docuvita.AutoProfiler

Administrationshandbuch

2016
docuvita GmbH & Co. KG
2.0.632.6039

docuvita
Dokumentenmanagement

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Bevor Sie beginnen…

1 Bevor Sie beginnen…

In diesem Kapitel erfahren Sie mehr über die Einsatzmöglichkeiten des docuvita.AutoProfilers und die zu erfüllenden Systemvoraussetzungen.

1.1 Über dieses Handbuch

Dieses Handbuch richtet sich an Systemadministratoren und IT-Verantwortliche, die Ihr docuvita Dokumentenmanagement-System um automatisierte Importstrecken erweitern wollen. Es erläutert die Schritte, die für eine Konfiguration des docuvita.AutoProfilers erforderlich sind.

Beachten Sie, dass für die Aufbewahrung bestimmter Unterlagen in elektronischer Form verschiedene rechtliche Anforderungen erfüllt werden müssen. Bitte informieren Sie sich bei Ihrem docuvita Partner, welche zusätzlichen Maßnahmen ergriffen werden müssen, um alle rechtlichen Anforderungen zu erfüllen.

1.2 Symbolerklärung

An verschiedenen Stellen dieses Handbuchs finden sich Hinweis-Boxen, die Sie auf besonders wichtige oder wissenswerte Informationen aufmerksam machen sollen.

Achtung!

Diese Hinweise sollten Sie unbedingt beachten.

Wissenswerte Information

Hinweise dieser Kategorie enthalten hilfreiche Tipps.

1.3 Einsatzmöglichkeiten

Mit Hilfe des docuvita.AutoProfilers ist es möglich, Dokumente oder Objekte in das Dokumentenmanagement-System docuvita zu importieren, sie zu aktualisieren und in einer Ablagestruktur neu zuzuordnen. Dabei werden zu aktualisierende Dokumente je nach Einstellung automatisch um eine neue Dokumentversion erweitert.

Mögliche Einsatzzwecke sind zum Beispiel:

- Import von Bestandsdokumenten
- Periodische Übernahme von Stammdaten aus Fremdsystemen
- Import von in Fremdsystemen generierten Dokumenten (z.B. Archivierung von Belegen aus einem Warenwirtschaftssystem)
  - mit Hilfe des Dateinamens
  - mit Hilfe einer XML-Steuerdaten
  - mit Hilfe eines PDF-Archivdruckers
  - mit Hilfe von Steuerzeichen in PDF-Dokumenten
  - mit Hilfe sonstiger Dokumentbereitstellungsfunktionen
- Import von gescannten Dokumenten
  - mit Hilfe von Barcodes
- Import von E-Mails (E-Mailarchivierung)
  - aus einem Ordner im Dateisystem
  - durch Abholung mit Hilfe der Standardprotokolle POP3(/S) oder IMAP(/S)

Um automatisiert Importe durchführen zu können, ist der docuvita.Server auf das Vorhandensein einer oder mehrerer Steuerdatei(en) angewiesen. Diese können entweder zusammen mit den zu archivierenden Dokumente direkt von einem Fremdsystem erzeugt werden (siehe Kapitel Aufbau der dvImport-Datei) oder sie werden vom AutoProfiler selbst erzeugt. Dazu ist der docuvita.AutoProfiler in der Lage, Metadaten aus dem Inhalt von Dokumenten zu extrahieren, diese zusätzlich über externe Datenbanken anzureichern und für die Verschlagwortung und automatische Archivierung der Dokumente zu verwenden (siehe AutoProfiler).

Der docuvita.AutoProfiler arbeitet als Serverdienst in Intervallen definierte Verzeichnisse im Dateisystem oder über Netzwerkfreigaben sowie Postfächer ab und verarbeitet die dort vorgefundenen Dokumente und Steuerdateien automatisch.

1.4 Systemvoraussetzungen

Für den Betrieb des docuvita.AutoProfilers müssen folgende Voraussetzungen erfüllt sein:

- **Installiertes Microsoft .NET-Framework.**

  Bei docuvita handelt es sich um eine für das Microsoft .NET Framework 4.0 entwickelte Software. Das bedeutet, dass sowohl die docuvita.Server-Komponenten, als auch der docuvita.WindowsClient nur auf einem PC oder Server installiert werden können, wenn zuvor das Microsoft .NET Framework installiert wurde.

	Ältere Versionen des .NET Framework

Die docuvita Software benötigt das .NET Framework in der Version 4.0 oder 4.5. Eine Parallelinstallation mit älteren Versionen auf einem Rechner ist problemlos möglich.

Das Microsoft .NET Framework, Version 4.0, ist lizenzkostenfrei verfügbar und
bereits auf den meisten aktuellen Computern vorinstalliert. Ist dieses nicht der Fall wird es bei der Installation von docuvita automatisch heruntergeladen und installiert.

- **Installierter und konfigurierter docuvita Server**

  Die ServiceUrl des docuvita.Servers muss bekannt ein (siehe Basiskonfiguration).

- **docuvita-Import-Benutzer, in dessen Kontext der Import durchgeführt wird**

  Weitere Informationen zum Anlegen eines solchen Kontos finden im Kapitel Einrichtung Importbenutzer.

- **Windows-Dienstkonto für den AutoProfiler**

  Da der docuvita.AutoProfiler als Windows-Dienst installiert wird, benötigen Sie ein Windows-Benutzerkonto, unter dem dieser ausgeführt werden soll. Dieses Konto muss über die Berechtigung *als Dienst anmelden* verfügen und vollen Zugriff auf die zu überwachenden sowie die temporären Verarbeitungsverzeichnisse haben. Wollen Sie Verzeichnisse in einem Netzwerk überwachen empfiehlt sich die Nutzung eines Domänen-Accounts um den Zugriff sicherzustellen.

Groß-/Kleinschreibung

Bitte beachten Sie, dass bei der Verwendung von XML-Tags in den Konfigurations- und Importdateien die Groß-/Kleinschreibung relevant ist.

Ein Nichtbeachten der Schreibweise kann zu Fehlermeldungen führen.

XML-Konformität

Bei der Ausführung des AutoProfilers und beim Import in docuvita wird keine XML-Validierung (Konformitätsprüfung auf Einhaltung der XML-Standards) durchgeführt. Stößt die Verarbeitung auf einen XML-Fehler, führt dieser zum Abbruch des Importvorgangs. Führen Sie daher bei

automatisch erzeugten Importdateien Probeläufe aus. Bei manuellen/ einmaligen Datenimporten können Sie mit XML-Hilfsprogrammen vor dem Import die Gültigkeit der Importsteuerdatei prüfen.

1.5 FAQs

Bitte beachten Sie auch die aktuellen FAQs in unserem Partnerportal. Das Partnerportal ist nur für docuvita Fachhändler zugängig. Hier werden ggf. noch nicht in dieser Anleitung behandelte Themen und neue Funktionen stets aktuell beschrieben. Außerdem finden Sie dort weitere Beispiele aus der Praxis.

Installation & Konfiguration

2 Installation & Konfiguration

In diesem Abschnitt erhalten Sie Informationen darüber, wie Sie den docuvita.AutoProfiler konfigurieren und an die Bedürfnisse des jeweiligen Einsatzzweckes anpassen können.

2.1 Installation

Eine separate Installation ist nicht notwendig, der docuvita.AutoProfiler ist Bestandteil des docuvita.Server Installationspakets und wird automatisch bei der Installation eines docuvita.Servers installiert. Weitere Informationen zur Installation von docuvita finden Sie im Administrationshandbuch - docuvita.Server - Installation & Konfiguration.

Zur Verwendung des docuvita.AutoProfilers müssen einige Konfigurationsschritte durchgeführt werden. Diese werden in dem folgenden Abschnitt detailliert beschrieben.

docuvita.AutoProfiler auf separatem Server betreiben

Wenn Sie den AutoProfiler auf einem separaten Server zu Ihrer eigentlichen docuvita Installation installieren wollen (z.B. aus Lastgründen) nutzen Sie bitte die vollständige docuvita.Server Installation und konfigurieren im Anschluss ausschließlich den docuvita AutoProfiler.

2.2 Konfiguration des AutoProfiler-Dienstes

Melden Sie sich zur Konfiguration des docuvita.AutoProfiler-Dienstes am docuvita.Admin mit dem Benutzer *Administrator* und dem von Ihnen direkt nach der Installation vergebenen Kennwort an. Bei einer Standardinstallation erreichen Sie das docuvita.Admin Webinterface über https://localhost/admin/services/web. Das Standardkennwort für die erste Anmeldung nach der Installation lautet *docuvita*.

Nach der Anmeldung sollte die Ansicht nun wie in Abb. 2-1 aussehen (andere Serverkomponenten sind in dem Screenshot bereits fertig eingerichtet).

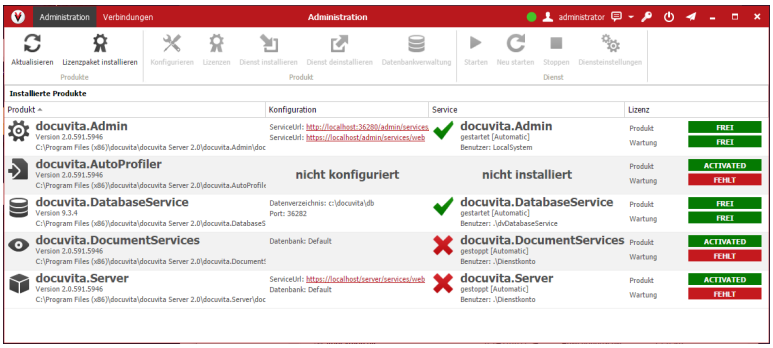


Abb. 2-1: docuvita.Admin

2.2.1 Basiskonfiguration

Um mit der Einrichtung des docuvita.AutoProfilers zu beginnen wählen Sie den *docuvita.AutoProfiler* aus und klicken anschließend im Ribbon auf *Konfigurieren*.

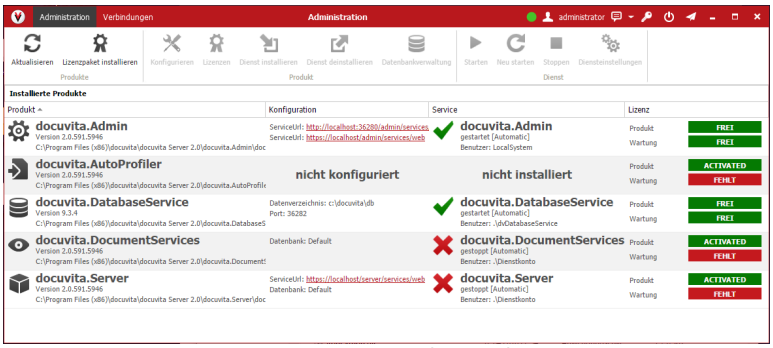


Abb. 2-2: docuvita.AutoProfiler konfigurieren

In der sich öffnenden Maske tragen Sie die ServiceUrl des docuvita.Servers in das Feld AutoProfilerServerUrl ein. Der letzte Abschnitt der URL */web* entfällt, da dieser nur für den Aufruf des docuvita.WebClients notwendig ist.

Abb. 2-3: Produkteinstellungen

Legen Sie außerdem das Intervall (in Sekunden) für die Durchläufe des AutoProfilers fest. In diesem Intervall überprüft der docuvita.AutoProfiler die zu überwachenden Verzeichnisse und Postfächer.

Ihre Konfiguration sollte nun folgendermaßen aussehen:

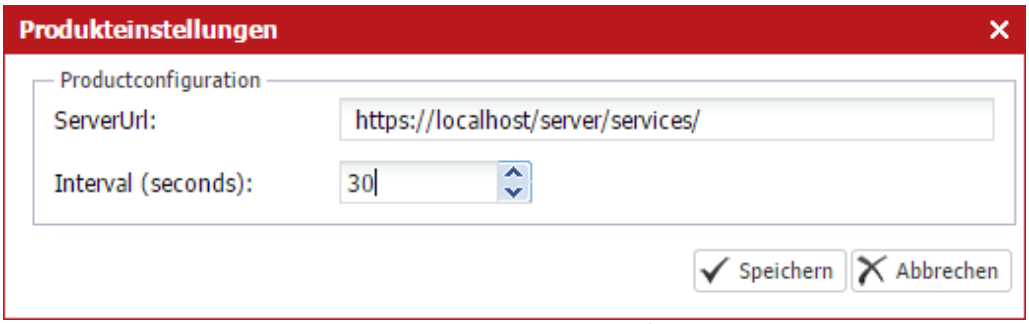


Abb. 2-4: Produkteinstellungen - konfiguriert

Ist der AutoProfiler nicht auf dem selber Server installiert wieder docuvita.Server mus in der *ServerURL* der Servername oder die IP-Adresse anstatt *localhost* verwendet werden.

2.2.2 Dienst einrichten

Um den *docuvita.AutoProfiler* als Dienst einzurichten wählen Sie *docuvita.AutoProfiler* aus und klicken anschließend auf *Dienst installieren*.

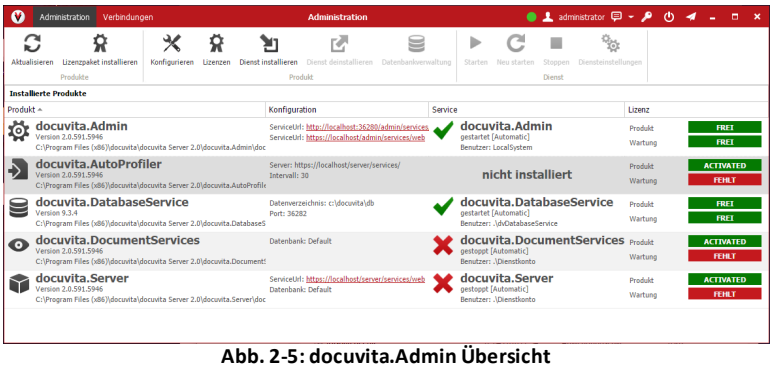


Abb. 2-5: docuvita.Admin Übersicht

Der AutoProfiler wird als Windows-Dienst mit den von Ihnen angegebenen Benutzerdaten eingerichtet. Bitte stellen Sie sicher, dass der verwendete Benutzer volle Zugriffsrechte auf die in im Verlauf dieser Anleitung einzurichtenden Verarbeitungsverzeichnissen bekommt.

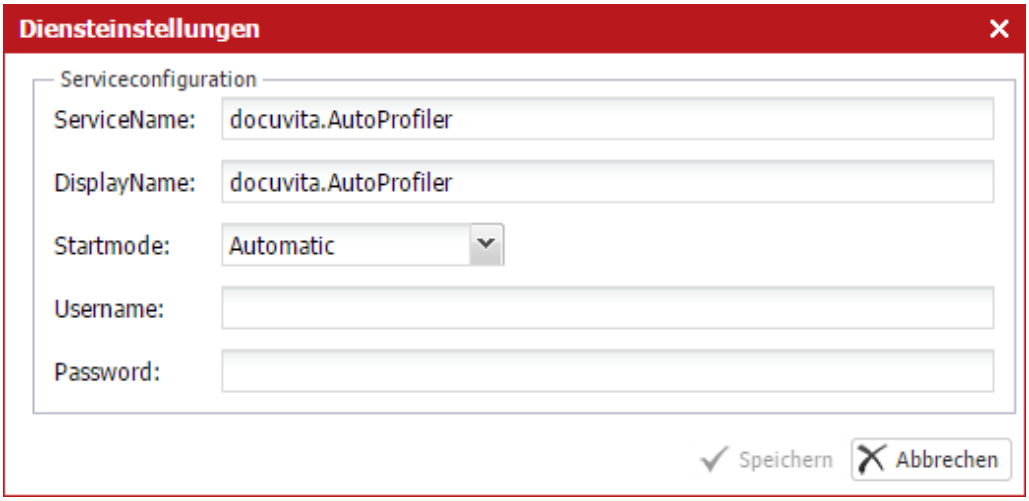


Abb. 2-6: Dienst installieren

- Der Eintrag für einen lokalen Benutzer muss in der Syntax *Benutzername* oder *.\Benutzername* erfolgen.
- Der Eintrag für einen Domänenbenutzer muss in der Syntax *Domäne* *\Benutzername* erfolgen.

Verwenden Sie am besten einen Domänen-Account, wenn der AutoProfiler auf freigegebene Verzeichnisse zugreifen soll.

Weitere Informationen zur Konfiguration finden Sie im Administrationshandbuch - docuvita.Server - Installation & Konfiguration.

