente indiziert. Auf beiden Systemen werden zusätzlich die wichtigsten Dateifelder indiziert.

In seinen Suchergebnissen liefert der Search Engine Server je gefundenem Dokument einen Datensatz. Jeder Datensatz besteht aus einer Menge von Dokumentfeldern, die bei der Indizierung mit den entsprechenden Werten belegt wurden (siehe Content-Indizierung). Die Dokumentfelder sind konfigurierbar (siehe Collections konfigurieren).

Neben den wichtigen Versions- und Dateieigenschaften steht in der Standardkonfiguration auch die Gewichtung eines gefundenen Dokuments als Dokumentfeld (*score*) zur Verfügung. Die Gewichtung eines Dokuments sagt aus, wie relevant ein Dokument in Bezug auf die Suchanfrage ist, bei der es gefunden wurde. Die Gewichtung wird mit einer Zahl im Bereich von 0 bis 100 ausgedrückt.

In der folgenden Tabelle sind die je Treffer gelieferten Dokumentfelder aufgeführt, unterschieden nach Redaktions- und Live-System:

Dokumentfeld	Redaktionssystem	Live-System
`collection`	•	•
`score`	•	•
`docId` (aus Versions-/Datei-ID)	•	•
`lastChanged`	•	•
`objId` (aus Datei-ID)	•	
`title`	•	•
`visiblePath`	•	•

Die Inhalte der Felder `lastChanged` und `title` werden vom Autonomy-Such-Modul mit den Werten aus den gleichnamigen Zonen belegt. Als Zonen werden die im Folgenden aufgeführten Versions- und Dateifelder indiziert:

Indizierte Dateifelder	Redaktionssystem	Live-System
`name`	•	•
`objClass`	•	•
`objType`	•	•
`suppressExport`	•	
`visiblePath`	•	•
`workFlowName`	•	
die Dateizugriffsrechte	•	

Indizierte Versionsfelder	Redaktionssystem	Live-System
`blobLength` (ab Version 6.5)	•	•
`exportBlob` (exportierte Datei, nicht bei Bildern)	•	•
`contentType`	•	•
`lastChanged`	•	•
`mimeType`	•	•
`state`	•	
`title`	•	•
`validFrom`	•	•
`validUntil`	•	•
kundenspezifische Felder (bis auf Unterschriften- und Linklistenfelder)	•	•

Die Zone und das Feld `visiblePath` sind auf dem Redaktionssystem leer. Auf dem Live-System enthalten sie den Pfad zum Dokument.

Bei kundenspezifischen Feldern vom Typ *Mehrfachauswahl* (`multienum`) wird jeder einzelne Feldwert als Zone mit dem Namen des Feldes indiziert. Hat ein solches Feld also mehrere Werte, so wird für jeden Wert eine Zone gleichen Namens indiziert.

Dies gilt auch für die Dateirechte: Jede Benutzergruppe, die ein bestimmtes Recht hat, wird als Zone mit dem Namen des Dateirechts indiziert. Eine Ausnahme hiervon ist das Liveserver-Leserecht (`permissionLiveServerRead`). Die entsprechende Zone enthält die Namen aller Gruppen, denen dieses Recht erteilt wurde. Bei Dokumenten, die keiner Zugriffsbeschränkung unterliegen, wird die Zone `noPermissionLiveServerRead` mit dem Inhalt `free` indiziert.

In Suchanfragen kann man mit dem Operator IN gezielt nach Dokumenten suchen, die den Suchbegriff in einer oder mehreren Zonen enthalten.

2.4.1 Mehrere Parser

Die Infopark Search Cartridge unterstützt Suchanfragen in mehreren Formaten. Für jedes Format ist ein sogenannter Parser zuständig. Ein Parser hat die Aufgabe, Eingaben – in diesem Falle Suchanfragen – zu analysieren und in ein allgemeines internes Format umzuwandeln, so dass die der Eingabe entsprechenden Aktionen ausgeführt werden können.

Suchanfragen können als Freitext, in expliziter Syntax oder in einfacher Syntax gestellt werden. Voreingestellt wird der Parser für Anfragen in einfacher Syntax verwendet.

Mit dem Freitext-Parser können Suchanfragen in geschriebener Sprache gestellt werden, d. h. ohne Operatoren zu verwenden (beispielsweise „freiwillige feuerwehr in ländlichen regionen"). Die Infopark Search Cartridge wandelt Freitext-Anfragen intern in Suchanfragen um, indem sie unwichtige Wörter wie Artikel, Konjunktionen und Präpositionen (so genannte Stoppwörter) entfernt, und die Besonderheiten der natürlichen Sprache wie Nominalphrasen und die Reihenfolge der Wörter berücksichtigt. (Siehe auch die Informationen zum Operator FREETEXT.)

Anfragen in expliziter oder einfacher Syntax dagegen analysiert die Suchmaschine unter Berücksichtigung von Operatoren, mit denen Suchbegriffe kombiniert werden können. Mehr über die einfache und explizite Syntax sowie die Operatoren erfahren Sie in den Abschnitten Die Syntax der Suchanfragen und Operatoren und Modifikatoren.

2.4.2 Vorverarbeitung und Nachbereitung

Beim Search Engine Server gibt es die Möglichkeit, jede Suchanfrage, die er von einem Client (also auch dem Content Manager oder der Template Engine) erhält, von einem Präprozessor bearbeiten zu lassen, bevor die Anfrage dem Autonomy-Such-Modul übergeben wird. Mit einem solchen Präprozessor ließen sich beispielsweise Suchanfragen um Begriffe oder Operationen erweitern oder nicht gestattete Suchbegriffe aus den Anfragen entfernen. Da die Suchanfrage, die der Präprozessor erhält, das ursprünglich an den Search Engine Server gesendete XML-Dokument ist, muss der Präprozessor in der Lage sein, XML-Dokumente zu verarbeiten.

Analog zur Vorverarbeitung funktioniert die Nachbearbeitung von Suchergebnissen. Hat der Search Engine Server dem Such-Modul eine (gegebenenfalls vorverarbeitete) Suchanfrage übergeben, so liefert das Modul ein Suchergebnis. Dieses Ergebnis kann durch einen Postprozessor bearbeitet werden, um beispielsweise die Liste der gefundenen Dokumente zu erweitern oder zu kürzen oder jeden Treffer mit einem zusätzlichen Dokumentfeld zu versehen, dessen jeweiliger Wert der Postprozessor berechnet hat.

2.4.3 Zeichensätze

Die Search Cartridge arbeitet mit der Zeichenkodierung UTF-8. Damit sie Suchergebnisse (d. h. vor allem die Inhalte von Dokumentfeldern) UTF-8-kodiert ausliefern kann, müssen die indizierten Dokumente ebenfalls diesen Zeichensatz haben. Dies wird vom Content Manager und von der Template Engine sichergestellt.

3

3 Konfiguration und Administration

3.1 Den Search Engine Server ausführen

Der Search Engine Server wird bei der Installation des CMS-Fiona-Packages mitinstalliert. Daher wird hier nur kurz gezeigt, wie der er gestartet wird.

Unter Linux und Solaris kann der Search Engine Server folgendermaßen von dem Benutzer ausgeführt werden, unter dessen Login das CMS installiert wurde:

```
~/instance/default/bin/rc.npsd start SES
```

Falls Sie einen Search Engine Server einer anderen Instanz starten möchten, ersetzen Sie `default` durch den betreffenden Verzeichnisnamen. Als Parameter des Startskripts `rc.npsd` können Sie statt `start` auch `stop`, `restart` oder `status` angeben, um den Server anzuhalten, neu zu starten bzw. seinen Status ausgeben zu lassen.

Das Verzeichnis `instance` befindet sich unterhalb des CMS-Installationsverzeichnisses.

Bitte beachten Sie unter Windows, dass Sie als Administrator an dem Rechner angemeldet sein müssen, auf dem der Search Engine Server gestartet werden soll. Sie können ihn aus dem Windows-Startmenü heraus ausführen.

Beachten Sie bitte, dass der Search Engine Server, der Content Management Server und die Template Engine aufeinander abgestimmt werden müssen, um in einer Fiona-Installation miteinander kommunizieren zu können. Auch in Bezug auf die zu verwendenden Collections sind Anpassungen erforderlich, wenn Sie mehr oder andere als die voreingestellten Collections verwenden möchten.

3.2 Suche und Indizierung konfigurieren

Der Search Engine Server selbst sowie sein Verhalten bei der Suche und Indizierung können mit den folgenden Systemkonfigurationseinträgen konfiguriert werden:

- `server.ses`
- `indexing`
- `searching`
- `tuning`

Die entsprechenden Konfigurationsdateien finden Sie im Verzeichnis `config` unterhalb des jeweiligen Instanzenverzeichnisses, also beispielsweise unter `instance/default/config`.

Wie alle CMS-Applikationen liest auch der Search Engine Server die Konfiguration nur beim Start ein. Daher muss er neu gestartet werden, nachdem Änderungen an der Konfiguration vorgenommen wurden.

3.3 Einen externen Präprozessor einbinden

3.3.1 Allgemeines

Mit einem externen Präprozessor können Dokumente geändert werden, bevor sie indiziert werden. Dies ermöglicht es beispielsweise, Binärdaten in Text umzuwandeln oder Metadaten (etwa von Bildern) zu extrahieren oder zu erzeugen und anschließend zu indizieren. Dadurch können die betreffenden Dokumente auch oder besser über die Suche gefunden werden.

Dokumente beliebiger MIME-Typen können den jeweils zu verwendenden externen Präprozessoren über den Abschnitt `indexing` in der Systemkonfiguration zugeordnet werden. Als ein externer Präprozessor kann jedes geeignete Programm angegeben werden. Optional können einem solchen Programm Argumente übergeben werden.

3.3.2 Funktionsweise

Das Präprozessor-Programm erhält vom Search Engine Server das zu indizierende Dokument in Form eines serialisierten XML-Dokuments über `stdin`. Der Präprozessor modifiziert dieses XML-Dokument in der gewünschten Weise und gibt es anschließend an den Search Engine Server auf `stdout` zurück. Dieser indiziert nun das modifizierte Dokument. Beispiel:

Ursprüngliche Daten:

```
<ses-indexDoc docId="2148" collection="cm-contents"
    mimeType="application/vnd.ms-excel">
<title encoding="plain">Ein Beispiel mit Excel-Daten</title>
<keyword encoding="plain">Beispiel</keyword>
<blob encoding="stream" mimeType="application/vnd.ms-excel">
  /Fiona_671/instance/default/tmp/externalPreprocessor/1.dat
</blob>
</ses-indexDoc>
```

Geänderte Daten:

```
<ses-indexDoc docId="2148" collection="cm-contents"
    mimeType="application/vnd.ms-excel">
<title encoding="plain">Excel-Daten als Text</title>
<keyword encoding="plain">Beispiel</keyword>
<blob encoding="stream" mimeType="text/plain">
  /Fiona_671/instance/default/tmp/text_data.dat
</blob>
</ses-indexDoc>
```

Das XML-Dokument enthält die zu indizierenden Felder (die Namen der XML-Elemente) und deren Werte (die Inhalte der XML-Elemente). Ein Feldwert kann direkt (encoding: `plain`) oder kodiert angegeben sein. Das Encoding wird über das Tag-Attribut `encoding` des Feld-Elements bestimmt, das folgende Werte haben kann:

- `plain`: Der Feldwert entspricht dem Inhalt des XML-Elements.
- `base64`: Der Feldwert ergibt sich aus dem base64-dekodierten Inhalt des XML-Elements.

- `stream`: Der Feldwert ist in der Datei enthalten, deren Pfad im Inhalt des XML-Elements angegeben ist.

Ab Version 6.7.1 wird bei allen Kodierungen außer `plain` zusätzlich der MIME-Typ des Dokuments im Feld-Element-Attribut `mimeType` übermittelt. Wenn der MIME-Typ während der Vorverarbeitung geändert wird, muss auch das `mimeType`-Attribut auf den resultierenden MIME-Typ gesetzt werden. Ist die Kodierung nicht `plain`, wird ein Feldwert nur dann indiziert, wenn der angegebene MIME-Typ dem Muster `text/*` genügt. Mit anderen Worten: wenn ein Präprozessor base64-kodierte oder per Streaming inkludierte Feldwerte liefert, muss er deren MIME-Typ auf einen Text-Typ setzen.

Bis einschließlich Version 6.7.0 dürden die vorverarbeiteten Feldwerte nicht kodiert sein (`encoding="plain"`). Kodierte Feldwerte werden nicht indiziert.

3.3.3 Konfiguration

Der zu verwendende Präprozessor kann zusammen mit den MIME-Typen, für die er zuständig ist, sowie den ihm zu übergebenden Argumenten in der Konfigurationsdatei `indexing.xml` eingetragen werden. Der betreffende Abschnitt sieht beispielsweise so aus:

```
...
<contentPreprocessors type="list">
  <preprocessor>
    <mimeTypes type="list">
      <mimeType>application/pdf</mimeType>
    </mimeTypes>
    <processor type="external">
      bin/tclsh
    </processor>
    <processorArguments type="list">
      <argument>/Fiona_671/instance/default/script/custom/pdf2TxtWrapper.tcl</argument>
    </processorArguments>
  </preprocessor>
  ...
</contentPreprocessors>
...
```

Hier sind als auszuführendes Programm der Tcl-Interpreter und als ein `argument` in `processorArguments` der Name des von ihm auszuführenden Skripts, `pdf2TxtWrapper.tcl`, angegeben. Das Skript kann nicht beim Start des Search Engine Servers geladen werden, sollte also nicht im `serverCmds`- oder `clientCmds`-Verzeichnis liegen.

Das folgende als Beispiel aufgeführte Skript `pdf2TxtWrapper.tcl` zeigt, wie ein im Feld `blob` enthaltenes PDF-Dokument eingelesen und zu Text konvertiert werden kann. Beachten Sie bitte, dass die Search Cartridge PDF-Dateien auch ohne Zuhilfenahme eines Präprozessors indizieren kann.

```
# Libraries
package require dom
package require base64
proc safeInterp {args} {}
source [file join [file dirname [info script]]\
    ../../../share/script/common/clientCmds/util.tcl]

# Read Data
set xmlRequest [read stdin]

# Parse XML
set docNode [::dom::DOMImplementation parse $xmlRequest]
set rootNode [::dom::document cget $docNode -documentElement]

# Select and handle element "blob"
```

```
set blobElement [lindex [::dom::selectNode $rootNode descendant::blob] 0]
array set attributes [array get [$blobElement cget -attributes]]
set blobTextNode [$blobElement cget -firstChild]
if {$blobTextNode ne ""} {
  set value [$blobTextNode cget -nodeValue]
  if {$value ne ""} {
    switch $attributes(encoding) {
      plain {
        # shouldn't happen with pdf
        set blob $value
      }
      base64 {
        set blob [::base64::decode $value]
      }
      stream {
        set blobFile $value
      }
    }
    set deletePdfFile 0
    if {![info exists blobFile]} {
      set blobFile "/tmp/convert_me_[pid].pdf"
      writeFile $blobFile $blob
      set deletePdfFile 1
    }
    set textFile "/tmp/converted_[pid].txt"
    # convert using ps2ascii
    if {![catch {
      exec ps2ascii $blobFile $textFile
    }]} {
      # modify the dom tree
      $blobTextNode configure -nodeValue $textFile
      ::dom::element setAttribute $blobElement mimeType "text/plain"
      ::dom::element setAttribute $blobElement encoding stream
    }
    if {$deletePdfFile} {
      file delete -force $blobFile
    }
  }
}
set xmlToReturn [string trimright [::dom::DOMImplementation serialize $docNode] "\n"]
set lines [split $xmlToReturn "\n"]
if {[string match "<!D*" [lindex $lines 1]]} {
  set xmlToReturn [join [lreplace $lines 1 1] "\n"]
}
# return the (modified) xml data
puts -nonewline $xmlToReturn
```

3.4 Collections konfigurieren

Neue Collections werden auf der Grundlage von Konfigurationsdateien erzeugt, mit denen vor allem die zu indizierenden Dokumentzonen und die dabei zu füllenden Dokumentfelder festgelegt werden. Nur diejenigen Elemente in den zu indizierenden Dokumenten, die zu Dokumentzonen werden, können in Dokumentfelder übernommen werden (siehe auch Dokumentzonen und -felder).

Diese Eigenschaften von Collections werden durch sogenannte Style-Dateien gesteuert. Legt man eine Collection an (siehe Befehle zur Administration), so werden die vorgegebenen Style-Dateien im Verzeichnis `config/vdk/style` als Vorlage in das Verzeichnis `style` unterhalb des jeweiligen neuen Collection-Verzeichnisses kopiert.

Wenn Sie die Eigenschaften aller künftigen Collections ändern möchten, so können Sie die Zonen- und Felddefinitionen in den Style-Vorlagen ändern. Möchten Sie dagegen die Konfiguration einer neu angelegten, noch leeren Collection ändern, so können Sie die Style-Dateien dieser Collection bearbeiten. Die Konfiguration einer Collection, die indizierte Dokumente enthält, sollte nicht geändert werden, ohne sie vorher mit dem Befehl `purgeCollection` geleert zu haben. Änderungen

an den Konfigurationsdateien einer Collection dürfen nicht während eines Indizierungsvorgangs vorgenommen werden. Gegebenenfalls muss der Search Engine Server vorher angehalten werden.

Nachdem Sie eine neue Collection angelegt haben, sollten Sie die Regeln anpassen, nach denen die bei der Indizierung zu verwendende Collection ausgewählt wird (eine CMS-Datei kann immer nur in eine Collection indiziert werden). Dies ist mit dem Systemkonfigurationseintrag `indexing.incrementalExport.collectionSelection` möglich. Die Suche dagegen bezieht immer alle Collections ein, es sei denn, es werden die zu durchsuchenden Collections explizit in der Suchanfrage angegeben.

3.4.1 Dokumentzonen definieren

Dokumentzonen können in der Datei `style.xml` definiert werden. Es stehen darin die folgenden Elemente als Anweisungen zur Verfügung, um die Behandlung von XML-Tags (also Feldern in der Version) festzulegen:

Die folgende Anweisung bewirkt, dass alle XML-Tags im Dokument ignoriert werden und nur die Inhalte der XML-Elemente indiziert wird.

```
<ignore xmltag="*"/>
```

Mit der folgenden Anweisung wird das als `xmltag` angegebene XML-Element ignoriert und nur der Inhalt zwischen seinem Start- und Ende-Tag indiziert.

```
<ignore xmltag="section_1"/>
```

Die folgende Anweisung bewirkt, dass das angegebene XML-Element als Zone indiziert wird, wenn gleichzeitig die Anweisung `ignore xmltag="*"` vorhanden ist:

```
<preserve xmltag="section_1"/>
```

Die folgende Anweisung unterdrückt das gesamte angegebene XML-Element. Das Tag, die Attribute und der Inhalt werden nicht indiziert:

```
<suppress xmltag="section_1"/>
```

Die folgende Anweisung bewirkt, dass der Inhalt des angegebenen XML-Elements als Feld mit dem Namen `fieldname` indiziert wird. Ist `fieldname` nicht angegeben, wird als Feldname der Tag-Name verwendet. Ein bereits bestehender Feldinhalt wird überschrieben, wenn das optionale Attribut `index="override"` angegeben ist.

```
<field xmltag="column_2" fieldname="vdk_field_2" index="override"/>
```

Die als Zonen zu indizierenden Elemente können nach dem Ausschlussverfahren oder dem Einschlussverfahren festgelegt werden. Beim Ausschlussverfahren werden alle Elemente bis auf diejenigen indiziert, deren Name mit `<ignore xmltag="..."/>` angegeben wurde. Beim Einschlussverfahren dagegen schließt man zuerst alle Elemente mit `<ignore xmltag="*"/>` aus, um dann die zu indizierenden mit `<preserve xmltag="..."/>` explizit aufzuführen.

Bei beiden Verfahren können die Inhalte der Elemente (d. h. der Zonen) in Feldern abgelegt werden, so dass sie bei der Suche je ermitteltem Dokument im Suchergebnis zurückgegeben werden können. Es ist nicht möglich, ein Element zu ignorieren und gleichzeitig seinen Inhalt in einem Dokumentfeld abzulegen.