Dienst wird automatisch gestartet

Die Startart des Importdienstes ist in der Voreinstellung auf *Automatisch* gesetzt. Sobald Sie den Dienst fertig eingerichtet und gestartet haben beginnt dieser mit seiner Arbeit. Auch nach einem Neustart es Servers nimmt der Dienst seine Arbeit umgehend wieder auf.

Bitte beachten Sie diesen Umstand, falls Sie über das AutoProfiler Konfigurationsprogramm bereits vor der Einrichtung des Dienstes eine Konfiguration hinterlegt haben.

2.2.3 Konfiguration abgeschlossen

Nach erfolgreicher Einrichtung des Dienstes sollten Sie die Konfigurationsparameter der Startseite des docuvita.Admin sehen können.

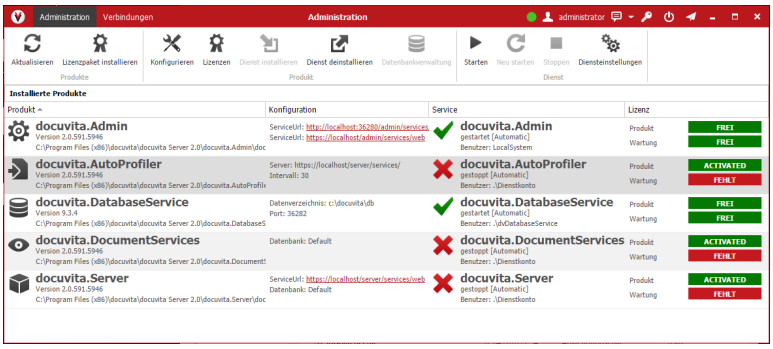


Abb. 2-8: Installation abgeschlossen

2.2.4 Lizenzierung

Damit der docuvita.AutoProfiler Dienst gestartet werden kann muss eine gültige Lizenz eingespielt werden. Wie Sie dieses tun erfahren Sie im Administrationshandbuch - docuvita.Server - Installation & Konfiguration.

Deatillierte Informationen zur Lizenzierung von docuvita finden Sie im Fachhaendlerhandbuch - Partnerportal und Lizenzierung.

2.3 Einrichtung Importbenutzer

Ein Importbenutzer kann durch einen berechtigten Benutzer angelegt werden. Der Benutzer, der dieses tut muss Mitglied der *Importadmingroup* sein. Für weitere Informationen zur *Access configuration* siehe Administrationshandbuch - docuvita.Server - Customizing.

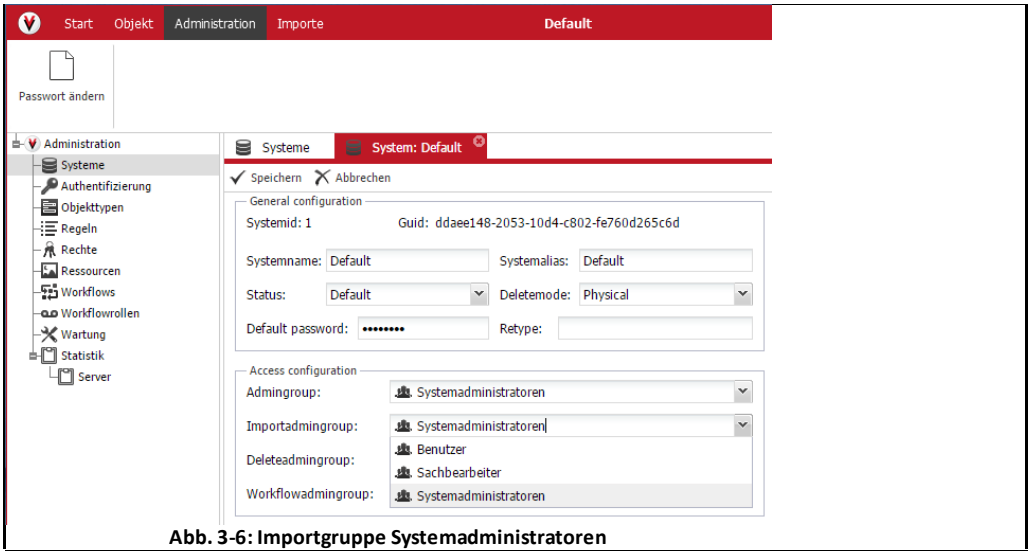


Abb. 3-6: Importgruppe Systemadministratoren

Dieser Benutzer kann in docuvita den Ribbon *Import* sehen und somit die Importbenutzer verwalten. Um einen neuen Importbenutzer anzulegen melden Sie sich an docuvita an und wechseln in den *Import*-Bereich.

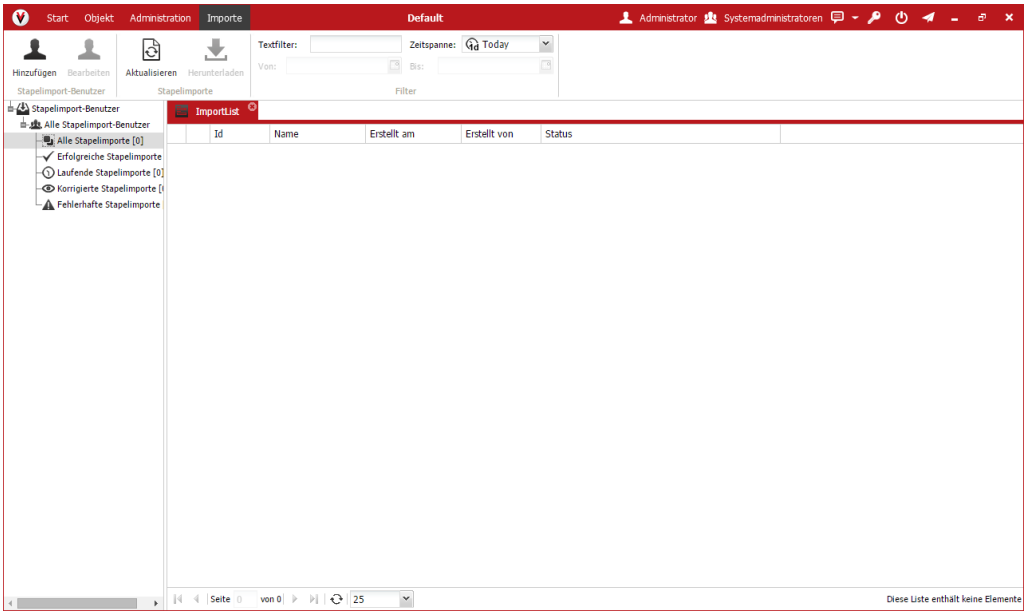


Abb. 3-7: Import-Bereich

Klicken Sie dort auf *Hinzufügen* und füllen die sich öffnende Maske aus. Das

Feld Email muss nur gefüllt werden, wenn eine Benachrichtigung bei Import-Fehlern erfolgen soll (siehe Benutzerhandbuch - docuvita.Client). Für die Funktion muss ein zentrales E-Mail Konto im docuvita System konfiguriert sein (siehe Administrationshandbuch - docuvita.Server - Customizing).

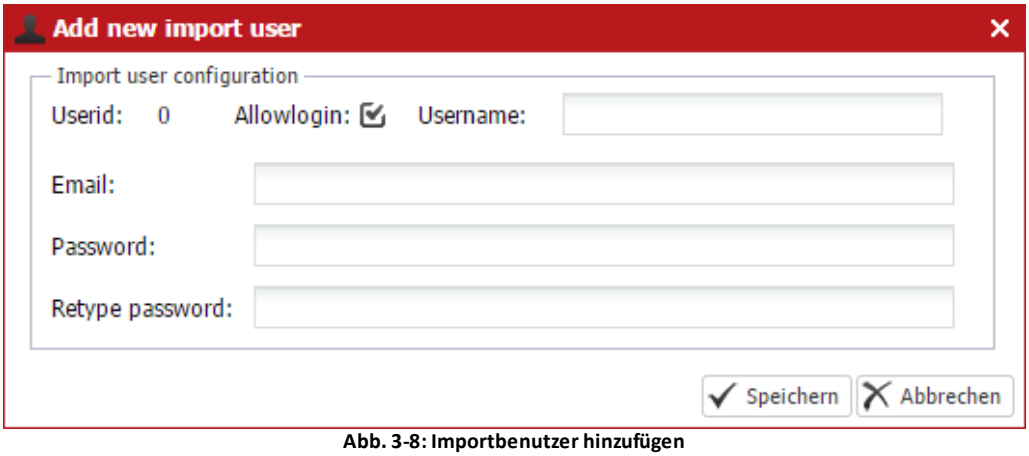


Abb. 3-8: Importbenutzer hinzufügen

docuvita.AutoProfiler
Konfigurationsprogramm

3 docuvita.AutoProfiler Konfigurationsprogramm

Über das docuvita.AutoProfiler Konfigurationsprogramm können sowohl Importkonfigurationen eingerichtet, als auch verschiedenste Tests zur Überprüfung der Konfiguration durchgeführt werden.

3.1 Grundeinstellungen

Starten Sie das docuvita.AutoProfiler Konfigurationsprogramm über das *Startmenü → Alle Programme → docuvita → docuvita AutoProfiler Configurator*.

Mehrere Instanzen des AutoProfilers

Bei mehreren Instanzen des docuvita.AutoProfilers starten Sie das Konfigurationsprogramm bitte direkt über das Programmverzeichnis der jeweiligen AutoProfiler-Instanz. Nur so ist gewährleistet, dass die jeweils gewünschte Konfiguration bearbeitet wird.

Die Anzeige der Seite *Grundeinstellungen* sollte mit Konfiguration im docuvita.Admin automatisch übereinstimmen (siehe Basiskonfiguration). Änderungen, die hier durchgeführt und gespeichert werden, werden automatisch in die Konfiguration des docuvita.Admins übernommen.

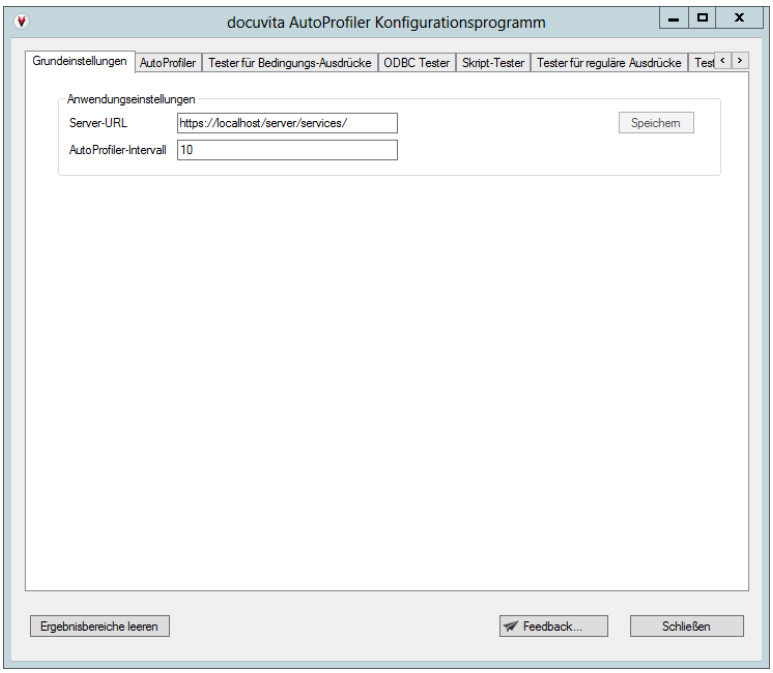


Abb. 3-1: Konfiguration – Grundeinstellungen

3.1.1 Server-URL

Legt die Verbindung zum docuvita.Server fest. Tragen Sie hier die ServiceUrl des docuvita.Servers ein. Der letzte Abschnitt der URL */web* entfällt, da es nur für den Aufruf des docuvita Webinterfaces notwendig ist (siehe auch Basiskonfiguration).

3.1.2 AutoProfiler-Intervall

Legt fest, in welchem Intervall die Verzeichnisse und Postfächer der eingerichteten AutoProfiler auf neue Dateien untersucht werden. Der hier angegebene Wert legt die Dauer in Sekunden fest (siehe auch Basiskonfiguration).

3.2 AutoProfiler-Konfiguration

Im Feld *Konfigurationsordner* können Sie überprüfen für welche Instanz des docuvita.AutoProfilers Sie das Konfigurationsprogramm gestartet haben.

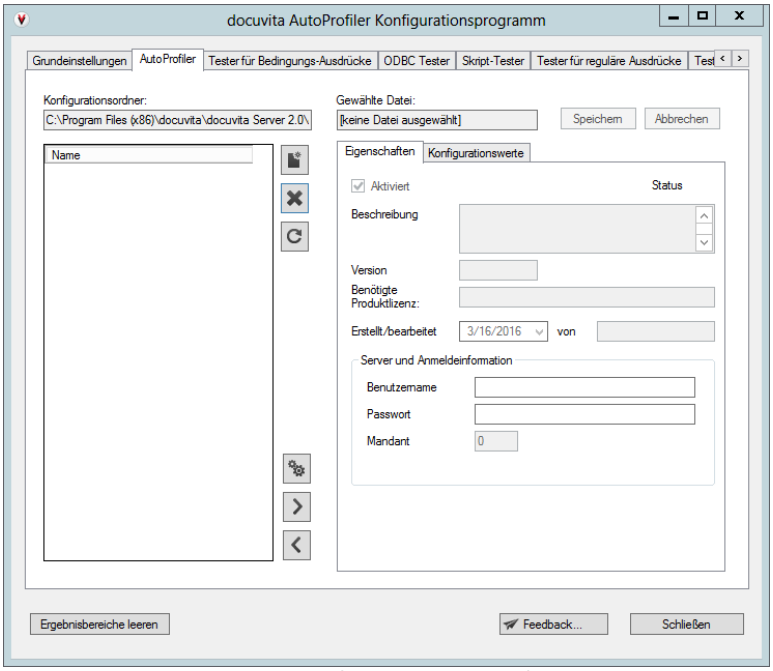


Abb. 3-2: Konfiguration – AutoProfiler

Über den Button Speichern können Sie Konfigurationsänderungen jederzeit speichern. Ein Sternchen (*) in der Titelzeile des Konfigurationsprogramms zeigt an, dass es nicht gespeicherte Änderungen gibt.

Über den Button Abbrechen können Sie getätigte Änderungen seit der letzten Speicherung verwerfen.

Beim Schließen des Programms gibt es eine Abfrage ob bisher nicht gespeicherte Änderungen nun gespeichert oder verworfen werden sollen.

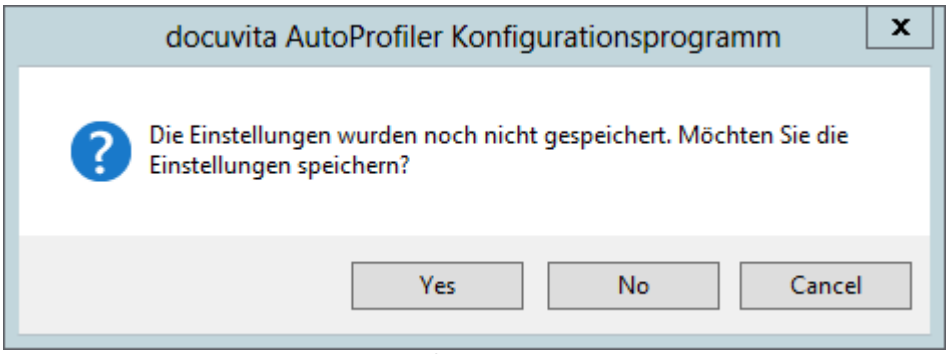


Abb. 3-3: Konfiguration abspeichern

3.2.1 Anlegen einer neuen Konfigurationsdatei

Durch Klick auf das Symbol  können Sie eine neue Konfigurationsdatei anlegen. Nach Klick auf das Symbol öffnet sich ein Speichern-Dialog. Geben Sie hier bitte den gewünschten Dateinamen für die neu Konfigurationsdatei an.

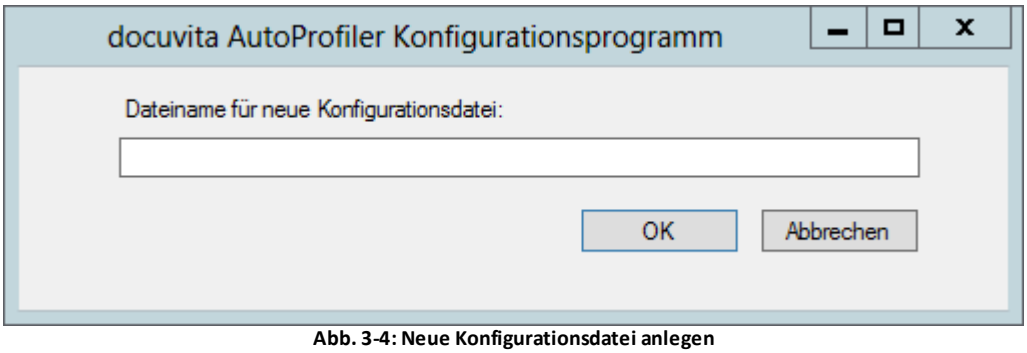


Abb. 3-4: Neue Konfigurationsdatei anlegen

3.2.2 Löschen der ausgewählten Konfiguration

Durch Klick auf den Button  wird die ausgewählte Konfiguration gelöscht. Vor dem Löschen erfolgt eine zusätzliche Sicherheitsabfrage.

3.2.3 Aktualisieren

Ließt die Liste der verfügbaren Konfigurationsdateien neu ein  . Dieses ist nur notwendig wenn Sie Dateien im Verzeichnis manuell ausgetauscht oder hinzugefügt haben.

3.2.4 Import / Export der ausgewählten Konfiguration

Über den Button  können bestehende Importkonfigurationen von docuvita 1.x importiert und in das neue Format umgewandelt werden. Außerdem

können bestehende Konfigurationen im JSON-Format importiert sowie die aktuelle Konfiguration im JSON-Format exportiert werden.

Import von docuvita 1.x Konfigurationsdateien

Achten Sie bitte auf ggf. geändertes Verhalten des AutoProfilers.

Wichtigste Änderung: Die Schlüsselwerte der Objekttypen müssen nun zentral innerhalb des docuvita Systems definiert werden. Testen Sie diese Konfigurationen vor Inbetriebnahme eines produktiven Systems ausführlich.

3.2.5 Export der ausgewählten Konfiguration

Über den Button  können Sie die ausgewählte Konfiguration im .cfg Format herunterladen und lokal speichern.

3.2.6 Import einer Konfiguration

Über den Button  können Sie eine lokal gespeicherte Konfiguration (.cfg Format) hochladen und verwenden / weiter bearbeiten.

3.3 AutoProfiler verwalten

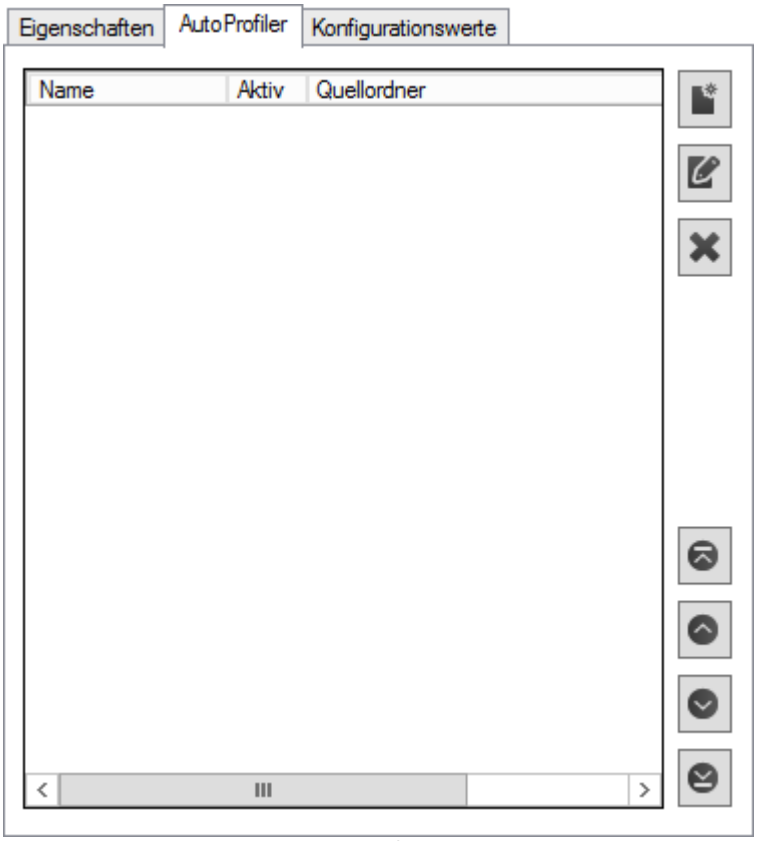


Abb. 3-8: AutoProfiler verwalten


Neuen AutoProfiler anlegen


Ausgewählten AutoProfiler bearbeiten


Ausgewählten AutoProfiler löschen (keine Sicherheitsabfrage)





Ausgewählten AutoProfiler in der Verarbeitungsreihenfolge nach (ganz) oben/unten verschieben

Sequentielle Verarbeitungsreihenfolge

Die AutoProfiler werden sequentiell in der angegebenen Reihenfolge abgearbeitet. Sie können die Sortierung dafür nutzen um aufeinander aufbauende Szenarien zu realisieren. So kann z.B. der letzte AutoProfiler alle Dokumente verarbeiten, die bis zu diesem Zeitpunkt

	von keinem anderen AutoProfiler verarbeitet wurden.

3.4 Eigenschaften der Konfiguration

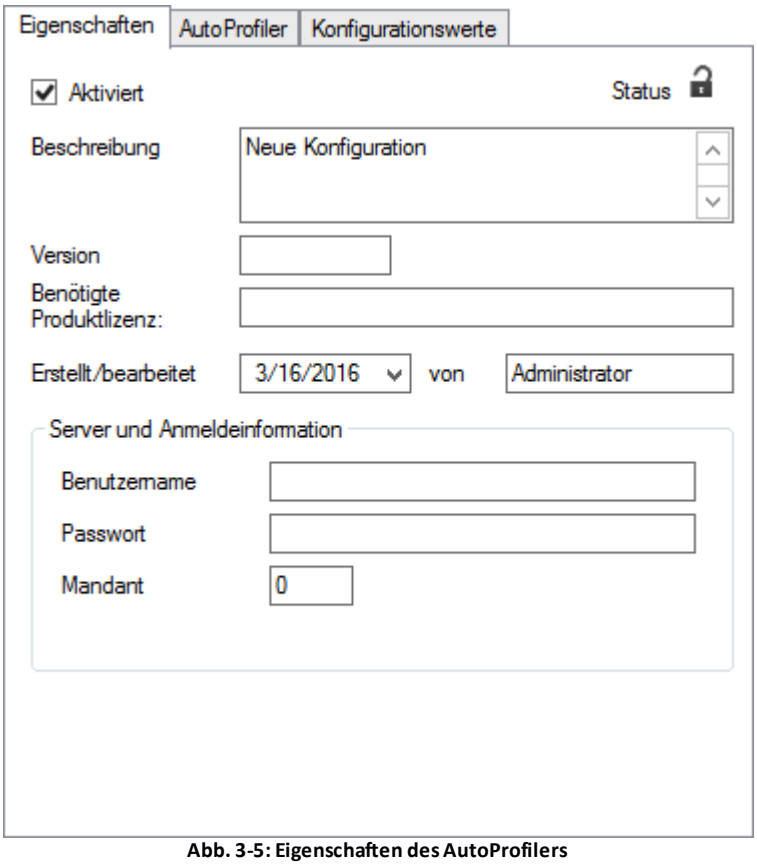


Abb. 3-5: Eigenschaften des AutoProfilers

3.4.1 Aktiviert

Mit der Checkbox *Aktiviert* legen Sie fest ob die aktuell auf der linken Seite ausgewählte Konfiguration vom docuvita.AutoProfiler Dienst berücksichtigt werden soll. Ist Sie nicht aktiviert, so wird diese Konfiguration nicht verwendet und die Dokumente werden nicht aus dem Quellverzeichnis abgeholt.

3.4.2 Beschreibung

Hier kann eine kurze Beschreibung der Konfiguration hinterlegt werden. Dieses Feld ist rein informativ und hat keinen Einfluss auf die Funktion der Konfiguration.

3.4.3 Version

Hier kann eine Versionsnummer der Importkonfiguration gepflegt werden. Dieses Feld ist rein informativ und hat keinen Einfluss auf die Funktion der Konfiguration.

3.4.4 Benötigte Produktlizenz

Diese Option ist derzeit noch ohne Funktion.

3.4.5 Erstellt/bearbeitet

Hier wird automatisch beim ersten Speichern das aktuelle Datum hinterlegt. Später kann das Feld manuell bearbeitet werden. Dieses Feld hat eine rein informative Funktion und hat keinen Einfluss auf die Funktion der Konfiguration.

3.4.6 Von

Hier wird automatisch beim ersten Speichern der aktuell an Windows angemeldete Benutzer als Bearbeiter hinterlegt. Später kann das Feld manuell bearbeitet werden. Dieses Feld ist rein informativ und hat keinen Einfluss auf die Funktion der Konfiguration.

3.4.7 Server und Anmeldeinformationen

Hier muss der docuvita Import-Benutzername sowie sein Passwort für die Anmeldung angegeben werden (siehe Einrichtung Importadmin).

Bei einem erfolgreichen Import wird als Benutzername für das Anlegen SYSTEM angezeigt und als userid wird die ID der ausgewählten *Importadmingroup* hinterlegt. Wird eine andere *Importadmingroup* als *Systemadministratoren* ausgewählt, so ist der Benutzername dennoch SYSTEM, lediglich die ID ändert sich. Für den Import kann auch ein konkreter Benutzer definiert werden (siehe Felder für sämtliche Objekttypen, Bereich: *obj.usercreated*).

3.4.8 Mandant

Legen Sie den docuvita Mandanten fest für den diese Konfiguration gültig ist. Dafür tragen Sie die Systemid Ihres Mandanten ein. Diesen finden Sie im docuvita.Client im Bereich *Administration* → *Systems* → „Ihr System“.

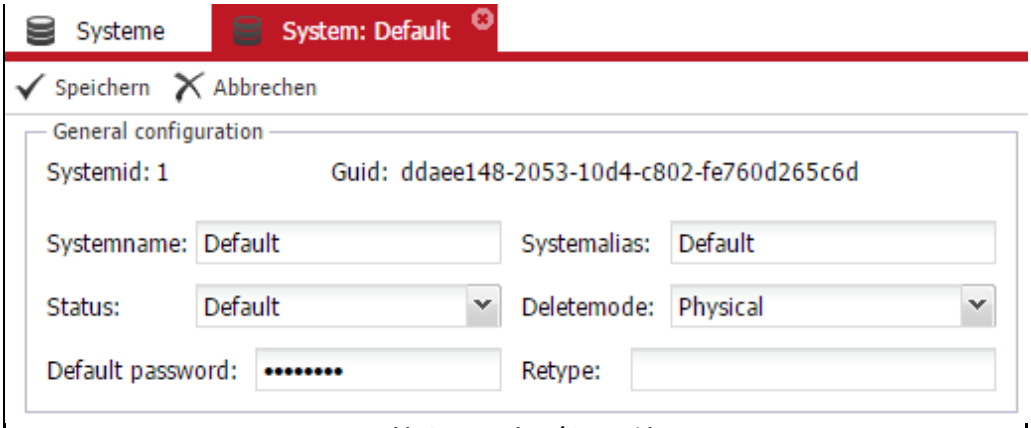


Abb. 3-7: Mandant / Systemid

Verwenden Sie mehrere Mandanten, so muss es pro Mandant eine Konfigurationsdatei geben (siehe Anlegen einer neuen Konfigurationsdatei). Für diesen Zweck kann die Konfigurationsdatei einfach kopiert und entsprechend angepasst werden. Siehe hierzu auch Import / Export der ausgewählten Konfiguratio.

3.5 Konfigurationswerte

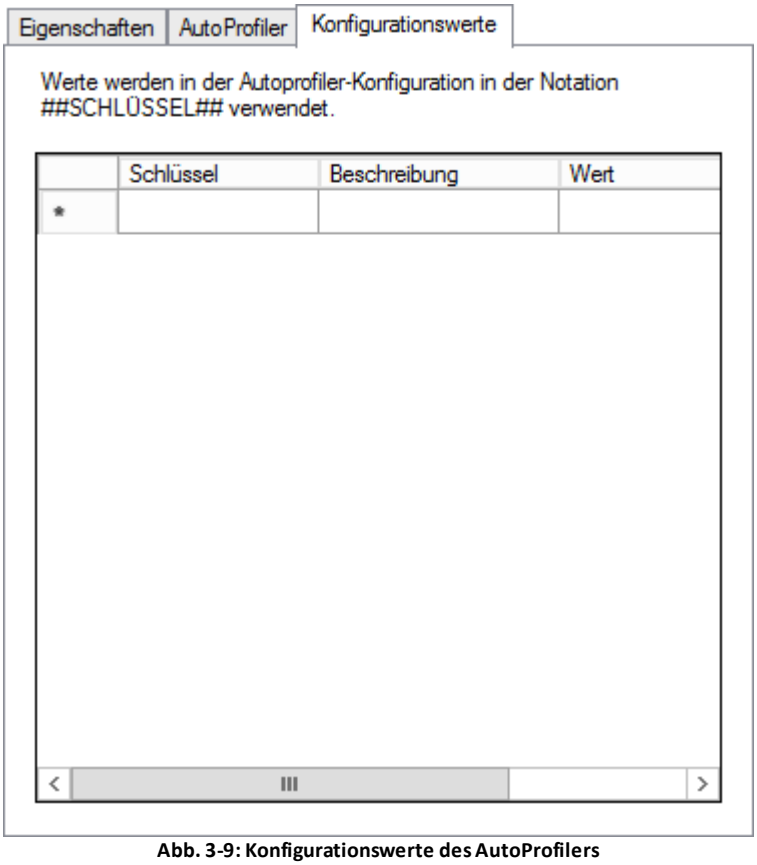


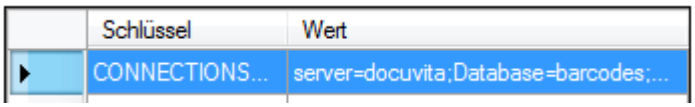
Abb. 3-9: Konfigurationswerte des AutoProfilers

Hier können Variablen für sämtliche AutoProfiler zur Verfügung gestellt werden. So können z.B. Connection Strings einmal zentral deklariert und dann in sämtlichen AutoProfilern verwendet werden. Das Feld Beschreibung ist rein

informativ und hat keinen Einfluss auf die Funktion der Konfiguration.

Variablen / Feld-Wert-Listen

Der docuvita.AutoProfiler arbeitet intern mit Feld-Wert-Listen, die auf unterschiedlichste Art und Weise gefüllt werden können. Über diese Möglichkeit sind automatische Ersetzungen von Variablen im Laufe der Verarbeitungsprozesse einfach zu realisieren.



Die zentral für alle AutoProfiler zu konfigurierenden Variablen werden von ## umschlossen (z.B. wird die innerhalb eines AutoProfilers verwendete Variable ##CONNECTIONSTRING## automatisch durch den Wert „server=docuvita;Database=barcode;Trusted_connection=true“ des Schlüssels „CONNECTIONSTRING“ ersetzt).

Alles Variablen, die nur innerhalb eines einzelnen AutoProfilers nutzbar sind werden von @@ umschlossen.

Die immer verfügbaren Standardvariablen werden von % umschlossen.

Weitere Informationen zum dynamischen Erhalt von Variablen durch das Auslesen von Dokumenten oder SQL-Abfragen finden Sie unter:

Modus und Suchmuster in den Kapiteln Dokumenten Auslesen und Daten-Lookup.

Informationen zu den standard und besonderen Variablen finden Sie im Kapitel Variablen.

AutoProfiler

4 AutoProfiler

Neben der Verarbeitung extern bereitgestellter Importsteuerdateien ist docuvita über den AutoProfiler in der Lage, diese Steuerdateien durch Auswertung von Dokument- und sonstigen Informationen selbst zu erzeugen.

Mit Hilfe des AutoProfilers lassen sich daher automatische Dokumentarchivierungsszenarien realisieren, z.B. für Ausgangs-/ Eingangsbelege oder E-Mails.

4.1 Überblick über die AutoProfiler-Funktion

Als AutoProfiler werden automatisierte Abläufe zum Auswerten und Verschlagworten von bereitgestellten Dokumenten bezeichnet. Es können für eine docuvita Installation beliebig viele AutoProfiler für die verschiedensten Verarbeitungsszenarien konfiguriert werden.