3.4.2 Dokumentfelder definieren

Bei Dokumentfeldern unterscheidet das Verity-Modul zwischen Standard- und benutzerdefinierten Feldern. Standardfelder werden in der Datei *style.sfl* , benutzerdefinierte in der Datei *style.ufl* konfiguriert. Wenn Sie Felder definieren möchten, um in den Suchergebnissen weitere Informationen über die Dokumente zur Verfügung zu haben, so beachten Sie bitte folgendes:

- Jedes Feld vergrößert den Speicherplatzbedarf proportional zur Anzahl der indizierten Dokumente.
- Felder werden bei der Indizierung gefüllt und können bei der Suche ins Suchergebnis aufgenommen werden. Diese Vorgänge dauern um so länger, je mehr Felder Sie definiert haben bzw. ins Suchergebnis aufnehmen lassen.

Die Namen der Standard-Dokumentfelder sind Vorgaben des Verity-Moduls, die Sie aus Gründen der Kompatibilität Ihrer Collections mit Collections aus anderen Quellen nicht ändern sollten. Stattdessen können Sie jedoch bei Bedarf Aliasnamen für Dokumentfelder vergeben.

Das Format der beiden Dateien `style.sfl` und *style.ufl* ist identisch. Sie werden vom Verity-Modul indirekt über die Datei `style.ddd` eingelesen. Felder werden in diesen Dateien nach dem folgenden Muster definiert:

```
data-table: nps{
    varwidth: objId       dda
    /minmax = yes
    /alias = objectId
    autoval:  collection  DBNAME}
```

Mit dem Schlüsselwort `data-table` wird der Name einer Tabelle des Verity-Moduls innerhalb eines Datensegments festgelegt. Im obigen Beispiel lautet der Name `nps`.

Jedes Feld hat einen Speichertyp, der bestimmt, woher der Wert des Feldes stammt und wie er gespeichert wird. Es gibt die folgenden Speichertypen:

Speichertyp	Bedeutung und Syntax
`autoval`	Weist dem Feld einen internen Wert des Verity-Moduls zu. **Syntax:** `autoval:` *FeldName* `DBNAME|DBPATH` *FeldName*: Name des Feldes. `DBNAME`: Name der Collection. `DBPATH`: Pfad der Collection.
`constant`	Weist dem Feld einen konstanten Wert zu. **Syntax:** `constant:` *Feldname Datentyp Wert* *FeldName*: Name des Feldes. *Datentyp*: Einer der unten aufgeführten Datentypen. *Wert*: Der Wert des Feldes. Enthält er Leerzeichen, so muss er in Anführungszeichen eingeschlossen werden.
`worm`	Weist dem Feld einen Wert aus einer Dokumentzone zu. Der Wert wird anschließend nicht mehr geändert (*write once, read many times*).

	Syntax: `worm: Feldname Datentyp` *FeldName*: Name des Feldes. *Datentyp*: Einer der unten aufgeführten Datentypen.
`fixwidth`	Weist dem Feld einen Wert aus einer Dokumentzone zu. Der Wert hat eine feste Länge und wird gegebenenfalls gekürzt. **Syntax:** `fixwidth: Feldname Länge Datentyp` *FeldName*: Name des Feldes. *Länge*: Die maximale Länge des Feldwertes in Zeichen. *Datentyp*: Einer der unten aufgeführten Datentypen. **Modifikator:** `/minmax = yes|no`: Beschleunigt die Suche in Werten dieses Feldes, insbesondere bei großen Collections und wenn die Feldwerte sich in einem spezifischen Bereich befinden.
`varwidth`	Weist dem Feld einen Wert beliebiger Länge aus einer Dokumentzone zu. **Syntax:** `varwidth: Feldname Speicherort` *FeldName*: Name des Feldes. *Speicherort*: Eine Zeichenkette aus 3 Zeichen, nicht mit _ oder `di` beginnend. **Modifikator:** `/minmax = yes|no`: s. `fixwidth`.

Feldnamen können aus bis zu 128 alphanumerischen Zeichen bestehen und dürfen nicht mit einem Unterstrich beginnen. Geben Sie als Speicherort beim Speichertyp `varwidth` beispielsweise `dda` an.

Jedes Dokumentfeld kann einen der folgenden Datentypen haben:

Datentyp	Bedeutung
`text`	Das Feld enthält ASCII-Zeichen.
`date`	Ein internes Datumsformat, das Datums- und Zeitangaben im Bereich von 1904 bis 2037 aufnimmt. Dieser Datentyp ist nicht kompatibel mit dem Format von Datumsangaben der Search Cartridge.
`signed-integer`	Eine vorzeichenbehaftete Ganzzahl. Der Wertebereich ist abhängig von der Größe des Feldes. 1 Byte: -128 bis 127 Bytes: -32768 bis 32767 4 Bytes: -2e31 bis 2e31 -1
`unsigned-integer`	Eine nicht vorzeichenbehaftete Ganzzahl. Der Wertebereich ist abhängig von der Größe des Feldes. 1 Byte: 0 bis 255 2 Bytes: 0 bis 65535 4 Bytes: 0 bis 2e32 -1

Dokumentfelder, die mit der Option `minmax` angelegt wurden, können bis zu 256 Bytes aufnehmen. Ist der zu speichernde Wert größer, so werden nur die ersten 256 Bytes im Feld abgelegt. Mit der Option `alias` kann ein alternativer Name angegeben werden, unter dem das Feld in Such-Requests angesprochen werden kann.

3.5 Stoppwörter definieren

Stoppwörter (*engl.* "stopwords") sind Wörter, die in Texten sehr häufig vorkommen (wie Artikel, Präpositionen etc.) und deshalb zu unnötig vielen und wenig relevanten Suchergebnissen führen, wenn man nach ihnen sucht.

Bei Suchanfragen, in deren Suchtext Stoppwörter vorkommen – der jedoch nicht ausschließlich aus Stoppwörtern besteht –, lassen sich die Treffer unterdrücken, die nur auf gefundene Stoppwörter zurückzuführen sind. Gehen Sie hierzu bitte folgendermaßen vor:

1. Tragen Sie die gewünschten Stoppwörter in die Datei *Installationsverzeichnis*/3rdparty/vdk/common/uni/vdk30.stp ein. Geben Sie je Zeile genau ein Wort an. (Als Vorlage können die Stoppwörter in den Dateien *Installationsverzeichnis*/3rdparty/vdk/common/uni/stopword.*xx* verwendet werden.)
2. Starten Sie den SES neu.

Die definierten Stoppwörter werden nun in FREETEXT-Suchanfragen beachtet. Beispiel: Wenn "der" als Stoppwort definiert ist, findet die Suchanfrage

```
<#FREETEXT> "der Teddy"
```

nur Dokumente, die das Wort "Teddy" enthalten.

Sofern eine FREETEXT-Suchanfrage ausschließlich Stoppwörter enthält, werden diese nicht ignoriert.

3.6 Synonyme definieren

Mit der Thesaurus-Funktion der Suchmaschine von Autonomy können Synonyme definiert werden. Dadurch werden bei einer Suche auch dann Treffer erzielt, wenn die Inhalte statt des gesuchten Begriffs einen Begriff mit ähnlicher oder gleicher Bedeutung – ein Synonym – enthalten.

Einen Thesaurus können Sie folgendermaßen erstellen und installieren:

1. Definieren Sie den Thesaurus als Textdatei vdk30.txt. Beispiel:

   ```
   $control:1
   synonyms:
   {
   list: "Veröffentlichung,Publikation,Zeitschrift,Zeitung,Journal"
   list: "Gesetz,Verordnung,Vorschrift,Bestimmung"
   }
   $$
   ```

   Die Definition besteht aus jeweils einer Liste von Synonymen pro Zeile.
2. Kompilieren Sie den Thesaurus mit mksyd. Bei Verwendung der Locale (länderspezifische Einstellungen) uni sind unter Linux folgende Schritte erforderlich:

   ```
   > export PATH=$PATH:Installationsverzeichnis/3rdparty/vdk/_ilnx21/bin
   > export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:Installationsverzeichnis/3rdparty/vdk/
   _ilnx21/bin
   > mksyd -locale uni -f vdk30.txt -syd vdk30.syd
   ```

   Unter Windows können Sie den Suchpfad mit

```
set PATH=%PATH%;Installationsverzeichnis\3rdparty\vdk\_ilnx21\bin
```

erweitern und dann das angegebene Kommando ausführen.

3. Um den Thesaurus zu installieren, kopieren Sie ihn bitte an die von der Suchmaschine erwartete Stelle:

```
> cp vdk30.syd Installationsverzeichnis/3rdparty/vdk/common/uni
```

Unter Windows verwenden Sie bitte `copy` statt `cp` und ersetzen die Schrägstriche durch Backslashes.

4. Starten Sie den SES neu:

```
> Installationsverzeichnis/instance/Instanzname/bin/rc.npsd restart SES
```

Die definierten Synonyme können nun in Suchanfragen mit Hilfe des `THESAURUS`-Operators verwendet werden. Beispiel: Die Suchanfrage

```
<#MANY><#THESAURUS>"Vorschrift"
```

findet alle Dokumente, die das Wort "Vorschrift" oder eines der definierten Synonyme dieses Wortes enthalten.

3.7 Bindestriche als Leerraum behandeln

In der Ländereinstellung `uni` werden Bindestriche im Content von der Suchmaschine voreingestellt wie alphabetische Zeichen behandelt. Enthält ein Dokument beispielsweise den Begriff `Heinz-Müller-Stiftung`, so wird das Dokument mit den Suchbegriffen `Müller*` und `Müller-Stiftung*` nicht gefunden.

Um dieses Verhalten zu korrigieren, muss der Bindestrich aus der Liste der zusätzlichen alphabetischen Zeichen entfernt werden. Diese Liste befindet sich in der Konfigurationsdatei `ctype.cfg` im Verzeichnis *`Installationsverzeichnis`*`/3rdparty/vdk/common/uni`. Voreingestellt enthält sie den Unterstrich, den Bindestrich und das kaufmännische Und-Zeichen, als hexadezimale Zeichenwerte:

```
ALPHA: 0x26,0x2d,0x5f
```

Entfernen Sie den Bindestrich (`0x2d`) aus der Liste, speichern Sie die Konfigurationsdatei `ctype.cfg` und starten Sie den SES neu. Anschließend muss der Content mit dem Tcl-Befehl `indexAllObjects` des Content Managers neu indiziert werden.

3.8 Die Tcl-Schnittstelle

3.8.1 Befehle zur Administration

Collections sind der Datenbestand der Infopark Search Cartridge. Sie enthalten im Wesentlichen die Indizes und Wörterlisten, die bei der Indizierung von Dokumenten entstehen.

Umschaltbare Collections – solche, die auf dem Live-Server verwendet werden – befinden sich im Verzeichnis *export/offline/collections* unterhalb des Instanzenverzeichnisses, also beispielsweise in `instance/default/export/offline/collections`. Darin hat jede Collection ein eigenes Verzeichnis, das den Namen der Collection trägt.

Nicht umschaltbare Collections werden vorwiegend auf dem Redaktionssystem eingesetzt. Auf dem Live-System werden sie nur verwendet, wenn dort keine Template Engine verfügbar ist. Der Speicherort nicht umschaltbarer Collections ist das Verzeichnis `data/ses/collections` unterhalb des Instanzenverzeichnisses von Fiona. Wie umschaltbare Collections haben auch nicht umschaltbare dort ein Unterverzeichnis, dessen Name dem Namen der Collection entspricht.

Das Suchmaschinenmodul von Autonomy (ehem. Verity) verwendet bei der Erzeugung von Collections sogenannte Style-Dateien. Legt man Collections mit dem in diesem Abschnitt beschriebenen Tcl-Kommando `createCollection` an, so werden die mitgelieferten Style-Dateien verwendet, die sich im Verzeichnis `config/vdk/style` befinden. Gibt es allerdings unterhalb dieses Verzeichnisses ein Verzeichnis mit dem Namen der anzulegenden Collection, so werden die darin enthaltenen Style-Dateien verwendet. Diese Dateien werden in das Verzeichnis `style` unterhalb des jeweiligen Collection-Verzeichnisses kopiert.

Selbstverständlich können Sie auch das Autonomy-Programm `mkvdk` verwenden, um Collections anzulegen. Das Tool befindet sich neben anderen Programmen zur Administration im Verzeichnis `3rdparty/vdk/_ilnx21/bin` (Unix) bzw. `3rdparty/vdk/_nti40/bin` (Windows).

Die Ländereinstellungen (*engl.* "locale") und Autonomy-Konfigurationsdateien sind im Verzeichnis `3rdparty/vdk/common` abgelegt. Jede Locale hat dort ein Verzeichnis, dessen Name dem Namen der Locale entspricht.

Zur Administration der Collections steht Ihnen eine Reihe von Tcl-Befehlen zur Verfügung, die Sie im sogenannten Single-Modus verwenden können. Im Single-Modus arbeitet der Search Engine Server nicht als Server, sondern als Kommandozeilenprogramm.

Bitte beachten Sie, dass der Search Engine Server angehalten werden muss, bevor Kommandos ausgeführt werden, die eine oder mehrere Collections modifizieren.

Sie führen den Search Engine Server im Single-Modus aus, indem Sie in einer Shell die ausführbare Datei mit dem Kommandozeilenargument `-single` aufrufen. Unter Linux und Solaris geschieht dies aus dem `bin`-Verzeichnis der Instanz mit

```
./SES -single
```

und unter Windows mit:

```
SES -single
```

Der Search Engine Server zeigt nun einen Prompt (eine Eingabeaufforderung) an. Hier können Sie alle Standard-Tcl-Kommandos und die im Folgenden aufgeführten Kommandos des Servers verwenden. Die meisten dieser Kommandos rufen das Autonomy-Programm `mkvdk` auf. Mit ihnen können administrative Aufgaben einfacher als mit `mkvdk` selbst erledigt werden.

3.8.2 aboutCollection

Verfügbar für: Search Engine Server

Aufgabe: Dieses Kommando gibt Informationen über die angegebene Collection (wie beispielsweise die Anzahl der indizierten Dokumente oder die verwendeten Ländereinstellungen) aus.

Syntax

`aboutCollection` *`collectionName`*

Funktionsparameter

- *`collectionName`*: Der Name der Collection, über die Informationen ausgegeben werden sollen. Bei umschaltbaren Collections bezieht sich dieser Befehl auf die offline geschaltete Collection.

Rückgabewert: Informationen über die Collection in Textform. Existiert die angegebene Collection nicht, so wird die Fehlermeldung "The collection '*`collectionName`*' does not exist" ausgegeben.

Beispiel (Auszug)

```
SES>aboutCollection sportsNews
mkvdk - Verity, Inc. Version 5.0.1 (_ilnx21, Aug 6 2003)
Collection about resources:
  Last Purge Date: 19-Oct-2010 02:16:50 pm
  Creation Date: 16-Oct-2010 12:11:29 pm
  Modification Date: 19-Oct-2010 02:20:57 pm
  Last Squeeze Date: 0000000000
  Number of Documents: 64071
  Collection Creator: INFOPARK AG
  Collection Name: sportsNews
  ...
mkvdk done
```

3.8.3 backupCollection

Verfügbar für: Search Engine Server

Aufgabe: Der Befehl legt eine Kopie einer Collection an.

Syntax

`backupCollection` *`collectionName targetDir`*

Funktionsparameter

- *`collectionName`*: Der Name der zu sichernden Collection. Bei umschaltbaren Collections bezieht sich dieser Befehl auf die offline geschaltete Collection.
- *`targetDir`*: Das Zielverzeichnis der neuen Collection, relativ zum Verzeichnis, das dem Collection-Verzeichnis übergeordnet ist. Unter Windows darf als Trenner in Pfaden nicht der Rückwärts-Schrägstrich (*engl.* "backslash", '\') verwendet werden. Verwenden Sie stattdessen zwei Backslashes oder den Schrägstrich.

Rückgabewert: keiner.

Beispiel

```
SES>backupCollection sportsNews liveSportsNews
```

3.8.4 createCollection

Verfügbar für: Search Engine Server

Aufgabe: Dieser Befehl erzeugt eine neue Collection.

Syntax

`createCollection` *`collectionName`* [*`switchable`*]

Funktionsparameter

- *`collectionName`*: Der Name der anzulegenden Collection. Dies muss ein zulässiger Dateiname sein. Eine Collection gleichen Namens, ob umschaltbar oder nicht, darf nicht bereits existieren.
- *`switchable`*: Der Wert kann 0 oder 1 sein. Er gibt an, ob die zu erzeugende Collection umschaltbar sein soll (1) oder nicht (0, die Voreinstellung). Für den Einsatz des Search Servers zusammen mit der Template Engine (d.h. im Rahmen des inkrementellen Exports) werden umschaltbare Collections benötigt. Nicht umschaltbare Collections sind ausschließlich für statisch exportierte Inhalte vorgesehen.

Rückgabewert: Das Kommando gibt bei erfolgreicher Ausführung den Namen der angelegten Collection zurück.

Zusatzinformationen

- Die Namen der voreingestellten Collections lauten cm-contents (Redaktionssystem) und live-docs (Live-System). Diese Collections legt der Search Engine Server beim Start automatisch an, wenn noch keine Collections existieren.
- Bei der Erzeugung werden die mitgelieferten Style-Dateien verwendet, die sich im Verzeichnis `config/vdk/style` befinden. Gibt es allerdings unterhalb dieses Verzeichnisses ein Verzeichnis mit dem Namen der anzulegenden Collection, so werden die darin enthaltenen Style-Dateien verwendet. Diese Dateien werden in das Verzeichnis `style` unterhalb des jeweiligen Collection-Verzeichnisses kopiert.

Beispiel

```
SES>createCollection sportsNews
sportsNews
```

3.8.5 deleteCollection

Verfügbar für: Search Engine Server

Aufgabe: Das Kommando löscht eine Collection vollständig, also einschließlich aller Daten und ihrer Konfiguration. Es wird keine Bestätigung der Aktion verlangt. Daher sollte das Kommando erst nach reiflicher Überlegung verwendet werden.

Syntax

`deleteCollection` *`collectionName`*

Funktionsparameter

- *collectionName*: Der Name der zu löschenden Collection. Bei umschaltbaren Collections bezieht sich dieser Befehl auf die offline geschaltete Collection.

Rückgabewert: keiner.

Beispiel

```
SES>deleteCollection sportsNews
```

3.8.6 listCollections

Verfügbar für: Search Engine Server

Aufgabe: Das Kommando liefert eine Liste mit den Namen aller Collections.

Syntax

`listCollections`

Funktionsparameter

Das Kommando hat keine Parameter.

Rückgabewert: Die Liste der Collections, die dem Autonomy-Suchmodul bekannt sind.

Beispiel

```
SES>listCollections
cm-contents live-docs
```

3.8.7 purgeCollection

Verfügbar für: Search Engine Server

Aufgabe: Dieses Kommando leert eine Collection, indem alle indizierten Dokumente aus ihr entfernt werden. Die Collection selbst bleibt jedoch bestehen.

Syntax

`purgeCollection` *`collectionName`*

Funktionsparameter

- *collectionName*: Der Name der zu leerenden Collection. Bei umschaltbaren Collections bezieht sich dieser Befehl auf die offline geschaltete Collection.

Rückgabewert: Das Kommando hat keinen Rückgabewert. Existiert die angegebene Collection nicht, so wird die Fehlermeldung "The collection '*collectionName*' does not exist." ausgegeben. Schlägt das Kommando fehl, so wird die Meldung "Purging collection '*collectionName*' failed." ausgegeben.

Beispiel

```
SES>purgeCollection sportsNews
```

3.8.8 repairCollection

Verfügbar für: Search Engine Server

Aufgabe: Dieses Kommando versucht, eine Collection zu reparieren, die beispielsweise durch eine versehentliche Löschaktion auf Dateiebene inkonsistent geworden ist.

Syntax

`repairCollection` *`collectionName`*

Funktionsparameter

- *`collectionName`*: Der Name der zu reparierenden Collection. Bei umschaltbaren Collections bezieht sich dieser Befehl auf die offline geschaltete Collection.

Rückgabewert: Das Kommando gibt Informationen über die durchgeführten Aktionen aus.