Der Ablauf jedes AutoProfilers gliedert sich grundsätzlich in drei Abschnitte:

	Abschnitt	Funktion
	Quelldokumente abholen und auslesen	Der erste Vorgang des AutoProfilers besteht in der „Abholung“ und Analyse bereitgestellter Dokumente. Diese Abholung erfolgt durch Überwachung definierten Ordner. Im Fall der E-Mail-Archivierung kann auch eine Abholung von einem definierten POP3(S) oder IMAP(S) Postfach erfolgen. Anschließend erfolgt als Kern der AutoProfiler-Funktion die Extraktion von Informationen über die zur Archivierung bereitstehenden Dokumente. Diese Informationen sind z.B.: • Dateiname (z.B. „Rechnung 2007010023.pdf“) • Im Dokument hinterlegte versteckte und daher unsichtbare Felder (Schrift in weißer Farbe auf weißem Hintergrund, daher auf Ausdrucken nicht lesbar. Nur für PDF-Dokumente und Word-Dokumente verfügbar) • E-Mail-Headerdaten, z.B. Absender, Empfänger, E-Maildomänen (nur für E-Mails verfügbar) • Barcodes (nur für PDF- und TIF-Dokumente verfügbar) Die gewonnenen Informationen werden in Variablen als Feld-Wertkombinationen gespeichert, z.B. Lieferscheinnummer=LIF7010337
	Daten anreichern	Die aus den zu importierenden Dokumenten entnommene Metainformation (vgl. erster

	(optional)	Verarbeitungsschritt) kann wahlweise durch einen Lookup-Vorgang oder eine Skriptausführung ergänzt werden. Mit Hilfe eines Lookup-Vorgangs wird eine ODBC-Datenquelle angefragt und Werte aus dieser Datenquelle können zusätzlich zur Verschlagwortung des aktuellen Dokuments verwendet werden. Beispiel: Scan und anschließende Archivierung unterschriebener eigener Lieferscheine. Die Lieferscheine werden mit einem Barcode bedruckt, in dem die Lieferschein-Nummer enthalten ist. Der Lookup-Vorgang sucht anhand des ausgelesenen Barcodes den entsprechenden Kunden (Name/ Nummer) zu diesem Lieferschein. Damit wird die Zuordnung in eine Kundenakte möglich. Zusätzlich zur Ausführung von Daten-Lookups ist es möglich, IronPython-Skripte auszuführen. Mit Hilfe der Skripte kann ein voll an die Kundenbedürfnisse bzw. Aufgabenstellung angepasstes Abarbeiten des Dokumentimports durchgeführt werden. Auch die Ergebnisse eines Daten-Lookups oder eines Skripts stehen zur weiteren Verarbeitung in Variablen als Feld-Wertkombinationen, z.B. Lieferscheindatum=04.06.2014, zur Verfügung.
	XML-Abschnitt mit Metadaten ausfüllen und verarbeiten	Der letzte Verarbeitungsschritt des AutoProfilers besteht in der Erstellung einer dvImport-Datei zur Weitergabe an den docuvita.Server. Hierzu wird ein definiertes Import-Template automatisch um die in den ersten beiden Abschnitten gesammelten Daten ergänzt. Beschreibung: Das Import-Template definiert eine genaue Ablagestruktur zur Einsortierung des zu verarbeitenden Dokumentes in das System. Bei dem Import-Template handelt es sich um einen gültigen XML-Abschnitt gemäß des Kapitels Template. In dem Template können die in den Abschnitten 1 und 2 gewonnenen Variablen als Platzhalter der Form @@Feldname@@ genutzt werden. In dem letzten Schritt wird nun das Template in eine dvImport-Datei umgewandelt. Dabei werden die im Template genutzten Platzhalter (Feldname) durch den jeweils aktuellen Inhalt der Variablen (Feldwert) ersetzt (siehe dazu auch Kapitel Variablen). Die so erzeugte dvImport-Datei wird zusammen mit der zu importierenden Datei an den docuvita.Server zur weiteren Verarbeitung gesendet.

In der Basiskonfiguration des AutoProfilers (siehe auch Kapitel Basiskonfiguration) wird das Intervall in Sekunden angegeben, in dem die AutoProfiler ausgeführt werden. Mehrere AutoProfiler werden in der Reihenfolge 0,1,2,… abgearbeitet. Nach der Ausführung wird wieder auf das Verstreichen des Intervalls gewartet usw. Daher wird stets nur ein AutoProfiler ausgeführt und es kommt nicht zum gleichzeitigen Zugriff mehrerer AutoProfiler.

4.2 Konfiguration

Zur Konfiguration eines AutoProfilers legen Sie wie in Abschnitt AutoProfiler verwalten beschrieben einen neuen AutoProfiler an oder klicken doppelt auf einen bestehenden AutoProfiler. Die Detailschritte werden in den kommenden Abschnitten erklärt.

4.3 Grundeinstellungen

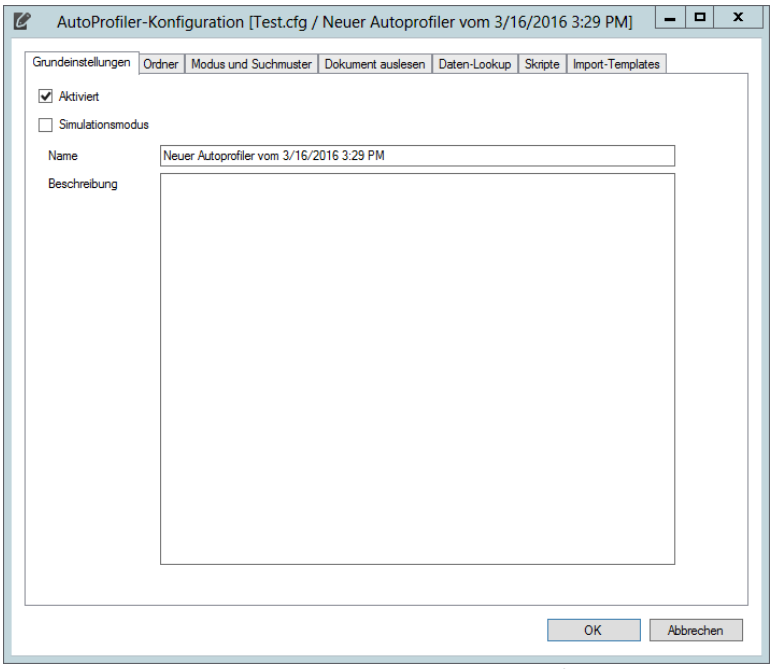


Abb. 4-1: Grundeinstellungen des AutoProfilers

4.3.1 Aktiviert

Bei aktivierter Checkbox wird diese Konfiguration bei der automatischen Verarbeitung berücksichtigt, bei deaktivierter Checkbox nicht.

Konfigurationsänderungen werden nach der Speicherung

umgehend übernommen

Sobald Sie eine einen AutoProfiler auf „Aktiviert“ setzen und die Konfiguration speichern wird die geänderte Konfiguration sofort zur Verarbeitung herangezogen. Um dieses zu verhindern stoppen Sie den docuvita.AutoProfiler Dienst über den docuvita.Admin (und sichern ihn ggf. gegen einen automatisch Start ab) oder setzen den Haken „aktiviert“ erst wenn Sie die Konfiguration des AutoProfilers vollständig abgeschlossen haben.

4.3.2 Name

Name des AutoProfilers. Dieses Feld ist rein informativ und hat keinen Einfluss auf die Funktion der Konfiguration. Wählen Sie für eine vereinfachte Analyse der Logdateien einen möglichst beschreibenden Namen.

4.3.3 Beschreibung

Beschreibung des AutoProfilers. Dieses Feld ist rein informativ und hat keinen Einfluss auf die Funktion der Konfiguration.

4.3.4 Simulationsmodus

Ist die Checkbox *Simulationsmodus* aktiviert findet eine komplette Verarbeitung (inkl. Löschen der Ausgangsdateien) statt, es findet jedoch kein Upload des Ergebnisses zum Server (und somit auch kein Import in docuvita) statt. Die Ergebnisdateien finden Sie im *Zielordner* (siehe Zielordner). Ein nachträglicher Upload an den Server ist nicht vorgesehen, kann jedoch bei Bedarf über einen dvImportContentParser (siehe DvImportContentParser) realisiert werden.

Sie wollen die AutoProfiler Funktionen nutzen, anschließend die Dokumente jedoch nicht nach docuvita importieren?

Nutzen Sie den Simulationsmodus. Sie erhalten das Dokument und die Steuerdateien (dvImport-Datei) mit den ausgelesenen und angereicherten Informationen im Zielverzeichnis zur weiteren Verarbeitung.

4.4 Ordner

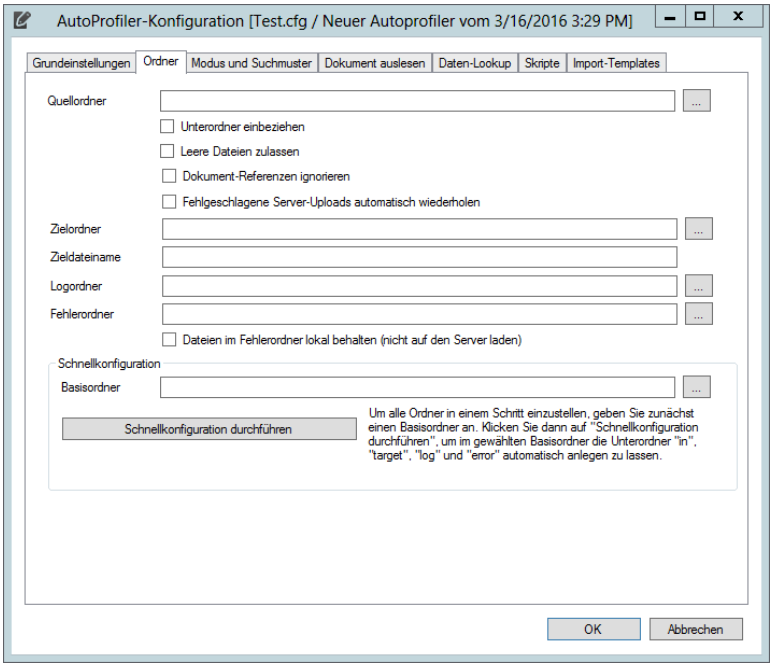


Abb. 4-2: Ordner

4.4.1 Quellordner

- **Quellordner**
  Der *Quellordner* legt den von diesem AutoProfiler zu überwachenden Ordner fest.

- **Unterordner einbeziehen**
  Durch setzen des Hakens *Unterordner einbeziehen* können Ordnerstrukturen rekursiv überwacht werden.

- **Leere Dateien zulassen**
  Durch setzen des Hakens *Leere Dateien zulassen* können auch Dateien verarbeitet werden, die 0 Byte groß sind (z.B. zum Auslösen gewissen Aktionen aus Dritt-Systemen heraus). Leeren Dateien können somit auch lediglich als Steuerdateien zum Start eines AutoProfilers genutzt werden und nicht importiert sondern gelöscht werden. Informationen darüber, sie eine (leere) Datei in der Verarbeitung des AutoProfilers automatisch löschen können, finden Sie in den Kapiteln Template, Standard-Felder für Dokument-Objekttypen und Systemvariablen.

- **Dokument-Referenzen ignorieren**
  Durch setzen des Hakens *Dokument-Referenzen ignorieren* wird ein angegebenes Dokument (siehe *doc.filename* im Kapitel Standard-Felder für Dokument-Objekttypen) nicht mit in das ZIP-File zum Upload an den Server übergeben. Der Server holt die Datei in diesem Fall beim Import vom angegebenen Pfad ab. Der Server muss Zugriff auf das im Attribut *doc.filename* angegebene Verzeichnis und die Datei haben. Die angegebene

Datei wird beim Import gelöscht. Die Funktion ist ggf. sinnvoll, wenn Sie selbst dvImprt-Dateien erstellen (siehe Aufbau der dvImport-Datei) und dies nicht vom AutoProfiler durchführen lassen.

- **Fehlgeschlagene Serveruploads wiederholen**
  Versucht ZIP-Dateien, die auf Grund von z.B. Netzwerkstörungen oder ungültigen Anmeldedaten nicht importiert werden konnten und somit im Zielordner (siehe Zielordner) verblieben sind, bei jedem Durchlauf des AutoProfilers erneut an den docuvita.Server zu übergeben.

Sie wollen Dokumente aus mehreren Verzeichnissen abholen und in einem AutoProfiler verwenden?

Pro AutoProfiler steht nur ein Quellordner zur Verfügung. Nutzen Sie als Workaround einen zusätzlichen AutoProfiler im Simulationsmodus (siehe Simulationsmodus) in Kombination mit dem Modus ParseNothing (siehe Modus und Suchmuster). Geben Sie in der Konfiguration als Fehlerordner den Quellordner des eigentlichen AutoProfilers an. Der so angelegte AutoProfiler dient als Sammler für den richtigen AutoProfiler, der die weitere Verarbeitung übernimmt.

4.4.2 Zielordner

Temporärer Ordner für die Verarbeitung. Hier liegen die fertig verarbeiteten Eingangsdateien zusammen mit den Steuerinformationen für den Import am docuvita.Server.

4.4.3 Zieldateiname

Hier können ausgelesene oder allgemein verfügbare Variablen verwendet werden um den Dateinamen zu ändern. Für den normalen Import ist dieses nicht notwendig. Der Einsatz dieser Funktion ist nur für Szenarien sinnvoll, in denen das Dokument nicht zum docuvita Server zur weiteren Verarbeitung hochgeladen wird sondern nach der Verarbeitung durch den AutoProfiler durch ein anderes Programm (z.b. AutoStore) weiterverwendet werden soll. Es darf keine Dateiendung angegeben werden; es wird die bestehende Dateiendung weiterverwendet.

4.4.4 Logordner

Der Wert im Feld Logordner legt fest, in welchem Verzeichnis der AutoProfiler die Protokolldateien speichern soll.

In dem in dieser Einstellung angegebenen Ordner werden täglich zwei Logdateien angelegt. Die Datei mit dem aktuellen Tagesdatum als Dateinamen enthält das vollständige Log gemäß der Konfiguration. Zusätzlich werden Fehler und Warnungen in eine separate „ERROR“-Datei, ebenfalls mit dem aktuellen Tagesdatum gekennzeichnet, protokolliert.

Import von docuvita 1.x Konfigurationsdateien

War in der übernommenen Konfigurationsdatei einer 1.x Version ein relativer Pfad (z.B. nur „log“) angegeben muss hier nun ein kompletter Pfad eingetragen werden.

4.4.5 Fehlerordner

Der Wert im Feld *Fehlerordner* legt fest, in welchen Ordner Dokumente verschoben werden, bei denen während der Verarbeitung durch den AutoProfiler oder den Import am docuvita.Server ein Fehler aufgetreten ist. Administratoren sollten den Backup-Ordner regelmäßig überwachen, um fehlgeschlagene Importe in Automatikszenarien zu korrigieren bzw. entsprechende Maßnahmen zu ergreifen. Die Überwachung kann auch in einer grafischen Übersicht über den docuvita.Client erfolgen. Siehe dazu auch Administrationshandbuch - docuvita.Server - Customizing.

- **Dateien im Fehlerordner lokal behalten (nicht auf den Server laden)**
  Wenn der Haken gesetzt ist, werden Dateien, bei deren Verarbeitung ein Fehler aufgetreten ist (z.B. kein Barcode auf Dokumenten in einer Scanstrecke mit BarcodeParser), nicht zum Import an den docuvita.Server übergeben. Logeinträge zu den Fehlern können denoch über den docuvita.Client eingesehen werden. Die Benachrichtigungsfunktion zu Fehlerfällen, per E-Mail, kann in diesem Fall ebenso genutzt werden.

4.4.6 Schnellkonfiguration

Geben Sie den Basisordner für eine schnelle Konfiguration an und klicken im Anschluss auf *Schnellkonfiguration durchführen*. Nun werden automatisch alle relevanten Ordner unterhalb des Basisordners angelegt.

Ordner werden automatisch angelegt

Ordner, die noch nicht vorhanden sind, werden vom AutoProfiler bei Bedarf und wenn möglich automatisch angelegt (entsprechende Zugriffsrechte des Benutzers in dessen Kontext der docuvita.AutoProfiler-Dienst läuft, vorausgesetzt).