Beispiel (für den Fall, dass die Collection nicht repariert werden muss)

```
SES>repairCollection sportsNews
mkvdk - Verity, Inc. Version 2.7.0 (_ilnx21, Feb 15 2001)
mkvdk done
```

3.8.9 Andere Befehle

Neben den Befehlen zur Administration von Collections stehen in der Tcl-Schnittstelle des Search Engine Servers die folgenden Befehle zur Verfügung:

Befehle, die nur im SES verfügbar sind

app flushQueue
app holdQueue
app resumeQueue

Befehle, die auch in anderen CMS-Fiona-Anwendungen verfügbar sind

app get
decodeData
decodeFile
decodeToString
encodeData
encodeFile
stream_uploadFile
stream_uploadBase64
stream_downloadFile
stream_downloadBase64

app flushQueue

Verfügbar für: Search Engine Server

Aufgabe: Dieser Befehl bewirkt, dass der Search Engine Server damit beginnt, Indizierungsanfragen zu verarbeiten, unabhängig von den unten genannten Systemkonfigurationseinstellungen. Der Befehl kann nur ausgeführt werden, wenn der Indizierungsstatus des SES `indexingNormal` ist.

Syntax

```
app flushQueue
```

Zusatzinformationen

Der Status des SES bezüglich der Indizierung kann der Datei `sesCommandState.state` im Verzeichnis *`cmsInstanceDir`*`/data/ses/otherData/002/008` entnommen werden. Hierbei bedeuten:

- `indexingNormal`: Indizierungsanfragen werden entgegen genommen und abgearbeitet, entsprechend der Vorgaben durch die Systemkonfigurationseinträge `tuning.indexing.interval` und `tuning.indexing.maxBulkSize`. Dieser Status wird durch den Befehl `app resumeQueue` ausgelöst.
- `indexingDelayed`: Indizierungsanfragen werden nicht abgearbeitet. Dieser Status wird durch den Befehl `app holdQueue` ausgelöst.

Funktionsparameter: keine.

Rückgabewert bei Erfolg: Statusinformationen.

app holdQueue

Verfügbar für: Search Engine Server

Aufgabe: Dieser Befehl bewirkt, dass der Search Engine Server keine neuen Indizierungsanfragen entgegen nimmt. Die Template Engine wendet diesen Befehl an, bevor sie Collections synchronisiert.

Syntax

```
app holdQueue
```

Zusatzinformationen

Der Status des SES bezüglich der Indizierung kann der Datei `sesCommandState.state` im Verzeichnis *`cmsInstanceDir`*`/data/ses/otherData/002/008` entnommen werden. Hierbei bedeuten:

- `indexingNormal`: Indizierungsanfragen werden entgegen genommen und abgearbeitet, entsprechend der Vorgaben durch die Systemkonfigurationseinträge `tuning.indexing.interval` und `tuning.indexing.maxBulkSize`. Dieser Status wird durch den Befehl `app resumeQueue` ausgelöst.
- `indexingDelayed`: Indizierungsanfragen werden nicht abgearbeitet. Dieser Status wird durch den Befehl `app holdQueue` ausgelöst.

Funktionsparameter: keine.

Rückgabewert bei Erfolg: Statusinformationen.

app resumeQueue

Verfügbar für: Search Engine Server

Aufgabe: Dieser Befehl bewirkt, dass der Search Engine Server Indizierungsanfragen wieder annimmt und verarbeitet. Die Template Engine wendet diesen Befehl an, nachdem sie die Synchronisierung von Collections abgeschlossen hat.

Syntax

```
app resumeQueue
```

Zusatzinformationen

Der Status des SES bezüglich der Indizierung kann der Datei `sesCommandState.state` im Verzeichnis *`cmsInstanceDir`*`/data/ses/otherData/002/008` entnommen werden. Hierbei bedeuten:

- `indexingNormal`: Indizierungsanfragen werden entgegen genommen und abgearbeitet, entsprechend der Vorgaben durch die Systemkonfigurationseinträge `tuning.indexing.interval` und `tuning.indexing.maxBulkSize`. Dieser Status wird durch den Befehl `app resumeQueue` ausgelöst.
- `indexingDelayed`: Indizierungsanfragen werden nicht abgearbeitet. Dieser Status wird durch den Befehl `app holdQueue` ausgelöst.

Funktionsparameter: keine.

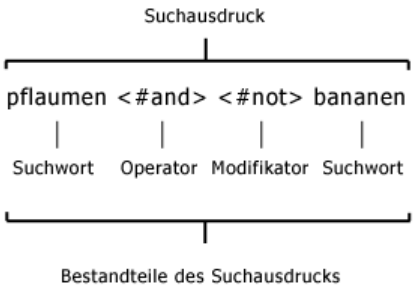
Rückgabewert bei Erfolg: Statusinformationen.

4

4 Die Syntax der Suchanfragen

4.1 Suchausdrücke

Mit einem Suchausdruck (der im Folgenden auch "Suchanfrage" genannt wird) werden die Kriterien festgelegt, nach denen der Infopark Search Server eine Suche durchführt. Die Bestandteile eines Suchausdrucks sind Suchwörter, Operatoren und Modifikatoren:



Operatoren und Modifikatoren sind Schlüsselwörter, die man in Suchausdrücken verwenden kann, um die Beziehung zwischen Suchwörtern festzulegen. Nur solche Dokumente, in denen die Suchwörter die angegebene Beziehung zueinander haben, können als Treffer im Suchergebnis erscheinen. So wird mit dem Suchausdruck

```
pflaumen <#AND> <#NOT> bananen
```

nach Dokumenten gesucht, die das Suchwort `pflaumen`, nicht jedoch das Suchwort `bananen` enthalten. Schlüsselwörter sollten grundsätzlich in spitze Klammern eingeschlossen werden. Wenn Sie die Infopark Search Engine mit deutscher Ländereinstellung betreiben, muss jedem Schlüsselwort ein Doppelkreuz vorangestellt werden. Bei der englischen Ländereinstellung ist dies nicht erforderlich.

Bei Operator- und Modifikatornamen unterscheidet die Search Engine nicht zwischen Groß- und Kleinschreibung. Suchwörter werden im Folgenden auch "Suchbegriffe" genannt.

4.1.1 Parser

Die Search Engine ist mit drei sogenannten Parsern ausgestattet, die die Suchanfragen der Benutzer analysieren und entsprechende Suchaktionen ausführen. Die Parser unterscheiden sich in der Art und Weise, wie sie Suchausdrücke analysieren und interpretieren. Jeder der drei Parser ist auf konkrete Anwendungssituationen zugeschnitten. Man kann den zu verwendenden Parser in XML-Suchanfrage-Requests an den Search Engine Server angeben und folglich je nach Benutzer-Zielgruppe in seinen Suchformularen eine mehr oder weniger umfangreiche Suchsyntax bereitstellen.

Ein Parser sollte nicht mit der von ihm unterstützten Suchsyntax gleichgesetzt werden. So kann man beispielsweise im einfachen Parser sowohl Anfragen in einfacher Syntax als auch in expliziter Syntax stellen. Ferner sind in expliziter Syntax Anfragen in mehreren Notationen erlaubt.

4.1.2 Einfacher Parser

Der einfache Parser ist universell, weil er sowohl Anfragen in einfacher Syntax als auch in expliziter Syntax zulässt. Der Parser wird bevorzugt in Umgebungen eingesetzt, in denen Benutzer mit möglichst geringem Aufwand möglichst gute Suchergebnisse erhalten sollen.

Der einfache Parser wandelt einfache Anfragen in explizite Anfragen um, wobei er sie um sinnvolle Operatoren ergänzt. Jedes Suchwort wird mit dem Modifikator `MANY` und dem Operator `STEM` versehen.

- `MANY` sorgt dafür, dass ein Dokument umso höher gewichtet wird, je größer die Dichte ist, mit der das Suchwort im Dokument vorkommt. Die Dichte ist ein relatives Maß, das das Verhältnis der Vorkommenshäufigkeit der Suchbegriffe zur Textmenge des Dokuments angibt.
- `STEM` bewirkt, dass die Suche auch Wörter umfasst, die abgeleitete Varianten des Suchwortes sind.

Gibt ein Benutzer durch Kommas getrennte Suchwörter ein, so werden die Wörter mit dem `ACCRUE`-Operator kombiniert. Dieser Operator gewichtet Dokumente umso höher, je größer die absolute Vorkommenshäufigkeit der Suchwörter in den Dokumenten ist. Eine Anfrage wie

```
apfel, banane, orange
```

wird vom einfachen Parser demnach folgendermaßen umgewandelt:

```
<#accrue>(<#many><#stem>apfel, <#many><#stem>banane, <#many><#stem>orange)
```

4.1.3 Expliziter Parser

Der explizite Parser gestattet nur Suchanfragen in expliziter Syntax. Er ist für Umgebungen konzipiert, in denen Suchanfragen programmgesteuert erzeugt werden. Die Benutzer wählen in solchen Umgebungen die auf die eingegebenen Suchwörter anzuwendenden Operationen beispielsweise mit Auswahlkästchen aus, anstatt eine Suchanfrage mit Operatoren zu versehen.

Suchanfragen in expliziter Syntax werden in Präfix- oder Infix-Notation gestellt:

- Die Präfix-Notation macht den Vorrang von Operatoren durch Klammerung explizit. So ist der Ausdruck `<#OR> (a, <#AND> (b, c))` eine Suchanfrage nach Dokumenten, die sowohl die Suchwörter `b` und `c` oder das Suchwort `a` enthalten.
- Bei der Infix-Notation ist der Vorrang von Operatoren implizit, d. h. durch die Operatoren selbst vorgegeben, es sei denn, der Vorrang wird durch Klammerung geändert. Dies betrifft insbesondere die Operatoren `AND` und `OR` , von denen `OR` den geringeren Vorrang hat. Dokumente werden also erst auf Suchbegriffe untersucht, die mit `AND` verknüpft sind, bevor sie auf `OR` -verknüpfte Suchbegriffe getestet werden. Beispiel: `a <#AND> b <#OR> c` sucht nach Dokumenten, die sowohl `a` und `b` oder `c` enthalten.

Hier ein Beispiel für Präfix-Notation:

```
<#paragraph>("fahrzeug", <#sentence>("sicherheit", <#phrase>("kein", "kompromiss")))
```

In Infix-Notation wird diese Anfrage folgendermaßen gestellt:

```
"fahrzeug" <#paragraph> "sicherheit" <#sentence> "kein" <#phrase> "kompromiss"
```

Literaler Text

Wenn Sie einzelne Wörter in doppelte Anführungszeichen einschließen, interpretiert der explizite Parser die Wörter, literal, d. h. wörtlich. Dadurch, dass Sie beispielsweise das Wort "film" ausdrücklich in doppelte Anführungszeichen einschließen, werden Wörter wie "filmt", "gefilmt" oder "filmen" bei der Suche nicht berücksichtigt:

```
"film"
```

Die Anführungszeichen sind ein syntaktisches Element der expliziten Syntax, die im einfachen und im expliziten Parser verwendet werden kann. Das folgende Beispiel ermittelt Dokumente, die die Wortfolge "Pharmazeutische Betriebe" und das Wort "Aktien" literal enthalten:

```
<#AND> ("Pharmazeutische Betriebe ", "Aktien ")
```

Im folgenden Beispiel wird nach Dokumenten gesucht, die die Wortfolge "schwarz und weiß" enthält:

```
<#PHRASE> (schwarz "und" weiß)
```

Das Wort "und" muss von doppelten Anführungszeichen umgeben sein, weil es andernfalls in einer Umgebung mit deutscher Ländereinstellung als Operator interpretiert würde.

Wenn Sie den Namen eines Themas (`topic`) in doppelte Anführungszeichen einschließen, so interpretiert ihn die Search Engine wörtlich und nicht als Themennamen. Auf diese Weise können Sie nach Wörtern suchen, die gleichzeitig Themennamen sind.

4.1.4 Freitext-Parser

Der Freitext-Parser erlaubt es, Suchanfragen zu stellen, die Sätze oder Teilsätze sind. Er behandelt sämtlichen Text als Folge von Suchwörtern. Operatoren werden folglich nicht erkannt.

Der Freitext-Parser generiert aus einer Suchanfrage eine Anfrage in expliziter Syntax, indem er unwichtige Wörter wie Artikel, Präpositionen und Konjunktionen aus der Anfrage entfernt und Suchwörter zu Wortfolgen verbindet. Der Parser ist dafür vorgesehen, dass Benutzer Ihre Anfragen in Form kurzer Fragen stellen können.

Mit dem `FREETEXT`-Operator kann die Freitext-Funktionalität auch im einfachen oder expliziten Parser verwendet werden.

4.2 Nicht englischsprachige Umgebungen

4.2.1 Die englische Anfragesprache verwenden

Die Schlüsselwörter (Operator- und Modifikatornamen) in Suchanfragen sind sprachspezifisch. Wenn beispielsweise die Search Cartridge mit der optionalen deutschen Ländereinstellung `germanx` (dem sogenannten *engl.* "locale") betrieben wird, so werden die deutschen Entsprechungen einiger englischer Operator- und Modifikatornamen auch dann als Operatoren bzw. Modifikatoren erkannt, wenn sie in Suchanfragen als Suchbegriffe vorkommen. Dies betrifft die Wörter "und", "oder", "alle", "beliebig" und "nicht". Wenn diese Wörter als Suchbegriffe verstanden werden sollen, so müssen sie mit Anführungszeichen umschlossen werden.

Es ist möglich und zu empfehlen, die englischen Operator- und Modifikatornamen in Suchanfragen zu verwenden. Dies geschieht, indem Sie jedem Operator- oder Modifikatornamen ein Doppelkreuz (#) voranstellen. Um beispielsweise in einer nicht englischsprachigen Umgebung nach der Wortfolge „Infopark AG" zu suchen, geben Sie den folgenden Ausdruck ein:

```
<#PHRASE> Infopark AG
```

In expliziter Syntax sieht der Ausdruck folgendermaßen aus:

```
<#PHRASE> (Infopark, AG)
```

4.2.2 Segmentierung von Wörtern

Bei Umgebungen, in denen Sprachen wie das Deutsche oder Englische verwendet werden, kann die Search Cartridge Leerzeichen als Separatoren zwischen Wörtern interpretieren. Spezielle sprachspezifische Segmentierer (Wort-Trenner) sind für solche Sprachen nicht erforderlich, weil die Search Cartridge Wörter, Wortfolgen und Sätze korrekt identifizieren kann. Für andere Sprachen wie Japanisch und Chinesisch ist jedoch ein Segmentierer erforderlich, um festzustellen, wo sich die Wortgrenzen befinden. Sollten Sie einen speziellen Segmentierer einsetzen, so treffen Teile der folgenden Beschreibungen der Anfragesprache in diesem Kapitel möglicherweise nicht zu.

5

5 Operatoren und Modifikatoren

5.1 Operatortypen

Ein Operator drückt eine Operation aus, die auf einen Bestandteil des Suchausdrucks angewendet wird. Diese Operation definiert Einschränkungen, denen ein Dokument unterliegen muss, um in die Trefferliste aufgenommen zu werden. Es gibt viele Operatoren, und sie lassen sich folgendermaßen nach Typ klassifizieren:

Standardoperatoren

- Konzeptoperatoren
- Evidenzoperatoren
- Abstandsoperatoren

Spezialoperatoren

- Operatoren zur Analyse geschriebener Sprache
- Gewichtungsoperatoren
- Feld- und Vergleichsoperatoren

5.1.1 Konzeptoperatoren

Konzeptoperatoren kombinieren die Bedeutung der Suchelemente, um Konzepte in Dokumenten ausfindig zu machen. Dokumente, die mit Konzeptoperatoren ermittelt werden, sind nach Relevanz bewertet. In der folgenden Tabelle werden die verfügbaren Konzeptoperatoren beschrieben:

Operatorname	Beschreibung
`ACCRUE`	Wählt Dokumente aus, die wenigstens einen der angegebenen Suchbegriffe enthalten. Je mehr Suchbegriffe vorhanden sind, desto höher wird ein Dokument bewertet.
`ALL`	Wählt Dokumente aus, die alle angegebenen Suchbegriffe enthalten. Jedes ermittelte Dokument wird mit 100 bewertet. `ALL` und `AND` sind insofern ähnlich, als sie das gleiche Suchergebnis haben. Anfragen mit `ALL` sind jedoch nicht gewichtet (alle Suchergebnisse werden mit 100 bewertet).
`AND`	Wählt Dokumente aus, die alle angegebenen Suchbegriffe enthalten. Für jedes Dokument wird eine Bewertung berechnet. `AND` und `ALL` sind insofern ähnlich, als sie das gleiche Suchergebnis haben. Anfragen mit `AND` sind jedoch gewichtet (die ermittelten Dokumente werden mit Werten zwischen 0 und 100 bewertet).

ANY	Wählt Dokumente aus, die wenigstens einen der angegebenen Suchbegriffe enthalten. Jedes Dokument wird mit 100 bewertet. ANY und OR sind insofern ähnlich, als sie das gleiche Suchergebnis haben. Anfragen mit ANY sind jedoch nicht gewichtet (alle Suchergebnisse werden mit 100 bewertet).
OR	Wählt Dokumente aus, die wenigstens einen der angegebenen Suchbegriffe enthalten. Für jedes Dokument wird eine Bewertung berechnet. OR und ANY sind insofern ähnlich, als sie das gleiche Suchergebnis haben. Anfragen mit OR sind jedoch gewichtet (die ermittelten Dokumente werden mit Werten zwischen 0 und 100 bewertet).
TOPIC	Wählt Dokumente aus, die wenigstens einen der angegebenen Suchbegriffe enthalten, die durch das spezifizierten Thema (*topic*) zusammengefasst werden. Für jedes Dokument wird eine Bewertung berechnet. Wie Themen funktionieren und konfiguriert werden, ist in der optional erhältlichen Verity-Dokumentation beschrieben.

5.1.2 Evidenzoperatoren

Evidenzoperatoren werden eingesetzt, um eine einfache oder eine intelligente Wortsuche durchzuführen. Eine einfache Wortsuche findet Dokumente, die nur das angegebene Wort oder die angegebenen Wörter enthalten. Eine intelligente Wortsuche erweitert die Suchbegriffe, um eine erweiterte Wörterliste zu erzeugen, so dass die Suche Dokumente zurückgibt, die Varianten der Suchbegriffe enthalten. So wählt der Operator THESAURUS sowohl Dokumente aus, die das angegebene Wort enthalten, als auch solche, die Synonyme des Wortes enthalten.

Dokumente, die mit Evidenzoperatoren ermittelt werden, sind nicht nach Relevanz bewertet, es sei denn, Sie verwenden den Modifikator MANY. In der folgenden Tabelle wird jeder Evidenzoperator beschrieben.

Operatorname	Beschreibung
WORD	Führt eine einfache Wortsuche durch, bei der Dokumente ausgewählt werden, in der das angegebene Wort einmal oder öfter vorkommt.
STEM	Erweitert die Suche, so dass sie nicht nur das angegebene Wort, sondern auch Varianten davon berücksichtigt.
THESAURUS	Erweitert die Suche so, dass sie das zu suchende Wort und dessen Synonyme berücksichtigt.
WILDCARD	Wendet Platzhalter in Suchzeichenketten an. Manche Zeichen werden automatisch als Platzhalter interpretiert.
SOUNDEX	Erweitert die Suche so, dass sie in Bezug auf die angegebenen Wörter auch "ähnlich klingende" Wörter umfasst. Collections haben voreingestellt keine Indizes ähnlich klingender Wörter. Um dieses Feature zu nutzen, müssen solche Indizes erstellt werden.
TYPO/*N*	Erweitert die Suche so, dass sie das angegebene Wort und diesem Wort ähnliche Wörter enthält. Dieser Operator führt einen Mustervergleich durch, um ähnliche Wörter zu finden.

5.1.3 Abstandsoperatoren

Abstandsoperatoren spezifizieren die relative Position spezifischer Wörter in dem Dokument, d. h. angegebene Wörter müssen in der gleichen Wortfolge, dem gleichen Satz oder Absatz vorkommen, damit das Dokument ein Treffer ist. Bei den Operatoren `NEAR` und `NEAR/`*`N`* sind die ermittelten Dokumente nach Relevanz bewertet, und zwar auf der Grundlage des Abstands der angegebenen Wörter zueinander.