4.5 Modus und Suchmuster

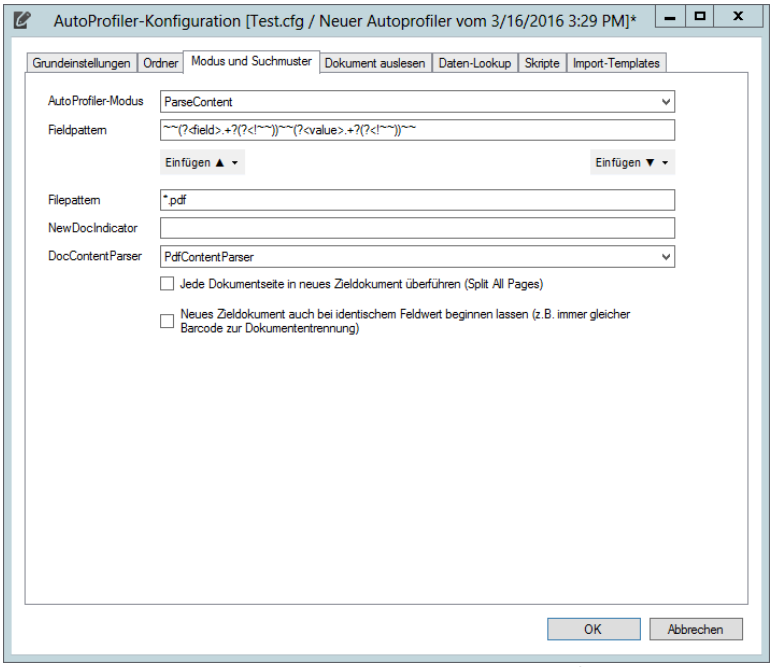


Abb. 4-3: Modus und Suchmuster des AutoProfilers

4.5.1 AutoProfiler-Modus

Legt fest, ob der AutoProfiler die Feldinformationen aus dem Dokumentinhalt und/oder aus dem Dateinamen ausliest. Die möglichen Optionen werden im folgenden Abschnitt genauer beschrieben.

4.5.1.1 ParseContent

Wendet den unter *fieldpattern* (siehe Kapitel Fieldpattern) angegebenen regulären Ausdruck auf den Dokumentinhalt an.

4.5.1.2 ParseFileName

Wendet den unter *filepattern* (siehe Kapitel Filepattern) angegebenen regulären Ausdruck oder das angegebene Suchmuster (Beispiel: *.TIF, Sie können mehrere Dateiformate mit einem Komma oder Semikolon getrennt angeben) auf den Dateinamen an. Es wird entweder ein regulärer Ausdruck oder ein Dateisystem-konformer Wildcard-Ausdruck erwartet.

4.5.1.3 ParseAll

Wendet *ParseContent* und *ParseFileName* auf die zu importierende Datei an. Dabei werden der Dokumentinhalt und der Dateiname unter Verwendung der jeweils in *Fieldpattern* und *Filepattern* angegebenen regulären Ausdrücke ausgewertet.

4.5.1.4 ParseNothing

Wertet keine Dateiinformationen aus. Daher kann zu einem späteren Zeitpunkt nur der Dateiname in Form der stets verfügbaren Variablen (siehe Kapitel

Immer verfügbare Variablen z.B. %fn%) verwendet werden.

4.5.2 Fieldpattern

Legt das Muster fest, mit dem Felder in den zu importierenden Dateien gekennzeichnet sind. Hierbei werden so genannte „reguläre Ausdrücke" verwendet.

Das folgende Beispiel findet Felder, die im Format `@@Feldname@@Wert@@` angegeben wurden:

```
@@(?<field>[^@@]+)@@(?<value>[^@@]+)@@
```

Sie können eigene reguläre Ausdrücke angeben, solange es als Rückgabewerte eine gültige Feld-Wertkombination erwartet wird. Der Feldname muss als „named capture“ `<field>` und der Wert des Feldes als „named capture“ `<value>` zurückgegeben werden. Wobei <field> für den Namen der Variable steht und <value> als Wert der Variable.

Mehrere RegEx-Ausdrücke im Fieldpattern

Man kann in den FieldPattern mehrere RegEx-Ausdrücke Semikolon-getrennt angeben. Falls man im Regex selbst ein Semikolon benötigt, muss es escaped werden (Backslash + Semikolon bzw. \;).

4.5.3 Filepattern

Mit diesem Schlüssel legen Sie fest, welche Dateien von diesem AutoProfiler verarbeitet werden sollen und wie gegebenenfalls Dokumentinformationen aus dem Dateinamen extrahiert werden können. Dateien, die vom *filepattern* eines AutoProfilers nicht erkannt werden, verbleiben im Quellordner, bis sie entweder von einem anderen AutoProfiler verarbeitet oder vom Anwender manuell entfernt werden.

Das folgende Beispiel extrahiert Informationen aus dem Dateinamen und stellt diese als Variable `@@Belegart@@` und `@@Datum@@` zur Verfügung:

```
(?<Belegart>[^_]+)_(?<Datum>[^_]+)\.(?i:pdf)$
```

Der obige reguläre Ausdruck gilt für folgende Dateinamenskonvention:

```
Rechnung_2005-03-22.pdf
```

Sie können eigene reguläre Ausdrücke verwenden, um Informationen aus dem Dateinamen auszulesen. Die von Ihnen verwendeten „named captures“ (z.B. *<Belegart>*) können innerhalb des Konfigurationsabschnitts Import-Templates als Variablen verwendet werden (z. B.: `@@Belegart@@`)

Korrekte Verwendung von Filepattern

Beachten Sie bitte, dass der reguläre Ausdruck in filepattern
- entweder mit einem ^ beginnen muss (Start eines Ausdrucks)
- und/oder mit einem $ enden muss (Ende eines Ausdrucks)

Sollen keine Feldinformationen aus dem Dateinamen ausgelesen werden, können Sie einen Betriebssystem-Platzhalter wie *.PDF verwenden.

4.5.4 NewDocIndicator

Legt das mittels fieldpattern ausgelesene Feld fest, anhand dessen entschieden wird, wann beim Import mehrerer Belege in **einer** Quelldatei ein neues Dokument begonnen wird. Immer, wenn sich der Wert des hier angegebenen Feldes auf einer Seite der zu importierenden Datei ändert, geht der docuvita.AutoProfiler davon aus, dass hier ein neuer Beleg beginnt und erstellt ein neues Dokument.

Um beispielsweise bei einem Wechsel der Kundennummer in einem Sammeldruck ein neues Dokument zu erstellen, muss das Feld NewDocIndicator mit dem Wert “Kundennummer“ gefüllt werden.

Sie können mehrere Felder mit Komma oder Semikolon getrennt angeben. Es dürfen keine Leerzeichen vorkommen. Die angegebenen Felder werden mit „ODER“ verknüpft, d.h. ein neuer Wert in einem der spezifizierten Felder löst den Neubeginn eines Dokuments aus.

Für den NewDocIndicator kann mit Hilfe von Regular Expressions auch auf einen Teil einer Variablen geprüft werden. Siehe dazu auch Kapitel Spezial Variablen.

Ein Beispiel zur Anwendung von NewDocIndicator ist der Sammeldruck von Belegen in ein PDF-Zieldokument. Wird dieses Zieldokument archiviert, können durch die Angabe der Belegnummer für NewDocIndicator dennoch einzelne Dokumente ins Archiv gespeichert werden. Das Feld Belegnummer wird für die Auftrennung des Sammeldrucks in Einzelbelege verwendet.

Das Attribut NewDocIndicator wird nur bei der Verarbeitungsweise ParseContent mit elektronisch auslesbaren PDF- sowie TIF/PDF-Dokumenten mit Barcode-Verarbeitung unterstützt.

Mehrere Belege in einer Quelldatei

Falls mehrere Belege aus technischen Gründen nur in einer gemeinsamen Quelldatei erstellt werden können, kann das

Importprogramm diese in mehrere Einzeldokumente aufteilen und einzeln importieren.

Dabei erstellt das Importprogramm immer dann ein neues Dokument, wenn sich der Wert des als Indikator angegebenen Feldes ändert (vgl. NewDocIndicator).

Diese Funktion steht nur bei elektronisch auslesbaren PDF-Belegen (i.d.R. Ausgangsbelege) und der Barcodeauswertung zur Verfügung (PdfContentParser und BarcodeContentParser).

4.5.5 DocContentParser

Die Angabe des DocContentParser-Moduls erlaubt die Steuerung der Auswertung von Quelldateien.

Gültige Werte sind:

- None
- PdfContentParser
- WordContentParser
- BarcodeContentParser
- MailContentParser
- dvImportContentParser
- XmlContenParser
- CscContentParser
- <optional installierte Plugins im Ordner *Dispatchers*>

Welche Funktionen über die verschiedenen DocContentParser möglich sind wird in dem kommenden Abschnitt genauer erklärt.

Import von 1.x Konfigurationen

Falls bisher Einstellungen nicht definiert wurden werden nun die Standardeinstellungen genutzt:

War kein AutoProfiler Modus angegeben ist es nun ParseContent

War kein DocContentParser angegeben ist es nun der PDFContentParser

4.5.6 Weitere Funktionen

- **Jede Dokumentseite in neues Zieldokument überführen (Split All Pages)**
  Die Option *Jede Dokumentseite in neues Zieldokument überführen (Split All Pages)* zerteilt ein eingehendes Dokument in Einzelseiten und verarbeitet im Weiteren jede Seite separat. Ein erneutes Zusammenführen der Einzelseiten zu einem Gesamtdokument ist nicht vorgesehen. Verwenden Sie diese Funktion nur, wenn wirklich jede Seite eines Scanstapels (auch digital übergebene PDF-Stapel) separat verarbeitet werden soll.
  - Funktioniert nur in Kombination mit dem *AutoProfiler-Modus ParseAll* und *ParseContent*.
  - Funktioniert nur mit den *DocContentParsern BarcodeContentParser* und *PdfContentParser*

- **Neues Zieldokument auch bei identischen Feldwerten beginnen lassen (z.B. immer gleicher Barcode zur Dokumententrennung)**
  Mit dieser Option können Sie ein und den selben Barcode immer wieder als Dokumententrenner verwenden. Immer wenn ein neuer oder der gleiche NewDocIndicator auf einer Seite eines Scanstapels (auch digital übergebene PDF-Stapel) auftaucht wird ein neues Zieldokument begonnen
  - Funktioniert nur in Kombination mit dem *AutoProfiler-Modus ParseAll* und *ParseContent*
  - Funktioniert nur mit den *DocContentParsern BarcodeContentParser* und *PdfContentParser*

4.6 Dokumenten Auslesen

4.6.1 PdfContentParser

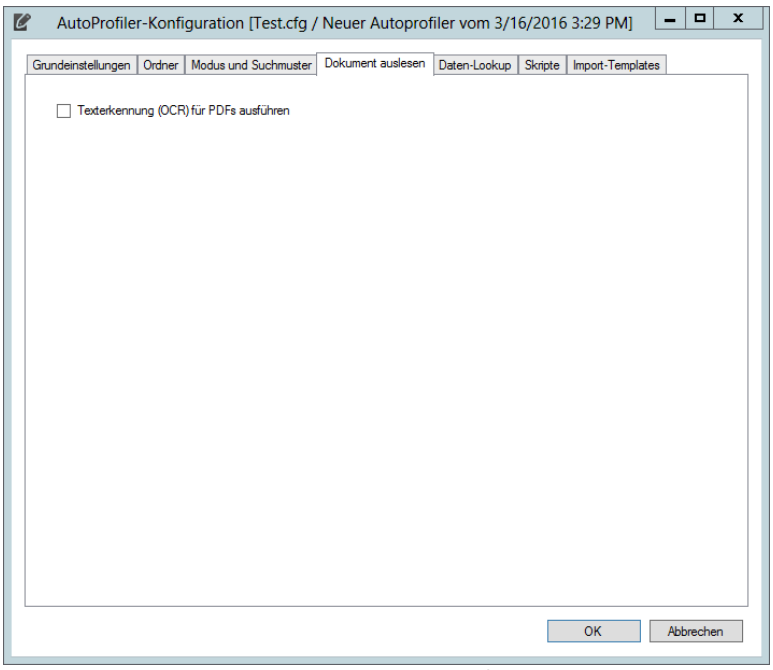


Abb. 4-4: Dokument auslesen mit PdfContentParser

Mit Hilfe des PdfContentParsers werden PDF-Dokumente analysiert. Setzen Sie

den Arbeitsmodus auf *ParseContent* oder *ParseAll*.

Aus PDF-Dokumenten werden in der Regel versteckte Felder ausgelesen. Der PdfContentParser dient dazu, elektronische PDFs (d.h. mit Textinhalt) seitenweise auszuwerten. Der Textinhalt jeder Seite wird bei der Verarbeitung ausgelesen, und mit Hilfe des regulären Ausdrucks in *fieldpattern* wird die Liste der Variablen gebildet. Die Variablen werden in Daten-Lookups, Bedingungsprüfungen und Templates in @@-Notation verwendet (z.B. `@@Belegnummer@@`). Zusätzlich können auch reguläre Ausdrücke angewendet werden. Die Ersetzungslogik ist dabei wie folgt:

1) Variable wird direkt gefunden → Ersetzung durch Variablenwert, z.B. `@@Belegnummer@@` → RE12345
2) Regulärer Ausdruck soll auf Variable angewendet werden →Ergebnis wird eingetragen. Beispiel: `@@Belegnummer::RE(\d{5})@@` → 12345
3) Regulärer Ausdruck wird auf den ausgelesenen Seitentext angewendet. Beispiel: `@@Mandant:\s(\d{3})\s+@@`

Die regulären Ausdrücke der Beispiele 2 und 3 müssen genau einen Rückgabe-/Ergebniswert aufweisen. Siehe dazu auch Kapitel Spezial Variablen.

Diese versteckten Felder fügen Sie im Quellprogramm, welches die Dokumente an den AutoProfiler übergibt ein, in das Dokument ein. Dieses geschieht meist in einem Belegdesigner eines ERP- oder Warenwirtschaftssystems. Achten Sie darauf, zusammenhängende Bereiche zu beschreiben. Im Regelfall wird in der Warenwirtschaft daher ein Formelfeld zum Einsatz kommen.

PDF-Dokumente gelangen beispielsweise über einen Software-Druckertreiber in den Quellordner des AutoProfilers. Damit lässt sich z.B. eine automatische Archivierung im Moment des Belegdrucks realisieren.
Bei PDF-Dokumenten wird das Attribut NewDocIndicator unterstützt (vgl. Hinweis in Kapitel NewDocIndicator).

Probleme beim Auslesen von versteckten Feldern

In den versteckten Feldern können unter Verwendung mancher PDF-Druckertreiber beim Auslesen Leerzeichen auftauchen, obwohl diese in der Druckdefinition nicht vorhanden sind. Passen Sie in diesem Fall die Schriftart und Größe in der Druckvorlage an.

4.6.1.1 Texterkennung (OCR) für PDFs durchführen

Es kann direkt vom AutoProfiler eine OCR durchgeführt werden. Somit können auch Schlagworte aus gescannten Dokumenten zur Dokumententrennung erkannt werden. Es ist so auch möglich gewisse Schlagworte aus dem Text herauszufiltern und zur Klassifizierung des Dokumentes heranzuziehen. Siehe dazu Kapitel Pagetext.

Erkennungsqualität und Performance

Bitte beachten Sie, dass es bei der Texterkennung von gescannten Dokumenten keine 100%-ige Sicherheit gibt. Die Erkennungsqualität hängt entscheidend von der Qualität des Originalbeleges und der Scanqualität ab.

Eine OCR bereits bei der Einlieferung der Dokumente durchzuführen kann zu Performanceeinbußen bei der Verarbeitung führen, da der AutoProfiler erst weiter arbeitet, wenn ein Vorgang abgeschlossen ist.

Es eignen sich nicht alle PDF-Dateien zur Durchführung einer Texterkennung. Geschützte PDFs werden z.B. nicht unterstützt.

Bitte berücksichtigen Sie diese Umstände, wenn Sie diese Funktion zum Auslesen von Variablen oder zur Dokumenttrennung nutzen wollen.