Wenn Abstandsoperatoren verschachtelt werden, sollten diejenigen mit der größten Reichweite zuerst angegeben werden. Wortfolgen (`PHRASE`) und einzelne Wörter können in `SENTENCE`- oder `PARAGRAPH`-Operatoren auftreten, und `SENTENCE`- in `PARAGRAPH`-Operatoren. In der folgenden Tabelle werden die einzelnen Abstandsoperatoren beschrieben.

Operatorname	Beschreibung
`IN`	Wählt Dokumente aus, die spezifizierte Werte in einer Zone oder mehrenen Zonen enthalten. Eine Dokumentzone entspricht einem Feld in der Dateiversion, wie beispielsweise dem Änderungsdatum *lastChanged* oder dem Hauptinhalt.
`PHRASE`	Wählt Dokumente aus, die eine von Ihnen spezifizierte Wortfolge enthalten. Eine Wortfolge ist eine Gruppe aus zwei oder mehr Wörtern, die in festgelegter Reihenfolge auftreten.
`SENTENCE`	Wählt Dokumente aus, die alle angegebenen Wörter innerhalb des gleichen Satzes enthält.
`PARAGRAPH`	Wählt Dokumente aus, die alle angegebenen Wörter innerhalb des gleichen Absatzes enthält.
`NEAR`	Wählt Dokumente aus, die die angegebenen Suchbegriffe enthalten, wobei die Bewertung des Dokuments um so höher ist, je näher die Wörter beieinander liegen.
`NEAR/N`	Wählt Dokumente aus, die zwei oder mehr Suchbegriffe innerhalb von *N* Wörtern enthalten, wobei *`N`* eine ganze Zahl bis 1024 ist. Je geringer der Abstand zwischen den Suchbegriffen ist, desto höher wird das Dokument bewertet.

5.1.4 Operatoren zur Analyse geschriebener Sprache

Die Operatoren `FREETEXT` und `LIKE` dienen dazu, geschriebene Sprache zu analysieren. Die Suchmaschine übersetzt den Text der Suchanfrage in die Suchsyntax, führt die Anfrage durch und gewichtet die gefundenen Dokumente entsprechend ihrer Relevanz. Die beiden Operatoren sind hauptsächlich für den Einsatz in der Applikationsentwicklung vorgesehen.

Operatorname	Beschreibung
`FREETEXT`	Der Operator wertet den Text der Suchanfrage so aus, als wäre die Suchanfrage im Freitext-Parser gestellt worden (siehe Suchausdrücke). Insbesondere werden Artikel, Präpositionen und Konjunktionen entfernt sowie die Gruppierung und Reihenfolge von Wörtern berücksichtigt. Der Freitext-Parser ist dafür optimiert, kurze Fragen in geschriebener Sprache zu analysieren und in die Verity-Anfragesprache zu übersetzen.

LIKE	Dieser Operator führt eine Suche auf der Basis positiver oder negativer Beispieltexte durch und bewertet die gefundenen Dokumente entsprechend des Grades der Übereinstimmung (QBE = *engl.* "Query by example").

5.1.5 Gewichtungsoperatoren

Die Bewertungsoperatoren beeinflussen, wie das Such-Modul die ermittelten Dokumente bewertet, d. h. deren Punktzahl berechnet. Sie können untereinander und mit anderen Operatoren der Anfragesprache kombiniert werden.

Wenn ein Bewertungsoperator verwendet wird, so berechnet die Suchmaschine zunächst eine separate Punktzahl für jeden Bestandteil des Suchausdrucks, der in einem Dokument gefunden wird, um anschließend aus den einzelnen Punktzahlen mit einer mathematischen Operation die Ergebniszahl zu berechnen.

Der Operator YESNO hat ein breites Anwendungsspektrum, während die Operatoren PRODUCT, SUM und COMPLEMENT hauptsächlich für Applikationsentwickler gedacht sind, die Anfragen programmgesteuert generieren möchten.

Operatorname	Beschreibung
COMPLEMENT	Der Operator subtrahiert die Gesamtbewertung eines jeden gefundenen Dokuments von 100.
PRODUCT	Mit diesem Operator lassen sich Dokumente feiner bewerten. Die Einzelergebnisse der Suche werden miteinander multipliziert, und das Ergebnis dieser Operation wird durch 100 dividiert.
SUM	Der Operator addiert die Einzelbewertungen bis zum Höchstwert 100.
YESNO	Mit dem Operator kann man eine Suche auf jene Dokumente beschränken, auf die eines der Suchkriterien zutrifft, ohne dass die Bewertung dieses Kriteriums die endgültige Bewertung beeinflusst. Der Operator setzt ein Einzelergebnis auf 100 Punkte, wenn es ungleich 0 ist, andernfalls bleibt es 0.

5.1.6 Feld- und Vergleichsoperatoren

Feldoperatoren durchsuchen Dokumentfelder, die in einer Collection definiert wurden. Diese Operatoren führen Filterfunktionen aus, indem sie Dokumente auswählen, deren Felder die angegebenen Werte haben. Eine Suche in Dokumentfeldern ist langsamer als in Zonen. Es ist normalerweise nicht erforderlich, in Feldern zu suchen, weil die meisten Felder ebenfalls als Zonen vorhanden sind.

Dokumente, die mit Feldoperatoren ermittelt werden, sind weder nach Relevanz bewertet, noch kann man den Modifikator MANY zusammen mit Feldoperatoren verwenden.

Mit Hilfe der folgenden Vergleichsoperatoren können die Werte von Dokumentfeldern mit Suchbegriffen verglichen werden: = (gleich), != (ungleich), > (größer als), >= (größer oder gleich), < (kleiner als), <= (kleiner oder gleich).

Für Textvergleiche stehen die folgenden Feldoperatoren zur Verfügung:

Operatorname	Beschreibung
CONTAINS	Wählt Dokumente aus, wenn das angegebene Wort oder die Wortfolge in den Werten eines bestimmten Dokumentfeldes enthalten ist. Dokumente werden nur ausgewählt, wenn die spezifizierten Suchelemente in der gleichen Abfolge im Wert des Feldes auftreten.
MATCHES	Wählt Dokumente aus, indem der Suchbegriff mit den Werten in einem bestimmten Dokumentfeld verglichen wird. Dokumente werden nur ausgewählt, wenn die spezifizierten Suchelemente exakt dem Wert des Feldes entsprechen. Bei partieller Übereinstimmung wird ein Dokument nicht ausgewählt.
STARTS	Wählt Dokumente aus, bei denen der Suchbegriff vom Anfang her mit den Zeichen der Werte eines bestimmten Dokumentfeldes übereinstimmt.
ENDS	Wählt Dokumente aus, bei denen der Suchbegriff vom Ende her mit den Zeichen der Werte eines bestimmten Dokumentfeldes übereinstimmt.
SUBSTRING	Wählt Dokumente aus, bei denen der Suchbegriff in den Zeichenketten der Werte eines bestimmten Dokumentfeldes enthalten ist.

5.2 Übersicht der Standardoperatoren

Dieser Abschnitt beschreibt detailliert die Standardoperatoren der Search Cartridge. Wo dies angebracht ist, wurde jede Beschreibung mit einem Beispiel für einfache und explizite Syntax versehen. Die Operatoren sind alphabetisch aufgeführt.

Bitte beachten Sie, dass in Umgebungen mit einer anderen Ländereinstellung als `english` die Operatoren immer in spitze Klammern eingeschlossen werden müssen und ihnen ein Doppelkreuz vorangestellt werden muss (Beispiel: `<#AND>`).

Wenn Sie ferner als Suchwörter "und", "oder", "beliebig", "alle" oder "nicht" angeben, so müssen diese Wörter von Anführungszeichen umschlossen werden, wenn Sie die deutsche Ländereinstellung verwenden. Andernfalls werden diese Wörter als die deutschen Entsprechungen der Operatoren `AND`, `OR`, `ANY` und `ALL` bzw. als Äquivalent des Modifikators `NOT` interpretiert.

5.2.1 ACCRUE

Wählt Dokumente aus, die wenigstens einen der angegebenen Suchbegriffe enthalten. Gültige Suchanfragen enthalten zwei oder mehr Wörter oder Wortfolgen. Die ermittelten Dokumente werden nach Relevanz bewertet.

Der `ACCRUE`-Operator bewertet ermittelte Dokumente nach dem Vorhandensein aller Suchbegriffe. Je mehr Suchbegriffe in einem Dokument vorhanden sind, desto höher wird es bewertet. Die folgenden Beispiele veranschaulichen die Such-Syntax.

Um Dokumente auszuwählen, die die Wörter "Computer" und "Laptops" sowie Wörter mit dem gleichen Wortstamm enthalten, können Sie jede der drei folgenden Suchanfragen eingeben:

```
Computer <#ACCRUE> Laptops
Computer, Laptops
<#ACCRUE> (Computer, Laptops)
```

5.2.2 ALL

Wählt Dokumente aus, die alle Ihre Suchbegriffe enthalten. Dokumente, die mit dem Operator `ALL` ermittelt werden, sind nicht nach Relevanz bewertet. Mit diesem Operator kann keine Gewichtung vorgenommen werden.

Um Dokumente auszuwählen, die die Wortfolge "Pharmazeutische Betriebe" und "Aktien" sowie davon abgeleitete Varianten enthalten, können Sie folgendes eingeben:

```
Pharmazeutische Betriebe <#ALL> Aktien
<#ALL>(Pharmazeutische Betriebe, Aktien)
```

Es werden nur solche Dokumente ermittelt, die beide Suchbestandteile oder davon abgeleitete Varianten enthalten (beispielsweise "Pharmazeutischer Betrieb", "Aktie" usw.). Jedes ermittelte Dokument wird mit 100 bewertet.

5.2.3 AND

Wählt Dokumente aus, die alle Ihre Suchbegriffe enthalten. Dokumente, die mit dem `AND`-Operator ermittelt werden, sind nach Relevanz gewichtet.

Um Dokumente auszuwählen, die die Wortfolge "Pharmazeutische Betriebe" und "Aktien" sowie davon abgeleitete Varianten enthalten, können Sie folgendes eingeben:

```
Pharmazeutische Betriebe <#AND> Aktien
<#AND>(Pharmazeutische Betriebe, Aktien)
```

Es werden nur solche Dokumente ermittelt, die beide Suchbestandteile oder davon abgeleitete Varianten enthalten (beispielsweise "Pharmazeutischer Betrieb", "Aktie" usw.). Jedem Dokument wird eine berechnete Bewertung zugewiesen.

5.2.4 ANY

Wählt Dokumente aus, in denen wenigstens einer Ihrer Suchbestandteile vorkommt. Die ermittelten Dokumente sind nicht nach Relevanz bewertet. Mit diesem Operator kann keine Gewichtung vorgenommen werden.

Um Dokumente auszuwählen, die eines der Wörter "Inhalte" und "Redaktion" oder die Wortfolge "Content Management" sowie davon abgeleitete Varianten enthalten, können Sie folgendes eingeben:

```
Inhalte <#ANY> Redaktion <#ANY> Content Management
<#ANY>(Inhalte, Redaktion, Content Management)
```

Es werden nur solche Dokumente ermittelt, die wenigstens einen der Suchbestandteile oder eine abgeleitete Variante wenigstens eines Suchbestandteils enthalten. Jedes Dokument wird mit 100 bewertet.

5.2.5 IN

Wählt Dokumente aus, bei denen eine oder mehrere Dokumentzonen die angegebenen Suchbegriffe enthalten. Eine Dokumentzone entspricht einem Feld in der Dateiversion, wie beispielsweise dem Titel (`title`) oder dem Hauptinhalt (`blob`). Der angegebene Zonenname muss genau dem tatsächlichen Namen entsprechen.

Die folgende Suchanfrage durchsucht Dokumentzonen mit dem Namen "summary“ nach dem Wort "Sicherheit".

```
"sicherheit" <#IN> summary
```

Um nach mehreren Wörtern, Wortfolgen oder Themen (*engl.* "topics") zu suchen, schließen Sie sie bitte in Klammern ein. Der folgende Anfrageausdruck durchsucht die Dokumentzonen mit dem Namen "summary" nach dem Wort "Sicherheit" und nach abgeleiteten Varianten des Wortes "Warnung".

```
("Sicherheit", warnung) <#IN> summary
```

Mehrere Zonen können Sie durchsuchen, indem Sie sie mit einem Komma trennen und in Klammern einschließen. Der folgende Anfrageausdruck durchsucht sowohl die Dokumentzonen mit dem Namen "myReadPermission" als auch mit dem Namen "myWritePermission" nach den Wörtern "public" und "partners".

```
("public", "partners") <#IN> (myReadPermission, myWritePermission)
```

5.2.6 NEAR

Wählt Dokumente aus, die die angegebenen Suchbegriffe nahe beieinander enthalten. Die Gewichtung der Dokumente wird auf der Basis der Anzahl Wörter berechnet, die sich zwischen den Suchbegriffen befinden. Wenn beispielsweise der Suchausdruck zwei Wörter umfasst und diese Wörter sich in einem Dokument nebeneinander befinden (so dass die Region zwei Wörter lang ist), wird das Dokument mit 100 Punkten bewertet. Das heißt, das Dokument, in dem alle Suchbegriffe innerhalb der kleinstmöglichen Region vorkommen, erhält die höchste Bewertung. Je größer der Abstand zwischen den Wörtern ist, desto niedriger fällt die Bewertung aus. Ein Dokument wird nur dann mit 0 Punkten bewertet, wenn es nicht alle Suchbegriffe enthält.

Der `NEAR`-Operator ist anderen Abstandsoperatoren insofern ähnlich, als die eingegebenen Suchbegriffe nahe beieinander liegen müssen. Anders als andere Abstandsoperatoren berechnet der `NEAR`-Operator jedoch relative Abstände und gewichtet Dokumente aufgrund dieser Berechnungen.

Um nach Relevanz gewichtete Dokumente zu ermitteln, die die Wörter "Krieg" und "Frieden" oder davon abgeleitete Wörter mit möglichst geringem Abstand voneinander enthalten, können Sie folgendes eingeben:

```
Krieg <#NEAR> Frieden<#NEAR>(Krieg, Frieden)
```

5.2.7 NEAR/N

Wählt Dokumente aus, die zwei oder mehr Wörter innerhalb von *N* Wörtern enthalten, wobei *N* eine ganze Zahl ist. Die Gewichtung des Dokuments wird auf der Basis des relativen Abstands der angegebenen Wörter zueinander berechnet, wenn sich *N* oder weniger Wörter zwischen den gesuchten Wörtern befinden.

Wenn beispielsweise der Suchausdruck `NEAR/5` verwendet wird, um zwei Wörter mit einem Abstand von maximal fünf Wörtern zu finden, so wird ein Dokument, bei dem die Wörter einen Abstand von drei Wörtern haben, höher bewertet als ein Dokument, bei dem der Abstand fünf Wörter beträgt.

Die Variable *N* kann eine ganze Zahl zwischen 1 und 1.024 sein, wobei `NEAR/1` nach zwei Wörtern sucht, die sich unmittelbar nebeneinander befinden. Wenn *N* größer oder gleich 1.000 ist, muss sein Wert ohne Dezimaltrenner angegeben werden (wie bei `NEAR/1000`). Man kann `NEAR/`*N* mehrfach verwenden und entsprechend viele Suchbegriffe angeben, so lange der Wert von *N* immer gleich ist.

Um beispielsweise nach Relevanz gewichtete Dokumente zu ermitteln, die die Wörter "verwandeln", "Fahrad", "Zug" und "Bus" oder davon abgeleitete Wörter mit einem Abstand von jeweils bis zu zehn Wörtern enthalten, kann man folgendes eingeben:

```
verwandeln <#NEAR/10> Fahrrad <#NEAR/10> Zug <#NEAR/10> Bus
```

Sie können den Operator `NEAR/`*N* zusammen mit dem Modifikator `ORDER` verwenden, um eine Suche nach Abstand und Abfolge durchzuführen.

5.2.8 OR

Wählt Dokumente aus, in denen wenigstens einer der Suchbestandteile auftritt. Dokumente, die mit dem `OR`-Operator ermittelt werden, sind nach Relevanz gewichtet.

Um Dokumente auszuwählen, die eines der Wörter "Inhalte" und "Redaktion" oder die Wortfolge "Content Management" sowie davon abgeleitete Varianten enthalten, können Sie folgendes eingeben:

```
Inhalte <#OR> Redaktion <#OR> Content Management
<#OR>(Inhalte, Redaktion, Content Management)
```

Es werden nur solche Dokumente ermittelt, die wenigstens einen der Suchbestandteile oder eine abgeleitete Variante wenigstens eines Suchbestandteils enthalten. Jedem Dokument wird eine berechnete Bewertung zugewiesen.

5.2.9 PARAGRAPH

Wählt Doumente aus, die alle Suchbegriffe innerhalb eines Absatzes enthalten. Gültige Suchbestandteile enthalten zwei oder mehr Wörter oder Wortfolgen. Dokumente werden dann ausgewählt, wenn die Suchbestandteile im gleichen Absatz auftreten.

Um nach Relevanz gewichtete Dokumente zu ermitteln, die das Wort "Medizin" und die Wortfolge "Grippaler Infekt" oder davon abgeleitete Varianten im gleichen Absatz enthalten, können Sie folgendes eingeben:

```
medizin <#PARAGRAPH> grippaler infekt
```

```
<#PARAGRAPH>(medizin, grippaler infekt)
```

Um nach drei oder mehr Wörtern oder Wortfolgen im gleichen Absatz zu suchen, muss der PARAGRAPH-Operator bei Verwendung der Infix-Notation (erstes der beiden obigen Beispiele) jeweils zwischen den Wörtern oder Wortfolgen angegeben werden.

Sie können den PARAGRAPH-Operator zusammen mit dem Modifikator ORDER verwenden, um eine Suche nach Abstand und Abfolge durchzuführen.

5.2.10 PHRASE

Wählt Dokumente aus, die eine von Ihnen angegebene Wortfolge enthält. Eine Wortfolge ist eine Gruppe von zwei oder mehr Wörtern, die in festgelegter Reihenfolge auftreten.

Voreingestellt werden zwei oder mehr Wörter, die durch ein Leerzeichen voneinander getrennt sind, in einfacher Syntax als Wortfolge betrachtet. Ferner werden zwei oder mehr Wörter in doppelten Anführungszeichen als Wortfolge interpretiert. Um nach Relevanz gewichtete Dokumente zu ermitteln, die die Wortfolge "Kleine Stadt" enthalten, können Sie jede der folgenden Suchanfragen eingeben:

```
kleine stadt
"kleine stadt"
kleine <#PHRASE> stadt
<#PHRASE> (kleine, stadt)
```

5.2.11 SENTENCE

Wählt die Dokumente aus, die alle von Ihnen angegebenen Wörter innerhalb eines Satzes enthalten. Sie können die Suchbegriffe sequenziell oder in beliebiger Reihenfolge eingeben. Sofern die Wörter im gleichen Satz auftreten, werden die Dokumente ins Suchergebnis aufgenommen.

Um nach Relevanz gewichtete Dokumente zu ermitteln, die die Wörter "Märchenhafter" und "Geschmack"oder davon abgeleitete Varianten innerhalb des gleichen Satzes enthalten, können Sie folgendes eingeben:

```
märchenhafter <#SENTENCE> geschmack
<#SENTENCE> (märchenhafter, geschmack)
```

Sie können den SENTENCE-Operator zusammen mit dem Modifikator ORDER verwenden, um eine Suche nach Abstand und Abfolge durchzuführen.

5.2.12 SOUNDEX

Wählt Dokumente aus, die ein oder mehrere Wörter enthalten, die ähnlich wie das angegebene Wort klingen oder ähnlich geschrieben werden. Die zu suchenden Wörter müssen mit dem gleichen Buchstaben beginnen wie die eingegebenen.

Um mit dem SOUNDEX-Operator in einer Collection suchen zu können, müssen sämtliche Dokumente mit der Wort-Indizierungsoption Soundex indiziert worden sein. Diese Option kann zum Parameter WORD-IDXOPTS in der Datei style.prm hinzugefügt werden. Anschließend müssen alle Dokumente neu indiziert werden.