4.6.2 WordContentParser

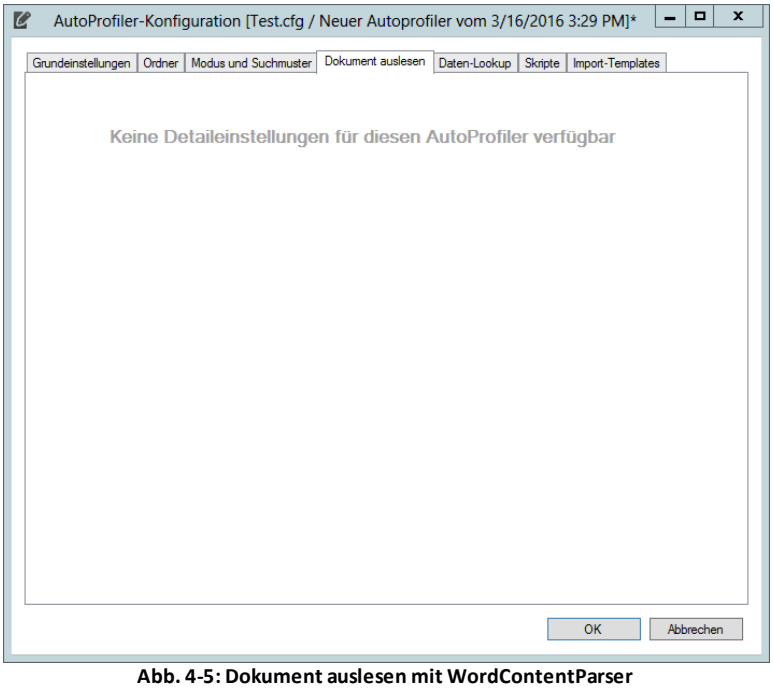


Abb. 4-5: Dokument auslesen mit WordContentParser

Die Verarbeitung von Word-Dokumenten erfolgt wie bei PDFs. Die ausgelesenen Werte aus dem Dokument werden jedoch nicht einer einzelnen Seite bzw. bis zur Trennung der Seiten zu einem neuen Dokument zugeordnet, sondern gesammelt und für das komplette Dokument verwendet.

Setzen Sie den Arbeitsmodus auf *ParseContent* oder *ParseAll*. Den Wert des

Attributes *DocContentParser* legen Sie auf *WordContentParser* fest.

Das Feld *NewDocIndocator* und damit eine Belegtrennung wird bei Word-Dokumenten nicht unterstützt.

Auch beim ***WordContentParser*** steht der gesamte Textinhalt als Variable zur Verfügung und kann weiter verwendet werden. Siehe dazu Kapitel Pagetext.

4.6.3 BarcodeContentParser

4.6.3.1 Allgemeine Einstellungen

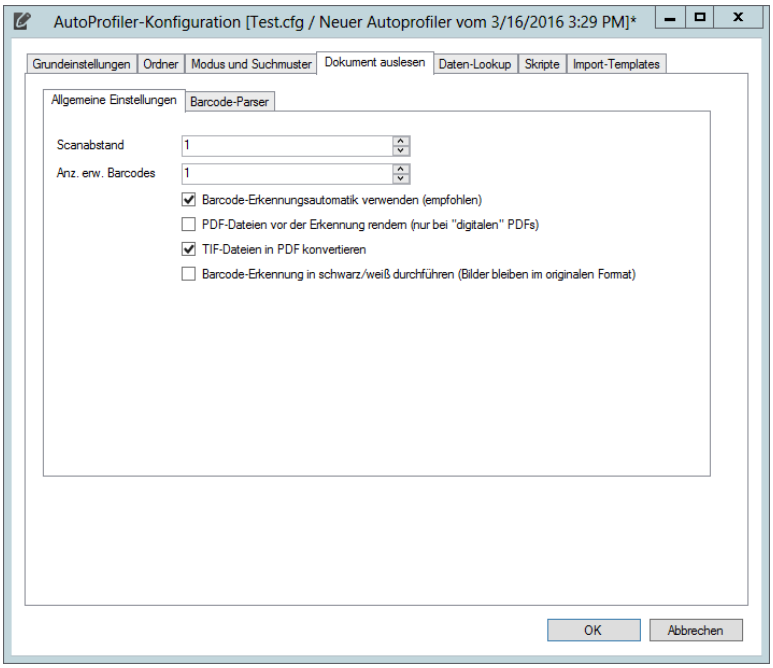


Abb. 4-6: Dokument auslesen mit BarcodeParser

Legen Sie den Wert des Attributes *DocContentParser* auf *BarcodeContentParser* fest, um Barcodes aus PDF- und TIF-Dokumenten auszulesen. Dabei muss der Arbeitsmodus des AutoProfilers auf *ParseContent* oder *ParseAll* festgelegt sein.

4.6.3.1.1 Scanabstand

Mit dem Parameter Scanabstand kann der Abstand des Scanintervalls definiert werden. Hier können Werte von 1 bis 10 angegeben werden. Je nach verwendetem Barcode sowie Druck- und Scanqualität kann die Barcodeerkennung mit diesem Parameter optimiert werden. Ein Wert von 10 bedeutet mehr Performance, bei einem Wert von 1 ist die Genauigkeit besser.

4.6.3.1.2 Anz. Erw. Barcodes

Die voreingestellte Anzahl der expectedbarcodes (wenn kein Wert eingetragen ist) entspricht der Anzahl der definierten Barcodeparser +2. Kommen wissentlich mehr Barcodes pro Seite zum Einsatz und sollen ausgelesen werden, kann die Performance der Verarbeitung über den Parameter Anz. Erw. Barcodes optimiert werden.

4.6.3.1.3 Barcode-Erkennungsautomatik verwenden

Es wird empfohlen die Barcode-Erkennungsautomatik zu verwenden. Dieses ist vor allem dann sinnvoll, wenn die Helligkeitsverteilung nicht einheitlich ist (z.B. bei Graustufen- und Farbscans).

4.6.3.1.4 PDF-Dateien vor Erkennung rendern

Um auch elektronisch erstellte PDFs mit mehreren Bildern z.B. auch über einen Barcodeparser verarbeiten zu können gibt es die Möglichkeit, dass alle PDFs eines AutoProfilers intern für die Verarbeitung mit 300 dpi als Bild aufbereitet werden.

Nicht alle PDF- und TIF-Formate eignen sich zur Verarbeitung durch den AutoProfiler

Es kann z.B. zu bei manchen komprimierten (kompakt-) PDFs zu Problemen in der Verarbeitung kommen. Auch bei manchen JPEG komprimierten TIFs sin Probleme möglich.

Unsere Scan-Empfehlung:

Format:	TIF
Komprimierung:	CCITT/G4 (s/w)
Auflösung:	300 dpi

Bitte beachten Sie außerdem, dass der AutoProfiler PDF-Dokumente bei der Verarbeitung seitenweise neu zusammenstellt. Hierdurch können z.B. bereits bestehende Signaturen gebrochen werden.

4.6.3.1.5 TIF-Dateien in PDF konvertieren

Das Barcodemodul kann Dokumente nach der Verarbeitung durch den AutoProfiler auch von TIF in PDF/A wandeln. Dazu muss der entsprechende Haken gesetzt werden.

4.6.3.1.6 Barcode-Erkennung in s/w

Für die Erkennung des Barcodes kann das Bild von einem Graustufen- oder Farbformat umgewandelt werden. Diese Umwandlung wird lediglich für die interne Verarbeitung verwendet, das Ausgabedokument bleibt im originalen Format vorhanden.

4.6.3.2 Barcode-Parser

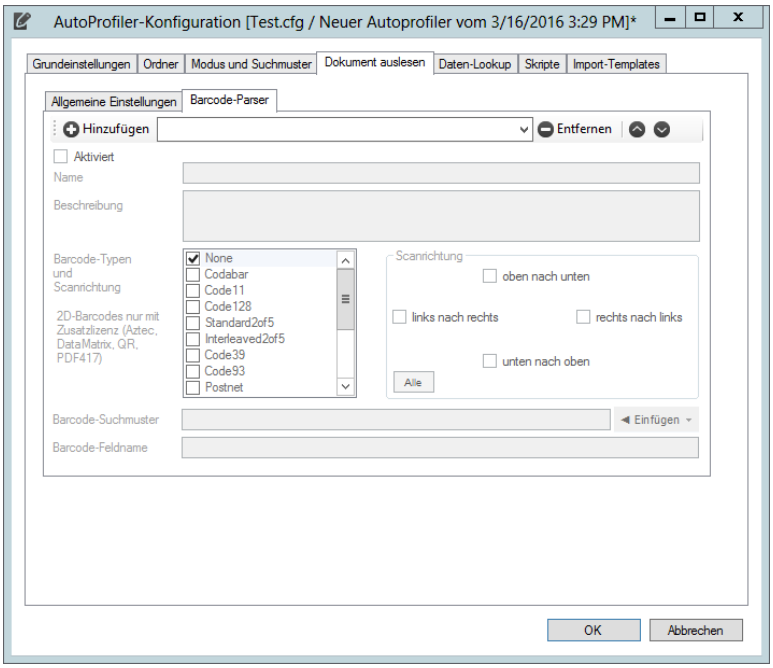
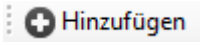
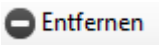


Abb. 4-7: Dokument auslesen mit BarcodeParser

Es können mehrere Barcodes ausgelesen werden. Für jeden zu bestimmenden Barcode wird ein eigenes Suchmuster definiert.

Über den Button 

können Sie ein zusätzliches Suchmuster hinzufügen. Über die Buttons  können Sie die Reihenfolge der Suchmuster ändern und über den Button 

ein bestehendes Suchmuster löschen.

4.6.3.2.1 Aktiviert

Über die Checkbox *Aktiviert* kann ein BarcodeParser aktiviert oder deaktiviert werden.

4.6.3.2.2 Name

Name des BarcodeParsers. Dieses Feld ist rein informativ und hat keinen Einfluss auf die Funktion der Konfiguration. Ein sinnvoll und sprechend vergebener Name kann jedoch die Auswertbarkeit von Log-Dateien stark verbessern.

4.6.3.2.3 Beschreibung

Beschreibung des BarcodeParsers. Dieses Feld ist rein informativ und hat keinen Einfluss auf die Funktion der Konfiguration.

4.6.3.2.4 Scanrichtung

Um sicherzustellen, dass beliebig aufgebrachte Barcodes gelesen werden können, wählen Sie die zu erwartenden Scanrichtungen aus. Die Barcodes müssen dennoch horizontal oder vertikal parallel zum Rand des Dokuments aufgebracht werden.

Gültige Barcodetypen

Folgende Strichcode-Barcodearten können erkannt werden:
- Codabar
- Code11
- Code128 (alphanumerisch mit Prüfziffer)
- Standard2of5
- Interleaved2of5
- Code39 (Barcode 3 aus 9 mit bzw. ohne Prüfziffer)
- Code93
- Postnet
- EAN13
- EAN8

Folgende 2D-Barcodearten können erkannt werden:
- PDF417
- PDF417Compact
- DataMatrix
- QR
- Aztec

Das 2D-Barcodemodul ist eine aufpreispflichtige Zusatzkomponente für den AutoProfiler und muss sparat lizenziert werden.

Barcodes werden immer horizontal oder vertikal parallel zum Dokument ausgerichtet gesucht

Aufgeklebte Barcodes müssen daher immer parallel zu einer Seite des Blatts aufgebracht werden damit sie auch korrekt ausgelesen werden können. Hier ein paar Beispiele:

4.6.3.2.5 Barcode-Suchmuster

Legt ein Erkennungsmuster für den Barcode als regulären Ausdruck fest. Damit kann der einzulesende Barcode z.B. auf eine bestimmte Länge oder etwa den Beginn mit Buchstaben geprüft werden. Nicht dem Erkennungsmuster entsprechende Barcodes werden verworfen.

Verwenden Sie als Erkennungsmuster einen regulären Ausdruck, z.B. `\d{7}` zur Eingrenzung auf Barcodes mit 7 Ziffern. Das Attribut kommt dann zum Einsatz, wenn mehrere Barcodes auf Dokumenten vorkommen können und ein bestimmter für die Verarbeitung benötigt wird.

Wird das Erkennungsmuster nicht festgelegt oder fehlt das entsprechende Attribut, so wird der erste gefundene Barcode, der nach dem angegebenen Attribut *Barcodetyp* gefunden wurde, zurückgegeben.

Entsprechen mehrere Barcodes den Suchmustern, wird der letzte Barcode in die Variable für die weitere Verarbeitung gespeichert. Achtung: Der letzte gefundene Barcode muss nicht dem letzten Barcode auf dem Papier entsprechen!

Nutzen Sie, wenn Sie die Möglichkeit haben, Barcodes mit speziellen Mustern und Qualifizierern

Mit einem speziellen Barcodemuster, wie z.B. dv123456, verhindern Sie, dass Barcodes auf externen Dokumenten in Ihr Suchraster fallen und somit die Verarbeitung behindern. So suchen Sie z.B. immer nach einem Barcode der mit *dv* beginnt und von einer 6-stelligen Zahlenfolge abgeschlossen wird.

4.6.3.2.6 Barcode-Feldname

Legt fest, mit welchem Feldnamen der Barcode zurückgegeben wird. Dieser Feldname wird als Variable in der Konvention `@@Feldname@@` im XML-Quellabschnitt zur Weiterverarbeitung eingetragen. Siehe dazu auch Kapitel Ausgelesene Variablen.

4.6.4 MailContentParser

Bei der E-Mailarchivierung können Sie im Import-Template (vgl. Import-Templates und Aufbau der dvImport-Datei) auf einen festen Satz an Variablen, die automatisch aus dem E-Mailheader ausgelesen werden, zugreifen (siehe E-Mail Variablen). Für weitere Informationen zur E-Mailarchivierung beachten Sie auch unsere separate Kurzanleitung zur E-Mailarchivierung mit Ablaufdiagramm.

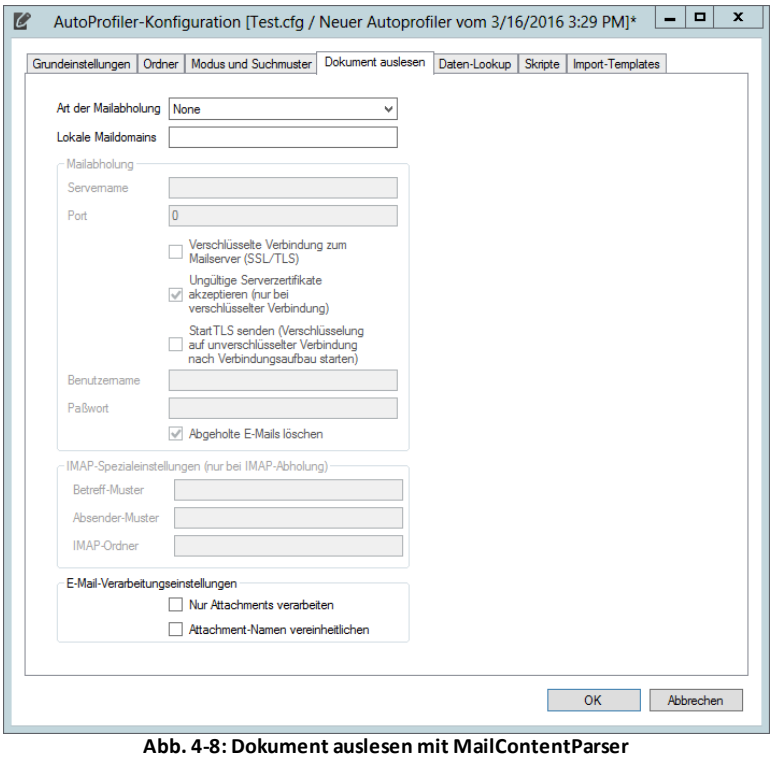


Abb. 4-8: Dokument auslesen mit MailContentParser

Wird der Arbeitsmodus auf *ParseContent* bzw. *ParseAll* festgelegt und als *DocContentParser* der Wert *MailContentParser* eingetragen, kann der AutoProfiler zur Verarbeitung von E-Mails genutzt werden.

Das zulässige Format für E-Mails ist RFC/EML. Es werden die unveränderten, d.h. bereitgestellten oder abgeholten E-Mails archiviert – alle Anhänge, Bilddateien (nicht wenn die Bilder in HTML E-Mails lediglich als enthalten Links sind) etc. eingeschlossen.

Spamfilter

Mit Hilfe des MailContentParser ist eine automatische Mailarchivierung möglich, die z.B. alle ein- und ausgehenden E-Mails betrifft. Vor der Archivierung sollte daher eine leistungsfähige Spamfilterung erfolgen, um die Archivierung von nicht brauchbaren E-Mails zu verhindern.

Bereitstellung ein- und ausgehender E-Mails

Die Archivierung ein- und ausgehender E-Mails kann mit Hilfe des MailContentParser automatisiert werden, wenn der E-Mailserver die zu archivierenden E-Mails in den Quellordner bereitstellt.

Microsoft Exchange Server

Beim Microsoft Exchange Server kann dies z.B. über ein SMTP-EventSink geschehen. Dadurch können automatisch z.B. alle über einen bestimmten SMTP-Konnektor laufenden Nachrichten in Kopie im Archivverzeichnis bereitgestellt werden.

4.6.4.1 Art der Mailabholung

Leg fest über welches Protokoll die E-Mails von Server abgerufen werden sollen (*POP3* oder *IMAP*). Wird *none* ausgewählt wird der unter dem Reiter *Ordner* eingestellte Quellordner auf E-Mails überprüft (siehe Quellordner).

Wird die Abholung von E-Mails von einem POP3- oder IMAP Postfach gewünscht, muss die entsprechende Konfiguration zur Mailabholung ausgefüllt werden.

4.6.4.2 Lokale Maildomains

Anhand der Einstellung *localMailDomains* wird eine externe E-Mail-Adresse ermittelt, die für die Archivierung/Verschlagwortung herangezogen werden kann. Es können auch mehrere Domains mit Komma oder Semikolon getrennt angegeben werden. Die externe E-Mail-Adresse wird wie folgt ermittelt:

- bei eingehenden Mails die externe Absenderadresse
- bei ausgehenden Mails die erste externe Empfängeradresse bzw. erste CC-Adresse, falls nur interne Empfänger
- bei internen Mails die erste Empfängeradresse

4.6.4.3 Server

Geben Sie hier den Namen oder die IP des E-Mailservers an.

4.6.4.4 Port

Geben Sie hier den Port des E-Mailservers an (in der Regel POP3: 110, POP3(S): 995, IMAP: 143, IMAP(S):993).

4.6.4.5 Verschlüsselte Verbindung zum Mailserver

Der Haken muss gesetzt werden wenn der AutoProfiler die E-Mails über einen verschlüsselten Weg abrufen soll.

4.6.4.6 Ungültige Serverzertifikate akzeptieren

Mit diesem Haken kann bei aktivierter Verschlüsselung die Überprüfung des Serverzertifikates deaktiviert werden. Tun Sie dieses nur, wenn Sie sich sicher sind, dass der AutoProfiler garantiert die Verbindung mit dem korrekten Server aufnimmt (dieses kann z.B. notwendig sein, wenn auf dem E-Mailserver nur ein selbst signiertes Zertifikat hinterlegt ist oder das bestehende Zertifikat abgelaufen ist).

Unsicherheit durch ungültige Serverzertifikate

Werden ungültige Serverzertifikate automatisch akzeptiert können darüber man-in-the-middle Angriffe ausgeführt werden und somit im schlimmsten Fall die Kommunikation zwischen AutoProfiler und E-Mailserver abgehört oder manipuliert werden.

4.6.4.7 StartTLS senden

StartTLS kann optional genutzt werden (wird von manchen Mailservern vorausgesetzt). Nach dem Verbindungsaufbau wird eine Verschlüsselung auf einer unverschlüsselten Verbindung gestartet.

4.6.4.8 Benutzername

Tragen Sie unter *username* den Benutzernamen ein, mit dem sich das Importprogramm am POP3-Server anmeldet (Zugriffsberechtigter Benutzer für das Archivpostfach).

4.6.4.9 Paßwort

Geben Sie das Kennwort des POP3- / IMAP-Benutzers an.

4.6.4.10 Abgeholte E-Mails löschen

Setzen Sie den Haken bei *Abgeholte E-Mails löschen* falls die E-Maildateien vom Server gelöscht werden sollen, nachdem sie abgeholt wurden. Für die erste Inbetriebnahme/Testbetrieb oder einen Testlauf sollten Sie den Haken nicht setzen. Im laufenden Betrieb muss der Haken gesetzt sein, da ansonsten die gleichen E-Mails bei jedem Durchlauf des AutoProfilers erneut verarbeitet werden.

4.6.4.11 Betreff-Muster

Hier kann eine Regular Expression angegeben werden. E-Mails bei denen der Betreff nicht dem Suchmuster der Regular Expression entspricht werden ignoriert und auf dem Server belassen.

Diese Funktion kann nur bei einem Zugriff per IMAP genutzt werden.

4.6.4.12 Absender-Muster

Hier kann eine Regular Expression angegeben werden. E-Mails bei denen der Absender nicht dem Suchmuster der Regular Expression entspricht werden ignoriert und auf dem Server belassen.

Diese Funktion kann nur bei einem Zugriff per IMAP genutzt werden.

4.6.4.13 IMAP-Ordner

Geben Sie einen speziellen IMAP-Ordner an aus dem die E-Mails abgeholt werden sollen. Es können auch einzelne Ordner aus verschachtelten Strukturen (z.B. Ablage/2014). Für Ordner unterhalb des Posteingangs wird für gewöhnlich der Bezeichner Inbox (kann je nach Mailserver abweichen) verwendet (z.B. Inbox/docuvita).

Diese Funktion kann nur bei einem Zugriff per IMAP genutzt werden.

4.6.4.14 Nur Attachments einlesen

Wenn Sie nur die Anhänge weiterverwenden wollen setzen Sie diesen Haken.

Exchange Archivpostfach

In einem Exchange Archivpostfach werden die einzelnen zu archivierenden E-Mails als Anhänge von Archivobjekten (Container-E-Mails) abgelegt. Da Sie im Regelfall nur die eigentliche E-Mail, nicht jedoch die Container-E-Mail archivieren wollen nutzen Sie die Option *Nur Attachments einlesen*.

4.6.4.15 Attachment-Namen vereinheitlichen

Betreff wird um Sonderzeichen bereinigt und als Dateiname für Anhänge genutzt. Bei mehreren Anhängen wird fortlaufend nummeriert (z.B. Betreff(2).pdf).

4.6.5 DvImportContentParser

Verarbeitet Dateien im docuvita dvImport-Format. Dateireferenzen werden systemweit ausgewertet und beim Importvorgang aus dem jeweiligen Quellverzeichnis abgeholt und für den docuvita.Server bereitgestellt. Bitte beachten Sie, dass die Quellverzeichnisse für das Dienstkonto des AutoProfilers zugänglich sind. Für das Format der dvImport-Dateien siehe Kapitel Aufbau der dvImport-Datei.

Legen Sie den Wert des Attributes *DocContentParser* auf *DvImportContentParser* fest, um direkt DvImportDateien zu verarbeiten. Dabei muss der Arbeitsmodus des AutoProfilers auf *ParseContent* oder *ParseAll* festgelegt sein.

4.6.6 XmlContentParser

Mit dem XmlContentParsers können einzelne Werte aus XML-Knoten ausgelesen werden. In dem folgenden Beispiel kann nach Auswertung der XML-Datei mit `@@element@@` der Wert aus `data` und mit `@@element2@@` der Wert `data2` in den Import-Templates verwendet werden:

```
<?xml version="1.0" encoding="UTF-8"?>
<beispiel>
  <element>data</element>
  <element2>data2</element2>
</beispiel>
```

Gleichnamiges PDF statt der XML-Datei importieren

Verwenden Sie ein Skript (siehe Skripte) welches Sie den Dateinamen zum Import manipulieren können und im Import-Template (siehe Import-Templates) dann wie folgt verwenden:

```
doc.Filename="@@PDFFILE@@.pdf"
```

Hier das notwendige Skript:

```
import clr
import System
from System import Convert,String,DateTime
from System.Collections import Hashtable
from System.IO import *

fi = FileInfo( sourceFilename )
fieldValues["PDFFILE"] = Path.Combine( fi.DirectoryName,
Path.GetFileNameWithoutExtension( sourceFilename))
log.InfoFormat("PDF-Datei: {0}", fieldValues["PDFFILE"])
#File.Delete(docFilename)
```

Die Quell-XML kann gelöscht werden, wenn im Import-Template zusätzlich zum `doc.Filename` mit der aus der XML ausgelesenen Variablen (z.B. `doc.Filename="@@VariableAusDerXML@@")` noch `doc.FileDelete="@@FileDelete@@"` angegeben wird. Weitere Informationen hierzu finden Sie in den Kapiteln Template, Standard-Felder für Dokument-Objekttypen und Systemvariablen.

Legen Sie den Wert des Attributes *DocContentParser* auf *XmlContentParser* fest, um XML-Dateien auszulesen. Dabei muss der Arbeitsmodus des AutoProfilers auf *ParseContent* oder *ParseAll* festgelegt sein.

4.6.6.1 XSLT-Transformation

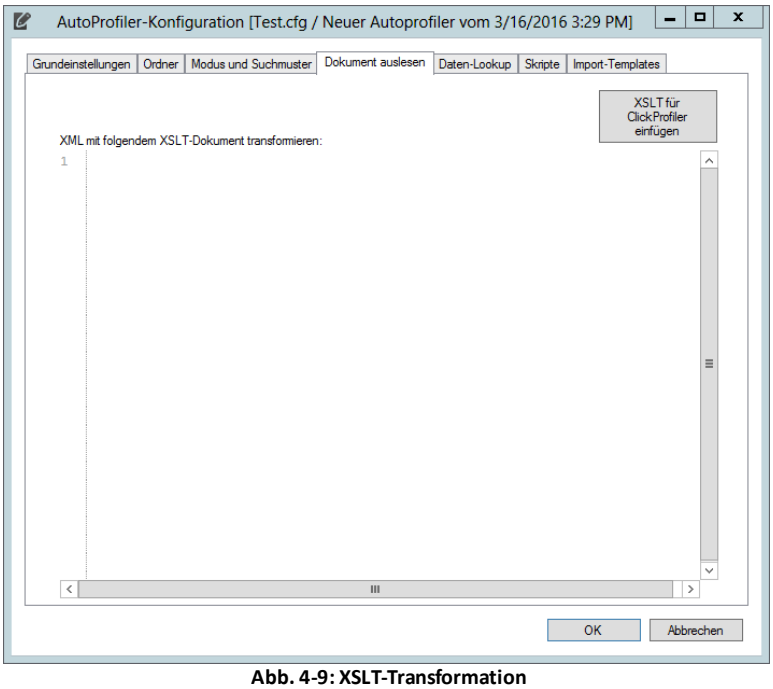


Abb. 4-9: XSLT-Transformation

Beispiel

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://
www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" indent="yes" omit-xml-
declaration="no" />

    <xsl:key name="kFieldName" match="TagMatch"
use="@identifier"/>

    <!-- Nur Child-Element(e) des Root-Elements
verarbeiten -->
    <xsl:template match="/*">
        <xsl:apply-templates/>
    </xsl:template>

    <!-- "alles" kopieren. Falls Unter-templates gefunden
werden, diese verarbeiten -->
    <xsl:template match="DocumentContainer/Tags/TagMatch/
@* | node()">
    <xsl:copy>
      <xsl:apply-templates select="DocumentContainer/
Tags/TagMatch/@* | node()"/>
    </xsl:copy>
    </xsl:template>

  <!-- Unter-Template "eindeutige ID" -->
  <xsl:template
    match="TagMatch[ generate-id() = generate-
id(key('kFieldName', @identifier)[1]) ]">

        <xsl:element name="{@identifier}">
            <xsl:value-of select="Value" />
        </xsl:element>
   </xsl:template>

   <!-- Templates, die "verschluckt" werden sollen -->
   <xsl:template match="@*" />
   <xsl:template match="TagMatch" />
   <xsl:template match="Notes" />
   <xsl:template match="Documents" />

</xsl:stylesheet>
```

4.6.7 CsvContentParser

Die Verarbeitung einer CSV-Datei erfolgt zeilenweise und ist, anders als bei allen anderen Parsern, nicht ein Vorgang pro Datei, sondern ein Vorgang pro Zeile.

Der CsvContentParser eignet sich im Besonderen für den Import von Stamm- oder Bestandsdaten. Wenn die CSV-Datei Spaltenüberschriften enthält können

diese in Folge als Variablen verwendet werden. Enthält die CSV-datei keine Spaltenüberschriften, so werden dynamisch Variablennamen vergeben. Weitere Informationen finden Sie unter Einlesen ab Zeile.

Legen Sie den Wert des Attributes *DocContentParser* auf *CsvContentParser* fest, um Datensätze aus CSV-Dateien auszulesen. Dabei muss der Arbeitsmodus des AutoProfilers auf *ParseContent* oder *ParseAll* festgelegt sein.

4.6.7.1 Einlesen ab Zeile

Die Einstellung *Einlesen ab Zeile* steuert, ob zu Beginn der Datei Zeilen (die ggf. nicht ins CSV-Schema passen, z.B. Überschrift der gesamten Tabelle) übersprungen werden sollen. Die Zählung beginnt bei 1 für die erste Spalte.

Enthält die erste eingelesenen Zeile Spaltenüberschriften, so müssen Sie dieses mit der Einstellung *Erste eingelesene Zeile enthält Spaltenüberschriften* festlegen. Bei der ersten eingelesenen Zeile handelt es sich um die Zeile, die unter *Einlesen ab Zeile* angegeben wird.

- Wenn die Funktion aktiviert ist, dann stehen die Variablen je Zeile/ Datensatz unter den Spaltenbezeichnern zur Verfügung (@@Spaltenbezeichnung1@@, @@Spaltenbezeichnung2@@ usw.).
- Wenn die Funktion nicht aktiviert ist, stehen die Variablen je Zeile/ Datensatz unter @@CSV0@@, @@CSV1@@ usw. zur Verfügung. Die Zählung der Spalten beginnt bei 0.

4.6.7.2 Feldtrenn- und Begrenzungsszeichen

Das *Feldtrennzeichen* wird für die Trennung der einzelnen Spalten verwendet. Für gewöhnlich werden hierfür die Zeichen *;* oder *,* verwendet.

Das *Begrenzungszeichen* wird für verwendet. Für gewöhnlich werden hierfür die Zeichen " oder ' verwendet.

4.6.7.3 Datenkodierung

Um alle Sonderzeichen korrekt verwenden zu können muss die passende *Dateikodierung* ausgewählt werden. Zur Verfügung stehen:
„Default“ (ANSI)

4.6.8 None

Es stehen keine ausgelesenen Werte zur Verfügung, lediglich die Standardvariablen (siehe Immer verfügbare Variablen) sind verfügbar.

4.7 Daten-Lookup

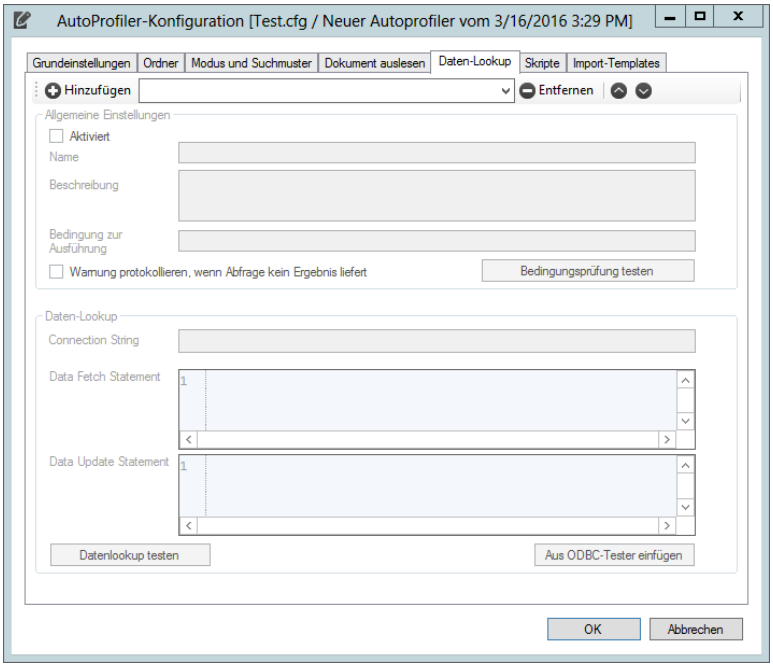


Abb. 4-10: Daten-Lookup

Zwischen dem Auslesen der Informationen aus den bereitgestellten Belegen und deren Archivierung gliedert sich optional die Komponente Daten-Lookup ein. Mit Hilfe des Daten-Lookups ist es möglich, zusätzliche Daten zu einem Dokument aus einer externen (ODBC-) Datenquelle zu extrahieren und für die Archivierung zu verwenden. Dieser Verarbeitungsschritt ist unabhängig von der Herkunft bzw. dem verwendeten DocContentParser (vgl. Kapitel Dokumenten Auslesen).

Es können mehrere Daten-Lookup-Vorgänge pro AutoProfiler durchgeführt werden.

Über den Button Hinzufügen können Sie eine zusätzlichen Daten-Lookup hinzufügen. Über die Buttons können Sie die Reihenfolge der Daten-Lookups ändern und über den Button Entfernen einen bestehenden Daten-Lookup löschen.

Reihenfolge von Daten-Lookups

Daten-Lookups werden immer in der festgelegten Reihenfolge nacheinander ausgeführt und können aufeinander aufbauen. Somit kann das Resultat des ersten Daten-Lookups als Ausgansbasis für den nächsten Lookup dienen.

4.7.1 Aktiviert

Über die Checkbox *Aktiviert* kann ein Daten-Lookup aktiviert oder deaktiviert werden.

4.7.2 Name

Name des Daten-Lookups. Dieses Feld ist rein informativ und hat keinen Einfluss auf die Funktion der Konfiguration. Ein sinnvoll und sprechend vergebener Name kann jedoch die Auswertbarkeit von Log-Dateien stark verbessern.

4.7.3 Beschreibung

Beschreibung des Daten-Lookup. Dieses Feld ist rein informativ und hat keinen Einfluss auf die Funktion der Konfiguration.

4.7.4 Bedingung zur Ausführung

Optionales Attribut: Bestimmt durch Auswertung des angegebenen Ausdrucks zu Wahr bzw. Falsch, ob der Daten-Lookup ausgeführt wird oder nicht. Für die Angabe des Ausdrucks gelten die im Abschnitt Auswertung von Prüfbedingungen angegebenen Bedingungen und Regeln. Fehlt die Bedingung, so wird der Daten-Lookup für jedes Dokument dieses AutoProfilers ausgeführt.

Vergleichswerte in SQL-Abfragen

Sie sollten immer prüfen ob Ihr Vergleichswert in der SQL-Abfrage im Daten-Lookup (siehe Daten-Lookup) verfügbar ist. Dieses können Sie einfach mit einer Überprüfung auf ungleich NULL (`@@Barcode@@!=null`) durchführen. Für weitere Informationen zur Bedingungen siehe Kapitel Auswertung von Prüfbedingungen.

4.7.5 Warnung protokollieren

Durch die Option *Warnung protokollieren, wenn eine Abfrage kein Ergebnis liefert* kann die Logausgabe beeinflusst werden. So ist in der Übersicht des Loggings leichter festzustellen, ob es ein Problem bei der Datenbankabfrage gegeben hat. Deaktivieren Sie diese Option, wenn es durchaus sein kann, dass die Abfrage kein Ergebnis zurück liefert.

4.7.6 ConnectionString

Pflichtattribut: Der Connection String gibt an, wie sich das Importprogramm auf die ODBC-Datenquelle verbindet. Der Connection String muss gültig im Sinne der Microsoft ODBC-Konfiguration sein bzw. im Kontext des Benutzerkontos, mit welchem der docuvita Importdienst ausgeführt wird, entsprechende Zugriffsrechte auf die Datenquelle gewähren.

Um Konfigurationsaufwand zu sparen erstellen Sie einen zentralen

Konfigurationswert (siehe Konfigurationswerte) und verwenden ihn hier (z.B. ##CONNECTIONSTRING##).

Beispiel: Microsoft Access-Datenbank als ODBC-Datenquelle