Um Dokumente mit Wörtern zu ermitteln, die dem Wort "Verkauf" ähnlich sind, können Sie das Folgende eingeben:

```
<#SOUNDEX> Verkauf
```

Die ermittelten Dokumente werden Wörter wie "Verlauf" und "Verlaub" enthalten. Die Dokumente sind nicht nach Relevanz gewichtet, es sei denn, der Modifikator MANY wird verwendet:

```
<#MANY><#SOUNDEX> Verkauf
```

5.2.13 STEM

Wählt Dokumente aus, die eine oder mehrere Varianten des angegebenen Suchbegriffs enthalten. Um beispielsweise die Dokumente zu ermitteln, die eine Variante des Wortes "Film" enthalten, können Sie folgendes eingeben:

```
<#STEM> film
```

Dadurch werden Dokumente ausgewählt, die Wörterwie "Filme", "filmen" oder "gefilmt" enthalten. Die Dokumente sind nicht nach Relevanz gewichtet, es sei denn, der Modifikator MANY wurde angegeben:

```
<#MANY><#STEM> film
```

5.2.14 THESAURUS

Wählt Dokumente aus, die ein oder mehrere Synonyme des angegebenen Wortes enthalten. Um beispielsweise die Dokumente zu ermitteln, die Synonyme des Wortes "Höhe" enthalten, können Sie das Folgende eingeben:

```
<#THESAURUS> höhe
```

Die ermittelten Dokumente sind nicht nach Relevanz gewichtet, es sei denn, der Modifikator MANY wurde angegeben:

```
<#MANY><#THESAURUS> Höhe
```

5.2.15 TOPIC

Themen (*engl.* "topics") sind Suchbegriffe, die von der Search Cartridge zu Suchanfragen expandiert werden. Die erzeugte Suchanfrage verknüpft diejenigen Suchbegriffe miteinander, die als zum Thema gehörend konfiguriert wurden.

Themen werden im einfachen Parser auch dann als solche erkannt, wenn Sie nicht den Operator `TOPIC` verwenden. Im expliziten Parser können Sie jedes der folgenden Formate verwenden, um in einem Ausdruck ein Thema anzugeben:

```
{themenname}
<#TOPIC>themenname
<#TOPIC>(themenname)
```

In den obigen Beispielen ist *`themenname`* der Name des Themas, das in dem Ausdruck verwendet werden soll.

5.2.16 TYPO/N

Wählt Dokumente aus, die das angegebene Wort und diesem Wort ähnliche Wörter enthalten. Der Operator `TYPO/`*`N`* führt einen Mustervergleich durch, um ähnliche Wörter zu erkennen. Man kann ihn daher gut in Umgebungen einsetzen, in denen Dokumente mit einem Scanner unter Einsatz optischer Zeichenerkennung (OCR, *engl.* "Optical Character Recognition") eingelesen wurden.

Die optionale Variable *`N`* im Operatornamen drückt die maximal zulässige Anzahl der Unterschiede zwischen dem Suchbegriff und den Wörtern aus, mit denen er verglichen wird (Fehlerabstand). Wenn *`N`* nicht angegeben wird, so wird als Fehlerabstand 2 angenommen.

Die Anzahl der Unterschiede zwischen zwei Wörtern beruht auf der Berechnung von Fehlern, wobei es als Fehler gilt, wenn ein Zeichen entfallen ist, eingefügt oder durch ein anderes ersetzt wurde (Transposition). In den folgenden Beispielen stimmt jeweils das zweite mit dem ersten Wort bei einem Fehlerabstand von 1 überein:

```
Maus, Haus (M # H)
feilen, eilen (f ist entfallen)
Tor, Thor (h wurde eingefügt)
```

Auf die folgende Anfrage passen Dokumente mit den Wörtern "schaufeln" und "schummeln", weil es drei Transpositionen im zweiten Wort gibt (a # u, u # m, f # m).

```
<#TYPO/3> schaufeln
```

Die beiden unten aufgeführten Anfragen haben das gleiche Ergebnis. Dokumente, die die Wörter "Geiger" und "Tiger" enthalten, werden durch diese Anfragen gefunden, weil in "Tiger" gegenüber "Geiger" ein Zeichen entfallen ist und eines durch ein anderes ersetzt wurde.

```
<#TYPO/2> Geiger
<#TYPO> Geiger
```

Der Operator `TYPO/`*`N`* muss die Wörterliste der Collection absuchen, um Wörter zu finden, die auf die Anfrage passen könnten. Er sollte daher nicht bei großen Collections (mehr als 100.000 Dokumente) oder in performanzkritischen Umgebungen verwendet werden, es sei denn, eine übergreifende Wörterliste wurde erzeugt. Eine solche übergreifende Wörterliste für die zu verwendende Collection kann die Performanz verbessern.

Bitte beachten Sie die die folgenden Einschränkungen: Ein Begriff, nach dem mit `TYPO/`*`N`* gesucht wird, darf höchstens 32 Zeichen lang sein. Ferner wird `TYPO/`*`N`* bei Multibyte-Zeichensätzen nicht unterstützt.

5.2.17 WILDCARD

Wählt Dokumente aus, in denen Zeichenketten auf einen Suchbegriff mit Platzhaltern (Jokerzeichen) passen. Mit dem `WILDCARD`-Operator können Sie eine Zeichenkette mit Platzhaltern als Suchbegriff angeben, um auch Dokumente mit Wörtern zu finden, die anstelle des Platzhalters andere Zeichen enthalten.

Um beispielsweise Dokumente zu ermitteln, die Wörter wie "pharmazeutisch" oder "Pharmazie" enthalten, können Sie folgendes eingeben:

```
pharmaz*
```

Die Dokumente werden nicht nach Relevanz gewichtet, es sei denn, der Operator `MANY` wird angegeben:

```
<#MANY> pharmaz*
```

Die Zeichen "*" und "?" werden automatisch als Platzhalter interpretiert. Andere Formen der Suche mit Platzhaltern können Sie mit dem `WILDCARD`-Operator und den folgenden Zeichen durchführen:

Zeichen	Aufgabe
`?`	Bedeutet "ein beliebiges alphanumerisches Zeichen" wie in der Zeichenkette `?ut`, die beispielsweise auf "hut", "gut" oder "mut" passt. Es ist nicht erforderlich, den `WILDCARD`-Operator anzugeben, wenn Sie das Fragezeichen verwenden. Das Fragezeichen wird in Mengen (`[ ]`) oder alternativen Zeichenketten (`{ }`) ignoriert.
*	Bedeutet null oder mehr beliebige alphanumerische Zeichen wie in `alt*`, das auf "alt", "alter" oder "altruistisch" passt. Es ist nicht erforderlich, den `WILDCARD`-Operator anzugeben, wenn Sie das Sternchen verwenden. Sie sollten das Sternchen nicht als erstes Zeichen einer Zeichenkette mit Platzhaltern verwenden. Das Sternchen wird in Mengen (`[ ]`) und Alternativangaben (`{ }`) ignoriert.
`[ ]`	Legt eine Zeichenmenge fest. `<WILDCARD> 'h[aiu]t'` findet "hat", "hit" und "hut". Wörter, die Mengen enthalten, müssen in einfache Anführungszeichen eingeschlossen werden, und es dürfen keine Leerzeichen in Mengen vorkommen.
`{ }`	Legt eine Menge alternativer Zeichenketten fest. Die Alternativen sind durch Kommas getrennt. `<WILDCARD> 'bau{en,er,m}'` findet "bauen", "bauer" und "baum". Wörter, die Mengen enthalten, müssen in einfache Anführungszeichen eingeschlossen werden, und es dürfen keine Leerzeichen in Mengen vorkommen.

^	Legt fest, dass die angegebenen Zeichen in einer Menge nicht vorkommen dürfen. `<WILDCARD> 'lau[^ft]en'` schließt "laufen" und "lauten" aus, findet jedoch "laugen" und "lausen". Dieses Zeichen muss das erste Zeichen nach der linken Klammer (`[`) sein, die eine Menge öffnet.
-	Legt einen Bereich von Zeichen in einer Menge fest. `<WILDCARD> 'h[a-u]t'` passt auf alle Wörter mit drei Buchstaben im Bereich von "hat" bis "hut".

Nach nicht alphanumerischen Zeichen suchen

Bitte beachten Sie, dass man nur nach nicht alphanumerischen Zeichen suchen kann, wenn die Datei `style.lex`, die bei der Erzeugung der zu durchsuchenden Collections verwendet wurde, so eingerichtet wurde, dass die zu suchenden Zeichen erkannt werden. Wenden Sie sich bitte an den Verwalter der Collections, wenn Sie mehr Informationen benötigen.

Nach Platzhaltern als Literale suchen

Wenn die Datei `style.lex` für die zu durchsuchenden Collections eingerichtet wurde, können Sie nach einem Wort mit einem Platzhalterzeichen wie "/" oder "*" suchen, indem Sie dem Platzhalterzeichen einen Rückwärts-Schrägstrich (Backslash) voranstellen. So findet die Suchmaschine beim folgenden Suchbegriff Wörter mit fünf Zeichen, die auf "abc*d" passen.

```
abc\*d
```

Wenn Sie den Backslash als Literal finden möchten, so ist es erforderlich, zwei Backslashes einzugeben.

Nach Spezialzeichen als Literale suchen

Die folgenden nicht alphanumerischen Zeichen haben spezielle Aufgaben in der Suchmaschine und können in Zeichenketten mit Platzhaltern voreingestellt nicht als Literale behandelt werden:

- Komma ,
- Linke und rechte Klammern ( )
- Doppelte Anführungszeichen "
- Backslash \
- At-Zeichen @
- Linke geschweifte Klammer {
- Linke eckige Klammer [
- Kleiner-Zeichen <
- Backquote `

Um spezielle Zeichen als Literale zu interpretieren, muss die ganze Zeichenkette mit Platzhaltern in Backquotes (`) eingeschlossen werden. Ein Backquote ist ein allein stehender Accent grave. Um beispielsweise nach "a{b" zu suchen, schließen Sie die Zeichenkette folgendermaßen in Backquotes ein:

```
```

```
<#WILDCARD> `a{b`
```

Um unter Berücksichtigung von Platzhaltern nach Zeichenketten zu suchen, die den Backquote als Literal enthalten, müssen zwei Backquotes zusammen verwendet, und die gesamte Zeichenkette muss in Backquotes eingeschlossen werden:

```
<#WILDCARD> `*``nt`
```

Man kann nur dann nach Backquotes suchen, wenn die Datei `style.lex`, die bei der Erzeugung der zu durchsuchenden Collections verwendet wurde, so eingerichtet wurde, dass Backquotes erkannt werden.

5.2.18 WORD

Wählt Dokumente aus, in denen das von Ihnen angegebene Wort einmal oder öfter vorkommt. Um beispielsweise nach Dokumenten zu suchen, die das Wort "schön" enthalten, ohne gleichzeitig die Wörter "schöner", "schönen" oder "verschönern" zu berücksichtigen, können Sie folgendes eingeben:

```
<#WORD> schön
```

Die Dokumente werden nicht nach Relevanz gewichtet, es sei denn, der Modifikator `MANY` wird verwendet:

```
<#MANY><#WORD> schön
```

5.3 Übersicht der Spezialoperatoren

Die Dokumente in diesem Abschnitt beschreiben die Spezialoperatoren der Search Cartridge. Wo dies angebracht ist, wurde jede Beschreibung mit einem Beispiel für einfache und explizite Syntax versehen. Die Operatoren sind alphabetisch aufgeführt. Die Vergleichsoperatoren werden am Ende der Übersicht beschrieben.

Bitte beachten Sie, dass in Umgebungen mit einer anderen Ländereinstellung als `english` die Operatoren immer in spitze Klammern eingeschlossen werden müssen und ihnen ein Doppelkreuz vorangestellt werden muss (Beispiel: `<#AND>`).

Wenn Sie ferner als Suchwörter `und`, `oder`, `beliebig`, `alle` oder `nicht` angeben, so müssen diese Wörter von Anführungszeichen umschlossen werden, wenn Sie die deutsche Ländereinstellung verwenden. Andernfalls werden diese Wörter als die deutschen Entsprechungen der Operatoren `AND`, `OR`, `ANY` und `ALL` bzw. als Äquivalent des Modifikators `NOT` interpretiert.

5.3.1 COMPLEMENT

Der Operator berechnet die Punktzahl der Dokumente, die auf eine Suchanfrage passen, indem die Ergebniszahl von 100 subtrahiert wird. Ein Beispiel für die Suchsyntax:

```
<#COMPLEMENT> ("computer")
```

Der Operator ist ein unärer Operator. Er wird nicht einzeln auf die Ergebnisse der Suchbestandteile, sondern auf das Gesamtergebnis angewandt. Die Suchbestandteile werden voreingestellt mit dem ACCRUE-Operator kombiniert, um das Gesamtergebnis zu berechnen, von dem dann das Komplement berechnet wird. Hier eine Beispielanfrage mit zwei Suchbestandteilen:

```
<#COMPLEMENT> ("computer","laptop")
```

Im obigen Beispiel werden die Wörter "computer" und "laptop" mit dem ACCRUE-Operator kombiniert. Der COMPLEMENT-Operator wird auf das Ergebnis angewandt.

5.3.2 CONTAINS

Wählt Dokumente aus, indem das angegebene Wort oder die Wortfolge mit den Werten verglichen wird, die in einem bestimmten Dokumentfeld gespeichert sind. Es werden nur Dokumente ausgewählt, wenn die angegebenen Suchbestandteile in der gleichen Reihenfolge im Wert des Feldes auftreten.

Wenn man den Operator CONTAINS verwendet, gibt man den Namen des zu durchsuchenden Feldes und das gesuchte Wort oder die gesuchte Wortfolge an.

Beim CONTAINS-Operator werden die in einem Dokumentfeld gespeicherten Wörter als einzelne, aufeinanderfolgende Einheiten interpretiert. Sie können eine oder mehrere dieser Einheiten als Suchkriterium angeben. Wenn man mehrere Wörter angibt, so muss jedes Wort an der richtigen Position (d. h. der Position, an der es gesucht wird) stehen, und die Wörter müssen durch ein Leerzeichen voneinander getrennt sein.

Der folgende Beispiel-Titel enthält fünf Wörter in der angegebenen Reihenfolge:

```
Fischers Fritz fischt frische Fische
```

1. Fischers
2. Fritz
3. fischt
4. frische
5. Fische

Die folgenden Beispiele zeigen, wie man CONTAINS mit aufeinanderfolgenden Wörtern so verwendet, dass die Suchbegriffe auf den oben angegebenen Titel passen, vorausgesetzt, er ist in einem Dokumentfeld gespeichert:

```
TITLE <#CONTAINS> Fischers Fritz
TITLE <#CONTAINS> Fritz fischt
```

Die nächsten Beispiele zeigen, wie man das Fragezeichen (?) als Platzhalter für ein einzelnes Zeichen und den Stern (*) als Platzhalter für mehrere Zeichen eines Wortes verwendet:

```
TITLE <#CONTAINS> Fritz fischt fri*
TITLE <#CONTAINS> Fisch??? Fritz
```

Die beiden Platzhalter erfassen nicht den Leerraum zwischen Wörtern, sondern Zeichen, die Bestandteil eines einzigen Wortes sind.

Der `CONTAINS` -Operator interpretiert nicht-alphanumerische Zeichen als Leerzeichen und behandelt die separierten Wörter als Einheiten.

Wenn Sie beispielsweise den Bindestrich (-) als gültiges Zeichen definiert haben, und Sie geben Suchbegriffe ein, die dieses Zeichen enthalten, wie in "nicht-alphanumerisch", so wird der Wert folgendermaßen als zwei Einheiten interpretiert:

```
TITLE <#CONTAINS> nicht alphanumerisch
```

5.3.3 ENDS

Wählt Dokumente aus, bei denen ein Wert im angegebenen Dokumentfeld mit der gesuchten Zeichenkette endet. Wenn es beispielsweise ein Dokumentfeld `AUTOR` gibt, so kann man folgendermaßen nach Dokumenten suchen, die von Kästner, Wagner und Büchner geschrieben wurden:

```
AUTOR <#ENDS> ner
```

5.3.4 = (gleich)

Wählt Dokumente aus, deren Dokumentfelder exakt die gleichen Werte enthalten wie die angegebene gesuchte Zeichenkette. So kann man beispielsweise nach Dokumenten suchen, die zu Objekten vom Typ `document` gehören:

```
objType = document
```

5.3.5 FREETEXT

Der Operator ist in erster Linie für den Einsatz in der Applikationsentwicklung vorgesehen. Er interpretiert Texte mit dem Freitext-Anfrage-Parser und bewertet Dokumente entsprechend des resultierenden Anfrageausdrucks. Alle ermittelten Dokumente werden nach Relevanz bewertet.

Bei der Analyse des Suchtextes werden unwichtige Wörter (sogenannte Stoppwörter) wie Artikel, Konjunktionen und Präpositionen entfernt, und Besonderheiten der natürlichen Sprache wie Nominalphrasen und die Reihenfolge der Wörter werden und in der resultierenden Anfrage berücksichtigt. Alle ermittelten Dokumente werden nach Relevanz bewertet. So könnte beispielsweise die Anfrage

```
<#FREETEXT> ("freiwillige feuerwehr in ländlichen regionen")
```

folgendermaßen interpretiert werden:

```
<#ACCRUE> (<#PHRASE> "freiwillige feuerwehr", <#PHRASE> "ländlichen regionen")
```

Der FREETEXT-Operator stellt Ihnen die Funktionalität des Freitext-Parsers zur Verfügung, gibt Ihnen jedoch die Möglichkeit, Freitext-Suchanfragen mit anderen Suchkriterien zu kombinieren, wobei Ihnen der volle Umfang der Verity-Anfragesprache zur Verfügung steht. Beispiel:

```
<#FREETEXT> ("freiwillige feuerwehr in ländlichen regionen") <#AND> (DATE >
 20101224000000)
```

Die Anführungszeichen müssen angegeben werden. Wenn Anführungszeichen im Freitext vorkommen, muss ihnen wie im folgenden Beispiel jeweils ein Rückwärts-Schrägstrich (*engl.* backslash) vorausgehen:

```
<#FREETEXT> ( "\"Independence Day\""), ("\"The Arrival\""), science fiction" )
```

Bitte beachten Sie folgendes: In Fällen, in denen eine Anfrage oder ein Dokument nur Wörter enthält, die in der zur Collection gehörenden Datei style.stp als Stoppwörter definiert sind, verwendet der Freitext-Parser die Stoppwörter, um die Anfrage zu konstruieren und ignoriert die Stoppwörterliste.

Der FREETEXT-Operator kann auf die gleiche Weise wie der Operator ACCRUE mit anderen Operatoren kombiniert werden.

5.3.6 > (größer)

Wählt Dokumente aus, deren Dokumentfelder größere Werte enthalten als die angegebene gesuchte Zeichenkette. So kann man folgendermaßen nach Dokumenten suchen, die zuletzt nach dem 23. Dezember 2010, 0 Uhr, geändert wurden:

```
lastChanged > 20101223000000
```

5.3.7 >= (größer oder gleich)

Wählt Dokumente aus, in deren Dokumentfeldern Werte enthalten sind, die größer als die angegebene gesuchte Zeichenkette sind oder ihr entsprechen. Mit der folgenden Anfrage kann man nach Dokumenten suchen, die zuletzt am 23. Dezember 2010, 0 Uhr, oder später geändert wurden:

```
lastChanged >= 20101223000000
```

5.3.8 < (kleiner)

Wählt Dokumente aus, deren Dokumentfelder kleinere Werte enthalten als die angegebene gesuchte Zeichenkette. So kann man beispielsweise folgendermaßen nach Dokumenten suchen, die vor dem 23. Dezember 2010, 0 Uhr, zuletzt geändert wurden:

```
lastChanged < 20101223000000
```

5.3.9 <= (kleiner oder gleich)

Wählt Dokumente aus, in deren Dokumentfeldern Werte enthalten sind, die kleiner als die angegebene gesuchte Zeichenkette sind oder ihr entsprechen. Mit der folgenden Anfrage kann man nach Dokumenten suchen, die zuletzt am 23. Dezember 2010, 0 Uhr, oder früher geändert wurden:

```
lastChanged <= 20101223000000
```

5.3.10 LIKE

Der Operator `LIKE` ist in erster Linie für den Einsatz in der Applikationsentwicklung vorgesehen. Er sucht nach Dokumenten, die einem oder mehreren vorgegebenen Dokumenten oder Texpassagen ähnlich sind. Die Suchmaschine analysiert den vorgegebenen Text, um die für die Suche wichtigsten Begriffe ausfindig zu machen. Wenn mehrere Texte vorgegeben werden, geht die Suchmaschine davon aus, dass alle diese Texte das gleiche Thema haben und wählt aus allen Texte Begriffe aus, die textübergreifend relevant sind. Die ermittelten Dokumente werden nach Relevanz gewichtet.

Der `LIKE`-Operator hat genau einen Operanden, die sogenannte QBE-Spezifikation (QBE = *engl.* "query-by-example"). Die QBE-Spezifikation kann entweder der wörtliche Text des angegebenen Beispiels oder die Spezifikation eines oder mehrerer ganzer Dokumente und Textpassagen sein, die als positive und negative Beispiele verwendet werden sollen.

Bitte beachten Sie folgendes: Enthält eine Suchanfrage oder ein als Beispiel angegebenes Dokument nur Stoppwörter, so ermittelt eine Anfrage mit dem `LIKE` -Operator keine Dokumente, d. h. das Suchergebnis enthält in diesem Fall keine Treffer.

Syntax

Dokumente werden als Textreferenzen spezifiziert, die in geschweifte Klammern eingeschlossen sind. Referenzen werden mit der folgenden Syntax angegeben:

```
{[name =]type:value [name=]type:value ...}
```

Die Variablen haben die folgende Bedeutung:

- *name* kann entweder `posex` (positives Beispiel), oder `negex` (negatives Beispiel) sein. Ein negatives Beispiel reduziert das Gewicht von Begriffen, wenn diese in einem positiven Beispiel auftreten. Wenn Begriffe in einem negativen Beispiel nicht im positiven Beispiel vorkommen, so bleibt das negative Beispiel wirkungslos. (Ein negatives Beispiel hat für sich allein genommen folglich keinen Sinn). Die Variable `name` ist optional. Wird sie nicht angegeben, so wird `name` intern auf `posex` gesetzt. In diesem Fall darf auch das Gleichheitszeichen nicht angegeben werden.
- *type* kann einer der folgenden Werte sein:
  - `VdkVgwKey`, um ein Dokument über seine externe ID zu referenzieren (d. h. die Versions-ID im Content Manager oder die Datei-ID in der Template Engine).
  - `Text`, um den Text direkt anzugeben.
- *value* ist eine Referenz auf ein Textfragment, das als positives bzw. negatives Beispiel dient. Der Wert von *value* hängt vom Wert von *type* ab:
  - `VdkVgwKey`: die Dokument-ID (d. h. Versions- oder Datei-ID)
  - `Text`: Literaler Text.

Wenn *type* nicht angegeben wurde, wird der Wert *value* folgendermaßen interpretiert:

- als literaler Text, wenn er mit einem Anführungszeichen beginnt;
- als VdkVgwKey in allen anderen Fällen.

Der LIKE-Operator kann nach den gleichen Regeln mit anderen Operatoren kombiniert werden wie der Operator ACCRUE.

Beispiele

Die folgenden Beispiele illustrieren, wie der LIKE-Operator verwendet wird. Ganze Zahlen stellen immer eine Versions- oder Datei-ID dar.

Nur literaler Text:

```
<#LIKE> ("Nachts schlafen die Ratten doch.")
```

Explizite Angabe eines einzigen positiven Beispiels:

```
<#LIKE> ( "{posex=vdkvgwkey:650431}" )
```

Explizite Angabe mehrerer positiver und negativer Beispiele:

```
<#LIKE> ( "{posex=vdkvgwkey:7369 posex=vdkvgwkey:8457
negex=text:"freiwillige feuerwehr"}" )
```

Implizite Referenztypen:

```
<#LIKE> ("{posex=7369 posex=8457 negex=\"freiwillige feuerwehr\"}")
```

Implizite posex-Namen:

```
<#LIKE> ( "{vdkdocid:7369 vdkvgwkey:8457}" )
```

Implizite Referenztypen und implizite posex-Namen:

```
<#LIKE> ( "{#7369 8457}" )
```

Man kann eine Textreferenz mit literalem Text kombinieren:

```
<#LIKE> ( "{#7369 8457} und mehr text" )
```

Die vorige QBE ist äquivalent zu dieser:

```
<#LIKE> ( "{#7369 8457 text: \"und mehr text\"}" )
```

Die einfachste Art, ein einziges positives Beispiel mit Referenz auf eine ID anzugeben:

```
<#LIKE> ( "{650431}" )
```

Anführungszeichen, die in LIKE-Ausdrücken enthalten sind, müssen Rückwärts-Schrägstriche (Backslashes) vorausgehen. Der Backslash zeigt der Suchmaschine an, dass das darauf folgende Zeichen als literales Zeichen behandelt werden soll.

Überlegungen zur Effizienz

Um einen LIKE-Ausdruck zu verarbeiten, muss die Suchmaschine den gesamten Text der Beispiele in der QBE-Spezifikation analysieren. Dies kann sehr zeitaufwändig sein, insbesondere, wenn die Beispieldokumente groß sind oder aufwändig gefiltert werden müssen.

5.3.11 MATCHES

Dieser Operator wählt Dokumente aus, indem die angegebene Zeichenkette mit den Werten verglichen wird, die in einem bestimmten Dokumentfeld gespeichert sind. Es werden nur Dokumente ausgewählt, wenn die angegebenen Suchbegriffe exakt mit den Werten in den Feldern übereinstimmen.

Sie können Fragezeichen (?) verwenden, um einzelne variable Zeichen in einer Zeichenkette darzustellen. Ein Sternchen (*) passt auf mehrere Zeichen in einer Zeichenkette.

Angenommen, das Dokumentfeld mit dem Namen QUELLE enthält die folgenden Werte:

```
COMPUTER
COMPUTERWORLD
COMPUTER CURRENTS
PC COMPUTING
```

Um Dokumente zu finden, deren Quelle COMPUTER ist, wird der MATCHES-Operator folgendermaßen verwendet:

QUELLE <#MATCHES> computer

In diesem Fall stellt der MATCHES-Operator eine Übereinstimmung mit COMPUTER fest, nicht jedoch mit COMPUTERWORLD, COMPUTER CURRENTS oder PC COMPUTING.

Um Dokumente zu finden, deren Quelle COMPUTERWORLD ist, kann der MATCHES-Operator so verwendet werden:

```
QUELLE <#MATCHES> computer?????
```

Nun findet MATCHES eine Übereinstimmung mit COMPUTERWORLD, weil jedes Fragezeichen für ein einzelnes Zeichen an einer bestimmten Position innerhalb der Zeichenkette steht. Mit COMPUTER und COMPUTER CURRENTS gibt es keine Übereinstimmung, weil die Länge dieser Zeichenketten nicht der Länge des Suchbegriffs entspricht.

Um Dokumente zu finden, deren Quelle COMPUTER, COMPUTERWORLD oder COMPUTER CURRENTS ist, wird der MATCHES-Operator in der folgenden Art und Weise verwendet:

```
QUELLE <#MATCHES> computer*
```

Hier findet MATCHES Dokumente, in denen QUELLE den Werten COMPUTER, COMPUTERWORLD oder COMPUTER CURRENTS entspricht, weil das Sternchen (*) für jede Anzahl beliebiger Zeichen am Ende der Zeichenkette steht.

Um Dokumente zu finden, deren Quelle COMPUTER, COMPUTERWORLD, COMPUTER CURRENTS oder PC COMPUTING ist, kann der MATCHES-Operator so verwendet werden:

```
QUELLE <#MATCHES> *comput*
```

So findet MATCHES alle vier Vorkommen, weil dem Sternchen (*) eine beliebig lange Zeichenkette jeweils am Anfang und am Ende entspricht.

5.3.12 != (ungleich)

Wählt die Dokumente aus, deren Dokumentfelder nicht die angegebene gesuchte Zeichenkette enthalten. Um beispielsweise nach Dokumenten zu suchen, die nicht zu Dateien vom Typ document gehören, verwendet man die folgende Suchanfrage:

```
objType != document
```

Mit Hilfe des NOT-Modifikators und einem Test auf Gleichheit kann man die gleiche Suche auslösen:

```
<#NOT>(objType=document)
```

Beide Anfragen geben zwar das gleiche Ergebnis zurück, die erste Form der Anfrage ist jedoch deutlich effizienter als die zweite.

5.3.13 PRODUCT

Mit diesem Operator wird jedes Dokument wird in Bezug auf jeden Suchausdruck separat gewichtet, und die Einzelergebnisse werden anschließend miteinander multipliziert. Das Ergebnis dieser Operation wird abschließend durch 100 dividiert.

Die folgende Anfrage ist ein Beispiel für die Suchsyntax:

```
<#PRODUCT> ("computer","laptop")
```

Wenn die Suche nach "computer" 50 Punkte und nach "laptop" 20 Punkte ergeben würde, hätte die obige Suchanfrage 10 als Ergebnis (das Produkt der beiden Werte, dividiert durch 100).

5.3.14 STARTS

Wählt Dokumente aus, indem die angegebene Zeichenkette mit den Zeichen verglichen wird, mit denen die in einem bestimmten Dokumentfeld gespeicherten Werte beginnen. So können Sie nach Dokumenten suchen, die zuletzt am 23. Dezember 2010 geändert wurden, indem Sie im Feld lastChanged nach "20101223" suchen. Da die in diesem Feld gespeicherten Werte nicht nur das Datum, sondern auch die Uhrzeit enthalten, müssen die Feldwerte mit STARTS verglichen werden:

```
lastChanged <#STARTS> 20101223
```

Datumsangaben werden vom Content Manager und von der Template Engine als 14-stellige Ziffernfolgen folgendermaßen indiziert: Jahr (vierstellig), Monat, Tag, Stunde, Minute, Sekunde (jeweils zweistellig).

5.3.15 SUBSTRING

Wählt Dokumente aus, indem die angegebene Zeichenkette mit Teilen der Werte verglichen wird, die in einem bestimmten Dokumentfeld gespeichert sind. Die Zeichen, aus denen die gesuchte Zeichenkette besteht, kann am Anfang, am Ende oder mitten im Wert eines Feldes auftreten.

Sie können beispielsweise folgendes eingeben, um die Dokumente zu ermitteln, deren Titel Wörter wie "Resolution", "Solarenergie" oder "Besoldung" enthalten:

```
title <#SUBSTRING> sol
```

5.3.16 SUM

Berechnet die Punktzahl der Dokumente, die durch eine Suchanfrage ermittelt werden, indem die Punktzahlen der Einzelergebnisse der Suchbestandteile bis zu einem Höchstwert von 100 addiert werden. Beispiel für die Suchsyntax:

```
<#SUM> ("computer","laptop")
```

Wenn die Suche nach "computer" 50 Punkte und nach "laptop" 20 Punkte erzielen würde, hätte die vorausgegangene Suche 70 als Ergebnis. Hätte die Suche nach "computer" 50 Punkte und nach "laptop" 75 Punkte erzielt, so wäre das Ergebnis 100 (das Maximum).

5.3.17 YESNO

Erzwingt, dass ein Suchbestandteil mit 100 Punkten bewertet wird, wenn er eine Punktzahl ungleich 0 erhalten hat. Einige Beispiele sollen dies verdeutlichen:

```
<#YESNO> ("Chloe")
```

Wenn das Ergebnis der Suche nach "Chloe" mit 75 Punkten bewertet wird, so wird es durch den YESNO-Operator auf 100 gesetzt. Ist das Ergebnis dagegen 0, so bleibt es 0.

Mit dem Operator kann man eine Suche auf jene Dokumente beschränken, auf die eines der Suchkriterien zutrifft, ohne dass die Bewertung dieses Kriteriums die endgültige Bewertung beeinflusst. Um beispielsweise nach Dokumenten zu suchen, die "Chloe" und "Mead" enthalten, wobei "Mead" allein zur Bewertung des Dokuments herangezogen wird, kann man nicht einfach folgendes eingeben:

“Chloe” <#AND> “Mead”

Dies würde die ermittelten Dokumente mit einer aus beiden Suchbestandteilen kombinierten Bewertung versehen. Mit dem Folgenden könnten Sie das Gewünschte erreichen:

```
<#YESNO> ("Chloe") <#AND> "Mead"
```

Wenn die Einzelergebnisse der Suche nach "Chloe" mit 50 und nach "Mead" mit 75 Punkten bewertet würden, so wäre das kombinierte Ergebnis ohne den YESNO-Operator 50, mit YESNO-Operator jedoch 75, weil die Ergebniszahl beim AND-Operator das niedrigste aller Einzelergebnisse ist.

5.4 Übersicht der Modifikatoren

Modifikatoren werden zusammen mit Operatoren verwendet. Ein Modifikator ändert das normale Verhalten eines Operators. So können Sie beispielsweise den Modifikator `CASE` zusammen mit einem Operator verwenden, um zu erreichen, dass Groß- und Kleinschreibung als Bestandteil der Suche berücksichtigt wird. Die verfügbaren Modifikatoren sind `CASE`, `MANY`, `NOT` und `ORDER`.

Es gibt zwei syntaktische Formen, in denen Modifikatoren zusammen mit Operatoren angegeben werden können. Beim ersten Format gibt man den Namen des Modifikators vor dem Namen des Operators an, wie es in der folgenden Tabelle gezeigt wird. Beachten Sie bitte, dass der größte Teil der Modifikatoren nur mit bestimmten Operatoren verwendet werden kann.

Modifikator	Erlaubte Operatoren	Beispiele
`CASE`	`WORD` `WILDCARD`	`<#CASE><#WORD>` `LehrerInnen`

`MANY`	`WORD`
	`WILDCARD`
	`STEM`
	`SOUNDEX`
	`PHRASE`
	`SENTENCE`
	`PARAGRAPH`
		`<#MANY><#WORD> virtuell`

`NOT`	Alle Operatoren	`Katze <#AND> Hund <#AND> <#NOT> Haustier`

`ORDER`	`PARAGRAPH`
	`SENTENCE`
	`NEAR/N`
	`ALL`
		`Kanzler <#ORDER> <#PARAGRAPH> Berlin`
		
		`<#ORDER> <#SENTENCE> ("Kanzler", "Berlin")`

Bei der zweiten syntaktischen Form gibt man den Namen des Modifikators und den Operatornamen folgendermaßen an:

```
<#OperatorName/ModifikatorName>
```

Diese zweite Form ist nur bei den Modifikatoren `CASE` und `NOT` zulässig.

Modifikator	Erlaubte Operatoren	Beispiele

`CASE`	`WORD`
	`WILDCARD`
	`CONTAINS`
	`MATCHES`
	`STARTS`
	`ENDS`
	`SUBSTRING`
		`autor <#CONTAINS/CASE> Don`

`NOT`	Alle Operatoren	`autor <#CONTAINS/NOT> don` `autor <#STARTS/NOT> Mc`

5.4.1 CASE

Verwenden Sie den CASE-Modifikator mit dem Operator WORD oder WILDCARD, um eine Suche durchzuführen, die die Groß- und Kleinschreibung im angegebenen Suchbegriff berücksichtigt. Der Modifikator braucht nur dann verwendet zu werden, wenn auch Suchbegriffe, die ausschließlich groß oder klein geschriebene Zeichen enthalten, in dieser Schreibweise in den Dokumenten vorkommen müssen. Bei gemischter Groß- und Kleinschreibung findet die Search Cartridge automatisch nur Übereinstimmungen unter Berücksichtigung der Schreibweise.

Um den Modifikator CASE einzusetzen, gibt man einfach die gesuchten Wörter so ein, wie sie in den ermittelten Dokumenten auftreten sollen, ausschließlich in Großbuchstaben, ausschließlich in Kleinbuchstaben oder gemischt.

Um beispielsweise die Dokumente zu finden, die das Wort "siegen" in dieser Schreibweise enthalten, können Sie das Folgende eingeben:

```
<#CASE> <#WORD> siegen
```

Dadurch werden nur solche Dokumente ausgewählt, die das Wort "siegen" enthalten. Vorkommen der Wörter "Siegen", "siegend" und "SIEGEN" werden nicht berücksichtigt.

5.4.2 MANY

Dieser Modifikator berechnet die Dichte von Wörtern, abgeleiteten Wörtern und Wortfolgen in einem Dokument und gewichtet die ermittelten Dokumente entsprechend. Je häufiger ein Wort, ein abgeleitetes Wort oder eine Wortfolge in einem Dokument im Verhältnis zu dessen Textmenge vorkommt, desto höher wird das Dokument bewertet, wenn es ins Suchergebnis aufgenommen wird. Da der Modifikator MANY die Dichte als Verhältnis der Häufigkeit zur Textmenge berechnet, kann ein längeres Dokument, in dem ein Wort häufiger vorkommt, dennoch eine niedrigere Wertung erhalten als ein kürzeres Dokument, in dem das Wort seltener vorkommt. Sie können den Modifikator MANY zusammen mit den folgenden Operatoren verwenden: WORD, WILDCARD, STEM, SOUNDEX, PHRASE, SENTENCE, PARAGRAPH.

Um beispielsweise Dokumente nach der Dichte des Wortes "Apfel" und davon abgeleiteten Varianten auszuwählen, können Sie folgendes eingeben:

```
<#MANY> <#STEM> apfel
```

Um Dokumente nach der Dichte der Phrase "bürgerliche gesellschaft" auszuwählen, können Sie Folgendes eingeben:

```
<#MANY> bürgerliche gesellschaft
```

Der Modifikator MANY kann nicht zusammen mit den Operatoren AND, OR und ACCRUE und auch nicht zusammen mit Feld- und Vergleichsoperatoren verwendet werden.

5.4.3 NOT

Verwenden Sie den Modifikator NOT mit einem Wort oder einer Wortfolge, um Dokumente auszuschließen, die dieses Wort oder diese Wortfolge enthalten. Um beispielsweise nur die Dokumente

auszuwählen, die die Wörter "Katze" und "Maus", nicht jedoch das Wort "Hund" enthalten, können Sie folgendes eingeben:

```
Katze <#AND> Maus <#AND> <#NOT> Hund
```

5.4.4 ORDER

Verwenden Sie den Modifikator ORDER, um festzulegen, dass die Suchbestandteile in der gleichen Reihenfolge auftreten müssen wie in der Anfrage. Wenn die Suchbegriffe in einem Dokument nicht in der angegebenen Reihenfolge vorkommen, ist das Dokument kein Treffer. Sie können den ORDER-Modifikator zusammen mit diesen Operatoren verwenden: PARAGRAPH, SENTENCE, NEAR/*N* und ALL.

Geben Sie den Modifikator ORDER immer unmittelbar vor dem Operator an. Die folgenden Beispiele zeigen, wie Sie die einfache und die explizite Syntax verwenden können, um Dokumente zu ermitteln, die die Wörter "Kanzler" und "Berlin" in dieser Reihenfolge und im gleichen Absatz enthalten:

Einfache Syntax:

```
Kanzler <#ORDER><#PARAGRAPH> Berlin
```

Explizite Syntax:

```
<#ORDER><#PARAGRAPH> ("Kanzler", "Berlin")
```

Um nach Dokumenten zu suchen, die die Wörter "Taucher", "tötet" und "Hai" in dieser Reihenfolge und mit einem maximalen Abstand von jeweils 20 Wörtern enthalten, können Sie eine der folgenden Anfragen verwenden:

```
Taucher <#ORDER><#NEAR/20> tötet <#ORDER><#NEAR/20> Hai<
#ORDER> <#NEAR/20> (Taucher, tötet, Hai)
```

Man kann den Operator NEAR/*N* zusammen mit dem Modifikator ORDER verwenden, um das gleiche Verhalten wie beim PHRASE-Operator zu erzielen. So können Sie folgendes angeben, um nach Dokumenten zu suchen, die die Wortfolge "world wide web" enthält:

```
world <#ORDER><#NEAR/1> wide <#ORDER><#NEAR/1> web
```

Um nach einem Wort zu suchen, das sich zwischen zwei anderen Wörtern befindet, können Sie den ORDER-Modifikator folgendermaßen zusammen mit dem Operator ALL verwenden:

```
<#ORDER><#ALL>(Hund, Katze, Maus)
```

Die obige Anfrage sucht nach dem Wort "Katze" zwischen den Wörtern "Hund" und "Maus". Auch Dokumente, die von diesen Wörtern abgeleitete Varianten enthalten, passen auf diese Anfrage.

Die Anfrage nach einem Wort zwischen anderen Wörtern kann so erweitert werden, dass sie weitere Suchausdrücke enthält. Beispiel:

```
<#ORDER><#ALL>(Hund, kleine Katze, Maus)
```

Die obige Anfrage sucht nach der Wortfolge "kleine Katze" zwischen den Wörtern "Hund" und "Maus". Auch in diesem Fall führen abgeleitete Varianten der Wörter zu einem Treffer.

5.5 Bewertung der Suchergebnisse

Die Suchergebnisse einer Anfrage an die Autonomy-Suchmaschine können nach einem oder mehreren Feldern sortiert werden. Am üblichsten ist die Sortierung nach `score`. Hierbei handelt es sich um einen von der Suchmaschine berechneten Wert – eine Punktzahl –, der die Relevanz dieses Treffers in Bezug auf die Suchanfrage ausdrückt.

Der Score für ein einzelnes Dokument ergibt sich daraus, dass die Suchmaschine jedem Suchterm einen Score zuordnet (die genauen Algorithmen hierfür sind nicht bekannt) und diese Teil-Scores abhängig von den Suchoperatoren miteinander verrechnet:

- AND – Minimum der Teil-Scores
- OR – Maximum der Teil-Scores
- YESNO – ein positiver Teil-Score wird zu 1, 0 bleibt 0
- ANY – wie YESNO (OR ...)
- [*xx*] – multipliziert den Teil-Score mit 0.*xx*

Beispiele

Nimmt man folgende Teil-Scores für ein Suchergebnis an:

- `"Teddy"` liefert 0.4
- `"Bär"` liefert 0.8
- `"admins" <IN> permissionLiveServerRead` liefert 0.77
- `"free" <IN> noPermissionLiveServerRead` liefert 0

Dann ergibt sich aus den obigen Verknüpfungsregeln:

- `"Teddy" <OR> "Bär"` liefert max(0.4, 0.8) -> 0.8
- `"Teddy" <AND> "Bär"` liefert min(0.4, 0.8) -> 0.4
- `[90]"Teddy" <#OR> [10]"Bär"` liefert max(0.36, 0.08) -> 0.36
- `<#ANY> (("admins" <#IN> permissionLiveServerRead), ("free" <#IN> noPermissionLiveServerRead))` liefert `YESNO`(max(0.77, 0) -> 1

Weitere Details zur Score-Berechnung finden Sie in der Dukumentation der einzelnen Operatoren und Modifikatoren, siehe insbesondere MANY und ACCRUE.

6

6 MISE, das XML-Protokoll des Search Engine Servers

6.1 Payloads

Die XML-Dokumente, die Clients über die XML-Schnittstelle mit dem Search Engine Server austauschen, werden als "Payloads" bezeichnet. Das Element `ses-payload` ist das Wurzelelement aller Anfrage- und Antwort-Dokumente:

```
<!ELEMENT ses-payload (ses-header, (ses-response+ | ses-request+))>
<!ATTLIST ses-payload
  payload-id CDATA #REQUIRED
  timestamp CDATA #REQUIRED
  version CDATA #REQUIRED
>
```

Die Attribute des `ses-payload`-Elements haben die folgende Bedeutung:

- `payload-id`
  Identifikator des Payloads. Dieser Identifikator wird vom Erzeuger des Payloads generiert und muss innerhalb eines Kommunikationskontexts eindeutig sein. Ein solcher Kontext wird durch den Search Engine Server (beispielsweise einer Firma) und alle Clients gebildet, die mit dem Server kommunizieren. In der Regel wird die `payload-id` durch einen Algorithmus generiert.
- `timestamp`
  Datum und Uhrzeit (Zeitstempel) der Erzeugung des Payloads. Der Zeitstempel muss in kanonischer Form als 14stelliger String (von links beginnend: Jahr vierstellig, Monat zweistellig, Tag zweistellig, Stunde zweistellig, Minute zweistellig, Sekunde zweistellig) in GMT angegeben sein (Beispiel: 20110716020223).
- `version`
  Version des XML-Schnittstellenprotokolls. Der Aufbau des Payloads ist von der Version abhängig. Zum Zeitpunkt der Fertigstellung dieses Handbuchs hat das XML-Schnittstellenprotokoll die Version 2.1.

Ein Client gibt bei einer Anfrage mit dem Wert des `version`-Attributs an, welche Version des XML-Schnittstellenprotokolls er verwendet.

Der Search Engine Server unterstützt neben der aktuellen Version des Protokolls alle Versionen, die bisher gültig waren. Verwendet der Client eine dieser Versionen, so erzeugt der Server einen Antwort-Payload in dieser Version. Andernfalls antwortet er mit einer Fehlermeldung, die die Protokollinkompatibilität mitteilt. Diese Meldung erzeugt der Server in seiner aktuellen Version des XML-Schnittstellenprotokolls.

6.1.1 Header-Element

Das erste Unterelement aller Payloads ist `ses-header`. Das `ses-header`-Element enthält Informationen über die Identität der beiden Parteien, die das Payload austauschen.

```
<!ELEMENT ses-header (ses-sender, ses-receiver?, ses-authentication?)>
<!ELEMENT ses-sender EMPTY>
<!ATTLIST ses-sender
  sender-id CDATA #REQUIRED
  name CDATA #REQUIRED>
<!ELEMENT ses-receiver EMPTY>
<!ATTLIST ses-receiver
  receiver-id CDATA #REQUIRED
  name CDATA #REQUIRED>
<!ELEMENT ses-authentication EMPTY>
<!ATTLIST ses-authentication
  login CDATA #REQUIRED
  password CDATA #REQUIRED>
```

ses-sender

Das `ses-sender`-Element muss immer als Unterelement des `ses-header`-Elements erscheinen. Es spezifiziert die Identität des Absenders des Payloads. Die Attribute des `ses-sender`-Elements haben die folgende Bedeutung:

- `sender-id`
  Identifikator des Absenders des Payloads. Jeder CMS-Server-Anwendung und den Clients wird bei der Installation jeweils ein Identifikator zugewiesen, der innerhalb des Kommunikationskontextes eindeutig ist. Dieser Identifikator wird bei der Erzeugung des Payloads als `sender-id` verwendet.
- `name`
  Name des Absenders des Payloads. Der Name ist eine Zeichenkette, die die Anwendung bezeichnet, die den Payload überträgt. Der Search Engine Server verwendet als Name *SES*.

ses-receiver

Das `ses-receiver`-Element spezifiziert die Identität des gewünschten Empfängers des Payloads. Dieses Element ist optional, sofern es zwischen dem Server und einem Client eine individuelle Netzwerkverbindung gibt. In diesem Fall sind der Absender des Payloads und der Empfänger eindeutig identifiziert. Befindet sich allerdings zwischen dem Server und den Clients ein Proxy-Server, der mehrere Clients oder Server bedient, so können Server und Client das `ses-receiver` -Element nutzen, um dem Proxy-Server den Empfänger mitzuteilen. Die Attribute des `ses-receiver`-Elements haben die folgende Bedeutung:

- `receiver-id`
  Identifikator des Empfängers des Payloads. Dieses Attribut hat dieselbe Semantik wie das `sender-id` -Attribut des `ses-sender`-Elements. Es enthält den innerhalb eines Kommunikationskontextes eindeutigen Identifikator des Servers oder Clients, der das Payload empfangen soll. Bei Antwort-Payloads ist dieses Attribut immer eine Kopie des `sender-id-` Attributs des `ses-sender-` Elements im entsprechenden Anfrage-Payload.
- `name`
  Name des Empfängers. Der Name ist eine Zeichenkette, die die gewünschte Empfängeranwendung bezeichnet. Bei Antwort-Payloads ist dieses Attribut eine Kopie des `name-` Attributs des `ses-sender`-Elements im entsprechenden Anfrage-Payload.

ses-authentication

Das `ses-authentication`-Element ist optional. Es kann nur bei Anfrage-Payloads verwendet werden. Es enthält Informationen über den Benutzer, für den die Anfrage bearbeitet werden soll. Die Attribute des `ses-authentication`-Elements haben die folgende Bedeutung:

- `login`
  Der Anmeldename (das Login) des Benutzers des Search Engine Servers.
- `password`
  Das Passwort des Benutzers (Klartext).

6.1.2 Request-Element

Bei Anfrage-Payloads folgen auf das `ses-header`-Element im `ses-payload`-Wurzelelement ein oder mehrere `ses-request`-Elemente. Diese Elemente spezifizieren die Operationen, die vom Search Engine Server ausgeführt werden sollen.

```
<!ELEMENT ses-request (ses-indexDoc|ses-deleteDoc|ses-search)>
<!ATTLIST ses-request
  request-id CDATA #REQUIRED
  preclusive (true | false) "false">
```

Die Unterelemente des `ses-request`-Elements werden in den folgenden Abschnitten erläutert. Ein `ses-request`-Element hat folgende Attribute:

- `request-id`
  Identifikator des Requests. Dieser Identifikator wird vom Erzeuger des Anfrage-Payloads vergeben. Er muss innerhalb aller Payloads, die in einem bestimmten Kommunikationskontext ausgetauscht werden, eindeutig sein. Es reicht nicht aus, dass die ID innerhalb des aktuellen Payloads eindeutig ist.
- `preclusive`
  Wahrheitswert (`true` oder `false`). Das `preclusive`-Attribut erlaubt dem SES-Client, die Requests zu markieren, die für die weitere Bearbeitung des Payloads kritisch sind. Wenn die Bearbeitung eines als `preclusive` markierten Requests fehlschlägt, werden alle weitere Requests im Payload nicht bearbeitet, sondern mit einer Fehlermeldung beantwortet.

Alle Anfragen in einem Payload werden sequenziell, beginnend bei der ersten Anfrage, bearbeitet. Dadurch ist es einem Client möglich, auch voneinander abhängende Requests in einem einzigen Anfrage-Payload unterzubringen. Mit dem `preclusive`-Attribut kann der Client ferner sicherstellen, dass die abhängige Anfrage nur dann bearbeitet wird, wenn die vorausgehende keinen Fehler erzeugt hat.

6.1.3 Response-Element

Bei Antwort-Payloads enthält das `ses-payload`-Wurzelelement ein oder mehrere `ses-response`-Elemente nach dem `ses-header`-Element, mit denen die jeweiligen Ergebnisse der vom Server ausgeführten Operationen zurückgeliefert werden.

Wenn ein Anfrage-Payload vom Search Engine Server vollständig erkannt und bearbeitet wurde, entspricht jedes `ses-response`-Element im Antwort-Payload genau einem `ses-request`-Element des Anfrage-Payloads. In diesem Fall enthalten die `ses-response`-Elemente Meldungen, die sich auf die Inhalte der Requests beziehen (Request-Level-Meldungen).

Erhält der Search Engine Server dagegen ein ungültiges Anfrage-Payload (beispielsweise ohne `ses-header`-Element oder mit nicht erkennbaren `ses-request`-Elementen), so liefert er ein Antwort-Payload zurück, das nur ein `ses-response`-Element enthält, mit dem der allgemeine Fehler gemeldet wird (siehe Payload-Fehler). In diesem Fall bezieht sich das `ses-response`-Element auf den Payload (Payload-Level-Meldung). Ein `ses-response`-Element ist folgendermaßen aufgebaut.

```
<!ELEMENT ses-response (ses-code*)>
<!ATTLIST ses-response
  response-id CDATA #REQUIRED
  payload-id CDATA #IMPLIED
  request-id CDATA #IMPLIED
  success (true | false) #REQUIRED>
```

Die einzelnen Antworten auf die Requests werden in `ses-code`-Elementen zurückgegeben:

```
<!ELEMENT ses-code ANY>
<!ATTLIST ses-code numeric CDATA #REQUIRED phrase CDATA #REQUIRED>
```

Die Attribute des `ses-response`-Elements haben die folgende Bedeutung:

- `response-id`
  Identifikator der Antwort. Der Identifikator wird vom Erzeuger des Antwort-Payloads gesetzt. Er muss innerhalb aller Payloads, die in einem Kommunikationskontext ausgetauscht werden, eindeutig sein.
- `payload-id`
  Identifikator des Anfrage-Payloads. Er entspricht dem `payload-id` -Attribut des Anfrage-Payloads. Dieses Attribut wird vom Server nur dann eingefügt, wenn das `ses-response`-Element eine Payload-Level-Meldung im ersten `ses-code`-Element enthält. Ein Client kann folglich am Vorhandensein dieses Attributs erkennen, ob sein Anfrage-Payload als solches (nicht die in ihm enthaltenen Anfragen) vom Server interpretiert werden konnte.
- `request-id`
  Identifikator des Requests. Er entspricht dem `receiver-id`-Attribut des `ses-request-` Elements im Anfrage-Payload. Dieses Attribut ist nur vorhanden, wenn das `ses-code`-Element eine Request-Level-Meldung enthält.
- `success`
  Der Wert dieses Attributs ist `true`, wenn der Request erfolgreich bearbeitet werden konnte. Andernfalls ist er `false`.

Die Ergebnisse der ausgeführten Operationen werden mittels `ses-code`-Elementen innerhalb des `ses-response`-Elements zurückgeliefert. Die `ses-code`-Elemente enthalten eine Erfolgs- oder Fehlermeldung und weitere XML-Elemente, die das Resultat der Operation oder gegebenenfalls Fehlerinformationen darstellen.

Die Attribute des `ses-code`-Elements haben die folgende Bedeutung:

- `numeric`
  Fehlernummer (oder Erfolgsmeldungsnummer). Die den Nummern entsprechenden Meldungen werden im Abschnitt Fehlerbehandlung beschrieben.
- `phrase`
  Die Beschreibung des Fehlers.

Der Inhalt des `ses-code`-Elements hängt von der Operation ab, die im Request angegeben war. Bei Operationen, die nicht erfolgreich ausgeführt werden konnten, hängt der Inhalt des `ses-code`-

Elements vom aufgetretenen Fehler ab. Die möglichen Inhalte des `ses-code`-Elements werden für jede Operation in den folgenden Abschnitten aufgeführt.

6.2 Indizierungsanfragen

6.2.1 Request

Eine Indizierungsanfrage wird mit dem `ses-indexDoc`-Element in einem `ses-request`-Element kodiert. Es folgt ein Beispiel, das gleichzeitig zeigt, dass sich mehrere Requests in einem Payload befinden können:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ses-payload SYSTEM "http://www.infopark.com/ses.dtd">
  <ses-payload payload-id="B42TE241" timestamp="20100825172100" version="2.1">
  <ses-header>
    <ses-sender sender-id="FX45RTDT" name="CM-Server"/>
    <ses-authentication login="cm-server" password=""/>
  </ses-header>
  <ses-request request-id="BR12TI5X">
    <ses-indexDoc docId="4712" collection="collection1"
         mimeType="application/ms-word" usesStreaming="YES">
      <title encoding="plain">testdoc1</title>
      <customAttribute encoding="base64">MGHX2c5=</customAttribute>
      <blob encoding="stream">d--1157180779-000000001-X</blob>
    </ses-indexDoc>
  </ses-request>
  <ses-request request-id="BR12TI5Y">
    <ses-indexDoc docId="4713" collection="collection2">
      <title encoding="plain">testdoc2</title>
      ...
      <blob>Just the blob</blob>
    </ses-indexDoc>
  </ses-request></ses-payload>
```

Das `ses-indexDoc`-Element hat die folgenden Attribute:

- `docId`
  Die ID des zu indizierenden Dokuments. Dies ist üblicherweise die Versions-ID (Content Management Server) oder die Datei-ID (Template Engine).
- `collection`
  Der Name der Collection, zu der das zu indizierende Dokument hinzugefügt werden soll.
- `mimeType`
  Der MIME-Typ des zu indizierenden Dokuments. Anhand des MIME-Typs entscheidet der Search Engine Server, mit welchem Präprozessor das Dokument vorverarbeitet werden soll (siehe Suche und Indizierung konfigurieren).
- `usesStreaming`
  `YES`, wenn mindestens eines der zu indizierenden Felder über das Streaming-Interface zum Search Engine Server übertragen wurde, andernfalls `NO`.

Das `ses-indexDoc`-Element enthält als Unterelemente alle in Abschnitt Content-Indizierung aufgeführten Attribute, von denen in das obige Beispiel exemplarisch nur `title`, das kundenspezifische Attribut `customAttribute` und `blob` aufgenommen wurden. Wie die Inhalte der zu indizierenden Datei- und Versionsfelder kodiert sind, wird mit dem Tag-Attribut `encoding` in den jeweiligen Attribut-Tags angegeben. `encoding` kann einen der drei folgenden Werte haben:

- `plain`

Der Wert des zu indizierenden Attributs ist nicht kodiert und direkt als Wert des Elements angegeben. Dies ist die Voreinstellung.

- `base64`
  Der Wert des Attributs ist base64-kodiert.
- `stream`
  Der Wert des Attributs ist ein Streaming-Ticket. Das Ticket verweist auf einen Inhalt, den der Client zuvor über das Streaming-Interface dem Search Engine Server übermittelt hat (siehe die unten folgende Erläuterung).

Wenn ein Feldwert base64-kodiert ist oder über das Streaming-Interface zum Search Engine Server übertragen wurde, so muss für den MIME-Typ des Dokuments ein Präprozessor konfiguriert sein, der den Inhalt des Attributs in reinen Text umwandelt und den Wert von `encoding` auf `plain` setzt.

6.2.2 Streaming

Ein Client hat die Möglichkeit, die Werte von Feldern separat an den Search Engine Server zu übertragen, bevor er ihm einen Indizierungsrequest sendet. Dieses Verfahren ist insbesondere bei größeren binären Datenmengen zu empfehlen, weil es performanter ist, als diese Daten base64-kodiert in den Request aufzunehmen.

Ein Client verwendet das so genannte Streaming-Interface, um solche Daten zum Search Engine Server zu übertragen. Das Streaming-Interface wird mit einem POST-Request zum HTTP-Port des Search Engine Servers unter Angabe der URL */stream* angesprochen. Nachdem die Daten übertragen wurden, erhält der Client als Antwort ein Streaming-Ticket, das er im darauf folgenden Indizierungsrequest in der oben beschriebenen Weise angibt, um die Daten zu referenzieren.

Der Content Management Server überträgt die Inhalte von Dateien des Typs *Ressource* über das Streaming-Interface zum Search Engine Server. Auch der Hauptinhalt von Dateien des Typs *Ordner*, *Dokument* und *Layout* wird dem Search Engine Server auf diese Weise übergeben, wenn der Hauptinhalt größer als 8 KB ist. Bis auf Dateien vom Typ *Layout* gilt dies auch für die Template Engine (die Template Engine lässt Layoutdateien nicht indizieren). Der Mindestumfang der Daten, die per Streaming übertragen werden sollen, kann in der Systemkonfiguration des Content Managers und der Template Engine mit dem Eintrag `minStreamingDataLength` konfiguriert werden.

6.2.3 Response

Der Search Engine Server antwortet auf eine Indizierungsanfrage, die erfolgreich bearbeitet werden konnte, mit einem leeren `ses-code`-Element, das die in Abschnitt Response-Element beschriebenen Attribute hat. Hier ein Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ses-payload SYSTEM "http://www.infopark.com/ses.dtd">
<ses-payload payload-id="d--1950259307-000000002-X" timestamp="20101017150013"
 version="2.1">
  <ses-header>
    <ses-sender sender-id="SES-Infopark-DEV-0" name="SES"/>
    <ses-receiver name="CM Server" receiver-id="CM-Infopark-DEV-0"/>
  </ses-header>
  <ses-response response-id="0"
       request-id="d--1949717044-000000002-X" success="true">
    <ses-code phrase="OK" numeric="200"/>
  </ses-response>
</ses-payload>
```

6.3 Suchanfragen

6.3.1 Request

Suchanfragen werden mit dem `ses-search`-Element in einem `ses-request`-Element nach dem folgenden Muster gebildet:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ses-payload SYSTEM "http://www.infopark.com/ses.dtd">
<ses-payload payload-id="B42TE241" timestamp="20100825172100" ses.version="2.0">
  <ses-header>
    <ses-sender sender-id="FX45RTDT" name="CM-Server"/>
    <ses-authentication login="cm-server" password=""/>
  </ses-header>
  <ses-request request-id="BR12TI5X">
    <ses-search>
      <query parser="simple">www &lt;#AND&gt; business</query>
      <minRelevance>50</minRelevance>
      <maxDocs>200</maxDocs>
      <offset>
        <start>61</start>
        <length>20</length>
      </offset>
      <searchBase>
        <query parser="simple">title &lt;#CONTAINS&gt; business</query>
        <collection>collection1</collection>
        <collection>collection2</collection>
      </searchBase>
      <sortOrder>
        <sortField direction="desc">field23</sortField>
        ...
      </sortOrder>
      <resultRecord>
        <resultField format="ISO" timezone="MEZ"
             formatter="formatterAliasName">field1</resultField>
        ...
      </resultRecord>
      <searchDirection start="newest" />
    </ses-search>
  </ses-request>
</ses-payload>
```

Die Unterelemente des `ses-search`-Elements werden im Folgenden beschrieben. Die angegebenen Voreinstellungen, wie beispielsweise `500` für `maxDocs` (die maximale Anzahl Dokumente im Suchergebnis), werden vom Search Engine Server gegebenenfalls ergänzt, bevor die Anfrage an das Suchmaschinenmodul weitergegeben wird. Dies ist nur der Fall, wenn das Verity-Suchmaschinenmodul eingesetzt wird. Der Präprozessor erhält jedoch immer das ursprüngliche, nicht um Voreinstellungen ergänzte Anfragedokument.

- `query`
  Das Element ist optional. Es hat das optionale Attribut `parser`, dessen Wert `simple` (Voreinstellung), `explicit` oder `freetext` sein kann. Ist das Element nicht angegeben, so werden alle indizierten Dokumente der angegebenen Collections geliefert. Der Inhalt des Elements ist die Suchanfrage in der suchprozessorspezifischen Syntax.
- `minRelevance`
  Mit diesem optionalen Element wird die Mindestrelevanz der ins Suchergebnis aufzunehmenden Dokumente angegeben. Erlaubte Werte sind ganze Zahlen zwischen 0 und 100 (jeweils einschließlich), wobei 0 die geringste und 100 größte Relevanz ist. Der voreingestellte Wert ist 0.
- `maxDocs`

Dieses Element ist optional. Sein Inhalt gibt die maximale Anzahl der Dokumente an, die sich in der Trefferliste befinden dürfen. Zulässige Werte sind positive ganze Zahlen und die Zeichenkette `unlimited`. `unlimited` bedeutet, dass so viele Treffer in das Suchergebnis aufgenommen werden, wie die Plattform es zulässt.

- `offset`
  Mit diesem optionalen Element kann man den zu liefernden Ausschnitt des Suchergebnisses angeben. Unterelemente:
  - `start`
    Der Inhalt dieses Elements spezifiziert den Index des ersten zu liefernden Dokuments. Der Index des ersten Dokuments im Suchergebnis ist 1. Voreinstellung: 1.
  - `length`
    spezifiziert die Anzahl der zu liefernden Dokumente, beginnend mit dem durch `start` definierten Index. Voreinstellung: 20. Wird durch den mit `offset` spezifizierten Ausschnitt ein nicht oder teilweise nicht existierender Bereich des Suchergebnisses angefordert, so werden keine bzw. nur die im Bereich existierenden Dokumente geliefert. In diesen Fällen wird kein Fehler erzeugt.
- `searchBase`
  Dieses optionale Element schränkt die zu durchsuchenden Dokumente mit einer Suchanfrage oder durch die Angabe von Collections ein. Wird das Element nicht angegeben, so werden alle indizierten Dokumente aller Collections durchsucht. Unterelemente:
  - `query`
    Das Element ist optional. Es spezifiziert eine Suche, die ausgeführt wird, bevor die im `query`-Element unterhalb von `ses-search` angegebene Suchanfrage ausgeführt wird. Ist das Element nicht angegeben, so werden alle indizierten Dokumente der angegebenen Collections durchsucht. Das Element hat das optionale Attribut `parser`, dessen Wert `simple` (Voreinstellung), `explicit` oder `freetext` sein kann. Der Inhalt des Elements ist die Suchanfrage in der suchprozessorspezifischen Syntax.
  - `collection`
    Das Element ist optional und kann mehrmals angegeben werden. Sein Inhalt ist der Name einer Collection, deren Dokumente durchsucht werden sollen. Ist das Element nicht angegeben, so werden die Dokumente aller Collections durchsucht.
- `sortOrder`
  Mit diesem optionalen Element kann die Sortierung der Dokumente im Suchergebnis beeinflusst werden. Voreingestellt werden die Dokumente nach `score`, also nach Relevanz sortiert. Das Element kann mindestens ein und höchstens 16 `sortField`-Unterlemente haben:
  - `sortField`
    Jedes `sortField`-Element legt den Namen eines Dokumentfeldes fest, das als Suchkriterium verwendet werden soll, wobei die Reihenfolge der Elemente berücksichtigt wird. Das erste `sortField`-Element definiert das primäre Sortierkriterium, das zweite das sekundäre usw. Bei der Sortierung werden nur die ersten 64 Zeichen der Feldwerte berücksichtigt. Alle verfügbaren Felder und `score` können als Sortierkriterium angegeben werden. `sortField` hat das optionale Attribut `direction`, dessen Wert die Sortierreihenfolge festlegt. Erlaubte Werte sind `asc` (aufsteigend, die Voreinstellung) und `desc` (absteigend).
- `resultRecord`
  Dieses Element ist optional. Mit seinen Unterelementen werden die Felder spezifiziert, die zu jedem Dokument im Suchergebnis zurückgegeben werden sollen. Voreingestellt werden die Felder `id`, `title` und `score` geliefert. Unterelemente:
  - `resultField`
    Der Inhalt eines `resultField`-Elements spezifiziert den Namen des je Dokument zurückzugebenden Feldes. Es können mehrere `resultField`-Elemente im Inhalt von

`resultRecord` vorkommen. Alle verfügbaren Felder sowie `id` und `score` können angegeben werden. Wird ein nicht existierendes Feld angegeben, so wird als dessen Wert der leere Wert zurückgegeben. Das Element hat die drei Attribute `format`, `timezone` und `formatter`:
Der Wert von `format` ist der Name eines Datumsformats, mit dem der Wert des Feldes jeweils formatiert wird, wenn es sich dabei um einen Datumswert handelt. Die Formatnamen und ihre Formate sind im Systemkonfigurationseintrag `validDateTimeOutputFormats` abgelegt (siehe Den Search Engine Server ausführen). Voreingestellt wird das erste dort eingetragene Format verwendet.
Mit dem Attribut `timezone` kann die Zeitzone spezifiziert werden, in die die Werte von Datumsangaben umgerechnet werden sollen. Voreingestellt wird die Zeitzone des Rechners verwendet, auf dem der Search Engine Server läuft.
Mit dem Attribut `formatter` können die Werte der zurückgegebenen Dokumentfelder unabhängig von ihrem Typ formatiert werden. Der Wert des Attributs ist der Aliasname einer Tcl-Prozedur, dem im Systemkonfigurationseintrag `tclFormatterCommands` der tatsächliche Name der Tcl-Prozedur zugeordnet wurde (siehe Den Search Engine Server ausführen).

- `searchDirection`
  Dieses optionale Element legt fest, in welcher Reihenfolge die Dokumente in den angegebenen Collections durchsucht werden. Diese Reihenfolge wird mit dem Wert des Attributs `start` spezifiziert. Der Wert kann `newest` (Voreinstellung) oder `oldest` sein. Bei `newest` beginnt die Suche bei den neuesten Dokumenten, andernfalls bei den ältesten. Das Element hat keinen Inhalt.

6.3.2 Response

Die Antwort auf einen Suchanfrage-Request wird im Antwort-Payload im Element `searchResults` unterhalb des `ses-code`-Elements kodiert. Es enthält den angeforderten Ausschnitt aus dem Suchergebnis, der mit dem Element `offset` in der Anfrage spezifiziert wurde. Hier ein Beispiel:

```
<?xml version="1.0"?>
<!DOCTYPE ses-payload SYSTEM "http://www.infopark.com/ses.dtd">
<ses-payload payload-id="d--1950259307-000000004-X" timestamp="20101017150655"
 version="2.1">
  <ses-header>
    <ses-sender sender-id="SES-Infopark-DEV-0" name="SES"/>
    <ses-receiver name="CM Server" receiver-id="CM-Infopark-DEV-0"/>
  </ses-header>
  <ses-response response-id="0"
       request-id="d--1949717044-000000006-X" success="true">
    <ses-code phrase="OK" numeric="200">
      <searchResults hits="567" searched="2045381">
        <record index="21" offsetIndex="1">
          <title>A sample Document</title>
          <score>77</score>
          <docId>546381</docId>
        </record>
        ...
      </searchResults>
    </ses-code>
  </ses-response>
</ses-payload>
```

Das Element `searchResults` hat die beiden Attribute `hits` und `searched`. Der Wert von `hits` gibt an, wieviele Treffer das gesamte Suchergebnis umfasst. Der Wert von `searched` gibt an, wieviele Dokumente insgesamt durchsucht wurden. Der Inhalt von `searchResults` ist eine Liste von `record`-Elementen:

- `record`
  Jeder Treffer ist durch ein `record`-Element repräsentiert. Das Element hat die beiden Attribute `index` und `offsetIndex`. Der Wert von `index` ist der Index des Dokuments im gesamten

Suchergebnis, während der Wert von `offsetIndex` der Index des Dokuments im angeforderten Ausschnitt des Suchergebnisses ist. Der kleinste Index ist jeweils 1.
Der Inhalt des Elements ist eine Liste von Elementen, deren Name jeweils der Name eines Dokumentfeldes ist (siehe Content-Indizierung). Jedem dieser Elemente entspricht ein `resultField`-Element in der Suchanfrage.
Der Inhalt eines Dokumentfeld-Elements ist der Wert des Dokumentfeldes, nachdem dieser formatiert wurde. Die Formatierung wird mit den Attributen `format`, `timezone` und `formatter` im entsprechenden `resultField`-Element im Suchrequest bewirkt.

6.4 Dokument-Löschanfragen

6.4.1 Request

Dokument-Löschanfragen werden mit dem `ses-deleteDoc`-Element in einem `ses-request`-Element nach dem folgenden Muster gebildet:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ses-payload SYSTEM "http://www.infopark.com/ses.dtd">
<ses-payload payload-id="B42TE241" timestamp="20100825172100" version="2.1">
  <ses-header>
    <ses-sender sender-id="FX45RTDT" name="CM-Server"/>
    <ses-authentication login="cm-server" password=""/>
  </ses-header>
  <ses-request request-id="BR12TI5X">
    <ses-deleteDoc docId="4711" collection="collection1"/>
  </ses-request>
</ses-payload>
```

Das `ses-deleteDoc`-Element hat die folgenden Attribute:

- `docId`
  Die ID des zu löschenden Dokuments. Dies ist üblicherweise die Versions-ID (Content Management Server) oder die Datei-ID (Template Engine).
- `collection`
  Der Name der Collection, aus der das Dokument gelöscht werden soll.

6.4.2 Response

Der Search Engine Server antwortet auf eine Löschanfrage mit einem leeren `ses-code`-Element, das die in Abschnitt Response-Element beschriebenen Attribute hat. Hier ein Beispiel:

```
<?xml version="1.0"?>
<!DOCTYPE ses-payload SYSTEM "http://www.infopark.com/ses.dtd">
<ses-payload payload-id="d--1950259307-000000003-X" timestamp="20101017150617"
 version="2.1">
  <ses-header>
    <ses-sender sender-id="SES-Infopark-DEV-0" name="SES"/>
    <ses-receiver name="CM Server" receiver-id="CM-Infopark-DEV-0"/>
  </ses-header>
  <ses-response response-id="0" request-id="d--1949717044-000000004-X" success="true">
    <ses-code phrase="OK" numeric="200"/>
  </ses-response>
</ses-payload>
```

6.5 Collection-Löschanfragen

6.5.1 Request

Sämtliche in einer Collection indizierten Dokumente können mit einer Collection-Löschanfrage gelöscht werden. Eine solche Anfrage entfernt nicht die Collection insgesamt, sondern lediglich ihren Datenbestand. Die Konfiguration der Collection bleibt bei diesem Vorgang erhalten.

Collection-Löschanfragen werden mit dem `ses-purgeCollection`-Element in einem `ses-request`-Element nach dem folgenden Muster gebildet:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ses-payload SYSTEM "http://www.infopark.com/ses.dtd">
<ses-payload payload-id="B42TE241" timestamp="20100825172100" ses.version="2.0">
  <ses-header>
    <ses-sender sender-id="FX45RTDT" name="CM-Server"/>
    <ses-authentication login="cm-server" password=""/>
  </ses-header>
  <ses-request request-id="BR12TI5X">
    <ses-purgeCollection>collection1</ses-purgeCollection>
  </ses-request>
</ses-payload>
```

Das `ses-purgeCollection`-Element hat keine Attribute.

6.5.2 Response

Der Search Engine Server antwortet auf eine Collection-Löschanfrage mit einem leeren `ses-code`-Element, das die in Abschnitt Response-Element beschriebenen Attribute hat. Hier ein Beispiel:

```
<?xml version="1.0"?>
<!DOCTYPE ses-payload SYSTEM "http://www.infopark.com/ses.dtd">
<ses-payload payload-id="d--1950259307-000000003-X" timestamp="20101017150617"
 version="2.1">
  <ses-header>
    <ses-sender sender-id="SES-Infopark-DEV-0" name="SES"/>
    <ses-receiver name="CM Server" receiver-id="CM-Infopark-DEV-0"/>
  </ses-header>
  <ses-response response-id="0" request-id="d--1949717044-000000004-X" success="true">
    <ses-code phrase="OK" numeric="200"/>
  </ses-response>
</ses-payload>
```

6.6 Fehlerbehandlung

In diesem Abschnitt werden die Fehler aufgeführt, die innerhalb der `ses-code`-Elemente in Antwort-Payloads auftreten können. Hier werden nur sogenannte Protokollfehler genannt und nicht solche Fehler, die auftreten können, wenn der Search Engine Server eine Operation ausführt.

Bei den Protokollfehlern wird zwischen Fehlern auf Payload-Ebene und auf Request-Ebene unterschieden.

6.6.1 Payload-Fehler

Payload-Fehler werden erzeugt, wenn ein Anfrage-Payload fehlerhaft aufgebaut ist. Tritt ein solcher Fehler auf, so werden sämtliche im Payload enthaltenen Requests ignoriert. Der Search Engine Server gibt im Fehlerfall ein Antwort-Payload zurück, das ein einziges `ses-response`-Element enthält. Das öffnende Tag dieses Elements enthält statt des `request-id`-Attributs ein `payload-id`-Attribut, dessen Wert die ID des fehlerhaften Payloads ist.

Das `ses-response`-Element enthält ein `ses-code`-Element, in dessen öffnendem Tag die Attribute `numeric` und `phrase` auf die Fehlernummer bzw. den Meldungstext gesetzt sind. Das folgende Beispiel zeigt ein Antwort-Payload mit einer Fehlermeldung.

```
<?xml version="1.0" encoding="UTF-8">
<!DOCTYPE ses-payload SYSTEM "http://www.infopark.com/ses.dtd">
<ses-payload payload-id="B3BWPOIU" timestamp="20100906100205" version="2.1">
  <ses-header>
    <ses-sender sender-id="U2JWUE09" name="SES"/>
  </ses-header>
  <ses-response response-id="BR12TI5X"
       payload-id="AHZ97I28" success="false">
    <ses-code numeric="1"
        phrase="Payload incomplete / cannot parse">
    </ses-code>
  </ses-response>
</ses-payload>
```

Auf Payload-Ebene können die folgenden Fehler auftreten:

- **Payload incomplete / Cannot Parse**
  Anfrage-Payloads, die ungültigen XML-Code enthalten, werden mit dieser Meldung beantwortet.
- **Not well-formed Payload**
  Anfrage-Payloads, die gültigen XML-Code enthalten, jedoch kein gültiges Anfrage-Payload darstellen, werden mit dieser Meldung beantwortet.
- **Incompatible Version**
  Mit diesem Fehler werden Anfrage-Payloads beantwortet, die in einer Version des XML-Schnittstellenprotokolls formuliert sind, die der Server nicht unterstützt. Das Verhalten des Servers in diesen Fällen ist im Abschnitt Payloads beschrieben.
- **Authentication Failed**
  Die Informationen im `ses-authentication`-Element sind ungültig.

6.6.2 Request-Fehler

Request-Fehler betreffen nur einen einzelnen Request. Jeder Request-Fehler wird in der entsprechenden Response im Antwort-Payload zurückgegeben.

Auf Request-Ebene können die folgenden Fehler auftreten:

- **Not well-formed request**
  Das in dem `ses-request`-Element enthaltene XML-Fragment stellt keinen gültigen Request dar.
- **Execution precluded**
  Der Request wurde nicht bearbeitet, weil ein vorhergehender Request, der als `preclusive` markiert war, fehlschlagen ist.

7

7 MISE als DTD

In diesem Abschnitt ist die Definition von MISE als DTD wiedergegeben. Änderungen sind vorbehalten.

7.1 Request

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT ses-request
  (ses-indexDoc|ses-deleteDoc|ses-search|ses-purgeCollection|
  ses-optimizeCollections|ses-flushQueue|ses-holdQueue|
  ses-resumeQueue)>
<!ATTLIST ses-request
  request-id CDATA #REQUIRED
  preclusive (true | false) "false"
>
<!ELEMENT ses-indexDoc ANY>
<!ATTLIST ses-indexDoc
  docId CDATA #REQUIRED
  collection CDATA #REQUIRED
  usesStreaming ("YES" | "NO") #DEFAULT "NO"
  mimeType CDATA #REQUIRED
>
<!-- The following element is an example for an attribute element
     contained in the ses-indexDoc element-->
<!ELEMENT blob (#PCDATA)>
<!ATTLIST blob
  encoding (base64|plain|stream) #DEFAULT plain
>
<!ELEMENT ses-deleteDoc EMPTY>
<!ATTLIST ses-deleteDoc
   docId CDATA #REQUIRED
   collection CDATA #REQUIRED
>
<!ELEMENT ses-search
  (query?,
  minRelevance?,
  maxDocs?,
  offset?,
  searchBase?,
  sortOrder?,
  resultRecord?,
  searchDirection?) >
<!ELEMENT query (#PCDATA)
<!ATTLIST parser (simple|explicit|freetext) #DEFAULT simple>
<!ELEMENT minRelevance (#PCDATA)>
<!ELEMENT maxDocs (#PCDATA)>
<!ELEMENT offset (start,length)>
<!ELEMENT start (#PCDATA)>
<!ELEMENT length (#PCDATA)>
<!ELEMENT searchBase (collection+,query?>
<!ELEMENT collection (#PCDATA)>
<!ELEMENT sortOrder (sortField+)>
<!ELEMENT sortField (#PCDATA)>
```

```
<!ATTLIST sortField direction (asc,desc) #DEFAULT asc>
<!ELEMENT resultRecord (resultField+)>
<!ELEMENT resultField (#PCDATA)>
<!ATTLIST resultField
  format (#CDATA) #IMPLIED
  timezone (#CDATA) #IMPLIED
  formatter (#CDATA) #IMPLIED
>
<!ELEMENT searchDirection EMPTY>
<!ATTLIST searchDirection start (newest|oldest) #DEFAULT newest
>
<!ELEMENT ses-optimizeCollections EMPTY>
<!ELEMENT ses-purgeCollection (#PCDATA)>
<!ELEMENT ses-flushQueue EMPTY>
<!ELEMENT ses-holdQueue EMPTY>
<!ELEMENT ses-resumeQueue EMPTY>
```

7.2 Response (ses-search)

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT ses-response (ses-code*)>
<!ATTLIST ses-response
  response-id CDATA #REQUIRED
  payload-id CDATA #IMPLIED
  request-id CDATA #IMPLIED
  successful (true | false) #REQUIRED
>
<!ELEMENT ses-code ANY>
<!ATTLIST ses-code
  numeric CDATA #REQUIRED
  phrase CDATA #REQUIRED
>
<!ELEMENT searchResults (record*)>
<!ATTLIST searchResults
  hits CDATA #REQUIRED
  searched CDATA #REQUIRED
>
<!ELEMENT record ANY>
<!ATTLIST record
  index CDATA #REQUIRED
  offsetIndex CDATA #REQUIRED
>
<!ELEMENT title ANY>
<!ATTLIST field
  type CDATA #REQUIRED
>
```