```
connectionString="Driver={Microsoft Access Driver
(*.mdb)};Dbq=C:\Ordner\DemoAccess.mdb;Uid=Admin;Pwd=;"
```

Mit Hilfe des angegebenen Connection String verbindet sich das Importprogramm auf die Access-Datenbank DemoAccess.mdb im Verzeichnis C:\Ordner.

Weitere Informationen zu Connection Strings zu verschiedensten Datenbanksystemen finden Sie unter http://www.connectionstrings.org

Anstelle eines Connection Strings können Sie auch auf dem System eine System-DSN anlegen über die dann auf die Zieldatenbank zugegriffen werden kann (z.B. connectionString =“dsn=datenbank“).

4.7.7 Daten-Lookup

Die Feldwerte für *dataFetchStatement* und *dataUpdateStatement* sind jeweils optional. Wird also jeweils ein Feldwert angegeben, wird der Vorgang ausgeführt. Anderenfalls wird der Vorgang übersprungen.

4.7.7.1 dataFetchStatement

Tragen Sie hier eine SQL-Abfrage ein, mit dem Sie die gewünschten Daten aus der Datenquelle (bzw. Datenbank) nachschlagen wollen.

Als Rückgabe der SQL-Abfrage wird eine Zeile mit beliebig vielen Spalten erwartet. Jede Spalte kann anhand ihres Spaltentitels (Feldname) im dritten Verarbeitungsschritt des AutoProfilers genutzt werden (`@@Spaltenname@@`).

Beispiel: Nachschlagen von Lieferscheinnummern

```
dataFetchStatement="SELECT * FROM [Lieferschein]
WHERE LieferscheinNummer='@@Barcode@@'"
```

Mit Hilfe des angegebenen *dataFetchStatement* holt das Importprogramm alle Informationen aus der Tabelle Lieferschein zu der zugehörigen Lieferscheinnummer, die als Barcode auf den Beleg aufgedruckt und erkannt wurde.

4.7.7.2 dataUpdateStatement

Tragen Sie hier ein SQL-Update-Statement ein, mit dem Sie eine ODBC-Datenquelle aktualisieren können, wenn ein bestimmter Wert im Dokument erkannt wurde.

Beispiel: Eintragen gefundener Lieferscheine

```
dataUpdateStatement="UPDATE [Lieferschein]
SET LieferscheinArchiviert='T',
LieferscheinArchivdatum=GetDate()
WHERE LieferscheinNummer='@@Barcode@@'"
```

Mit Hilfe des angegebenen dataUpdateStatement trägt das Importprogramm in die Datenquelle ein, dass der bereitgestellte Lieferschein archiviert wurde. Ebenfalls wird der Archivierungszeitpunkt in die Datenbank zurückgeschrieben.

4.8 Skripte

Im Rahmen der Ausführung von AutoProfilern ist docuvita in der Lage, installationsspezifische Skripte in der Programmiersprache IronPython auszuführen. Die technischen Details sowie Information zur Verwendung von Skripten können Sie im Bereich Ausführung eines Skripts nachlesen. In diesem Abschnitt wird nur die Konfiguration des „Skripte“ Reiters thematisiert.

Ausführen von installationsspezifischen Skripten

Zusätzlich zum Nachschlagen von Daten in ODBC-Datenquellen ist es mit Hilfe einer benutzerdefinierten Skriptausführung möglich, speziellere Anforderungen beim Dokumentimport zu erfüllen als mit reinen Datenqueries und –updates möglich. Um diese Möglichkeiten zu nutzen, sind Skripte in der Programmiersprache IronPython notwendig. Mit Hilfe dieser Skripte kann auf die Funktionalität des Microsoft .NET Framework zugegriffen werden. IronPython unterstützt zusätzlich die grundlegende Basisfunktionalität und Syntax der Programmiersprache Python.

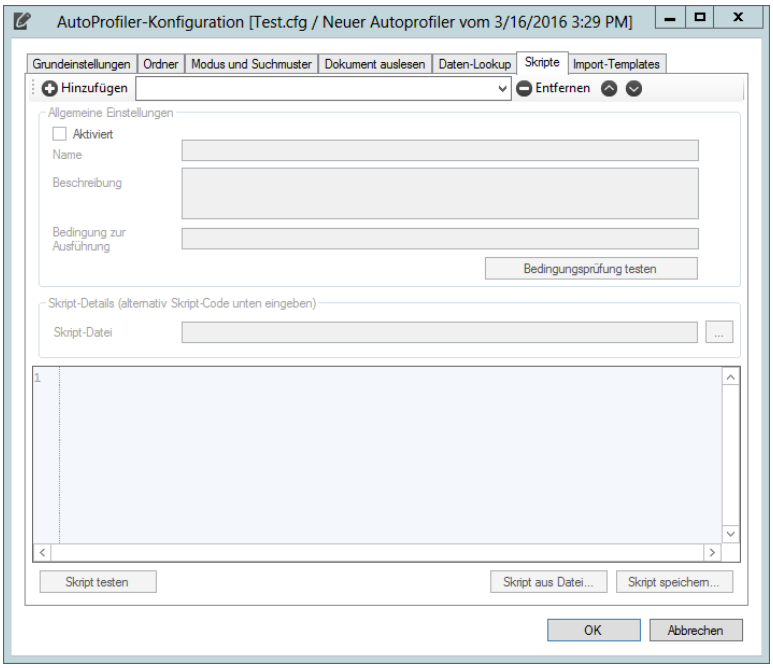
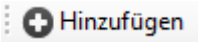
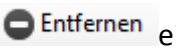


Abb. 4-11: Skripte

Es können mehrere Skripte pro AutoProfiler ausgeführt werden.

Über den Button  können Sie ein zusätzliches Skript hinzufügen. Über die Buttons  können Sie die Reihenfolge der Skripte ändern und über den Button  ein bestehendes Skript löschen.

Reihenfolge von Skripten

Skripte werden immer in der festgelegten Reihenfolge nacheinander ausgeführt und können aufeinander aufbauen. Somit können neue, aus dem ersten Skript übergebene Variablen als Ausgansbasis für das nächste Skript dienen.

4.8.1 Aktiviert

Über die Checkbox *Aktiviert* kann ein Skript aktiviert oder deaktiviert werden.

4.8.2 Name

Name des Skripts. Dieses Feld ist rein informativ und hat keinen Einfluss auf die Funktion der Konfiguration. Ein sinnvoll und sprechend vergebener Name kann jedoch die Auswertbarkeit von Log-Dateien stark verbessern.

4.8.3 Beschreibung

Beschreibung des Skripts. Dieses Feld ist rein informativ und hat keinen Einfluss auf die Funktion der Konfiguration.

4.8.4 Bedingung zur Ausführung

Optionales Attribut: Bestimmt durch Auswertung des angegebenen Ausdrucks zu Wahr bzw. Falsch, ob das Skript ausgeführt wird oder nicht. Für die Angabe des Ausdrucks gelten die im Abschnitt Auswertung von Prüfbedingungen angegebenen Bedingungen und Regeln. Fehlt die Bedingung, so wird das Skript für jedes Dokument dieses AutoProfilers ausgeführt.

Variablen vor Ausführung des Skriptes prüfen

Sie sollten vor der Ausführung eines Skripts immer prüfen ob im Skript verwendete Variablen verfügbar sind. Dieses können Sie einfach mit einer Überprüfung auf ungleich NULL (`@@Barcode@@!=null`) durchführen. Für weitere Informationen zur Bedingungen siehe Kapitel Auswertung von Prüfbedingungen.

4.8.5 Skript-Code / Skript-Datei

Der Skript-Code kann direkt in das entsprechende Feld in der Maske eingefügt werden. Alternativ kann in dem Feld *Skript Datei* der Pfad zu einer Skript-Datei angegeben werden. Bitte beachten Sie dabei, dass die Skriptdatei für das Dienstkonto des AutoProfilers erreichbar sein muss. Relative Verzeichnispfade können ausgehend vom Programmverzeichnis des AutoProfiler aus angegeben werden. Für die Erstellung einer Skript Datei siehe Kapitel Ausführung eines Skripts.

4.9 Import-Templates

Der letzte Verarbeitungsschritt des AutoProfilers besteht darin, die aus den Dokumenten bzw. optional über Daten-Lookups gewonnenen Informationen in einen XML-Quellabschnitt einzutragen. Anschließend erfolgt automatisch der Upload an den docuvita.Server zum Import des Dokumentes.

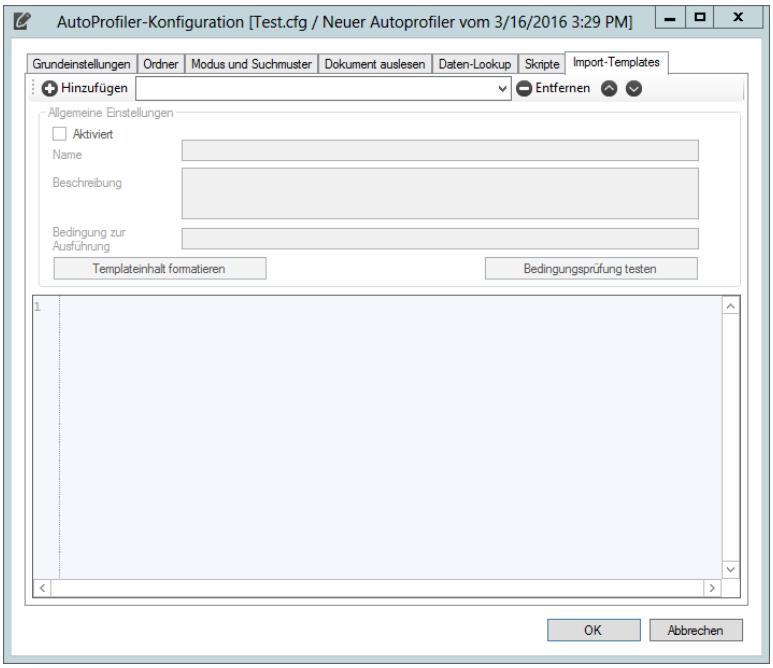
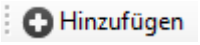


Abb. 4-12: Import Templates

Es können mehrere Import-Templates pro AutoProfiler verwendet werden.

Über den Button 

können Sie ein zusätzliches Import-Template hinzufügen. Über die Buttons  können Sie die Reihenfolge der Import-Templates ändern und über den Button  ein bestehendes Import-Template löschen.

4.9.1 Aktiviert

Über die Checkbox *Aktiviert* kann ein Import-Template aktiviert oder deaktiviert werden.

4.9.2 Name

Name des Import-Templates. Dieses Feld ist rein informativ und hat keinen Einfluss auf die Funktion der Konfiguration. Ein sinnvoll und sprechend vergebener Name kann jedoch die Auswertbarkeit von Log-Dateien stark verbessern.

4.9.3 Beschreibung

Beschreibung des Import-Templates. Dieses Feld ist rein informativ und hat keinen Einfluss auf die Funktion der Konfiguration.

4.9.4 Bedingung zur Ausführung

Optionales Attribut: Bestimmt durch Auswertung des angegebenen Ausdrucks zu Wahr bzw. Falsch, ob das Import-Template verwendet wird oder nicht. Für die Angabe des Ausdrucks gelten die im Abschnitt Auswertung von Prüfbedingungen angegebenen Bedingungen und Regeln. Fehlt die Bedingung, so wird das Import-Template verwendet für jedes Dokument dieses

AutoProfilers verwendet. Wenn ein Template auf ein Dokument Anwendung findet gilt das Dokument als verarbeitet und es wird nicht auf das Vorhandensein weiterer Templates geprüft.

Achtung

Sie sollten vor der Verwendung eines Templates immer prüfen ob sämtliche Variablen für das Schlüsselfeld (siehe Definition der zu importierenden Objekttypen / Schlüsselfelder) und ggf. vorhandene Pflichtfelder gefüllt sind. Dieses können Sie einfach mit einer Überprüfung auf ungleich NULL (`@@Barcode@@!=null`) durchführen. Für weitere Informationen zur Bedingungen siehe Kapitel Auswertung von Prüfbedingungen.

4.9.5 Template

Je nach Anforderung bzw. Komplexität gibt es drei Möglichkeiten, den XML-Quellabschnitt zu konfigurieren:

- Einfacher Fall: nur ein XML-Quellabschnitt, verschachtelte *object*-Elemente

```
<!-- ... XML-Quellabschnitt mit Platzhaltern -->
<object type="..." obj.Name="@@...@@" >
     <object type="..." obj.Name="@@...@@" ... />
     <.../>
</object>
```

Die Verwendung der Platzhalter erfolgt in der Form `@@Feldname@@`. Der Feldname kann dabei ein Rückgabewert des verwendeten DocContentParsers (z.B. bei Barcode), ein Wert aus dem ausgelesenen Dokument oder eine immer verfügbare Variable sein (siehe Variablen).

- Fallunterscheidungen: über die Reihenfolge der Templates und unter Zuhilfenahme der Bedingung zur Ausführung können je nach Anforderung Verzweigung und Fallunterscheidung realisiert werden. So kann je nach Inhalt einer Variablen ein definiertes Template gültig sein.

  Bei der Verarbeitung der Templates wird diejenige Vorlage verwendet, bei der zuerst die Prüfbedingung in der Bedingung zur Ausführung zutrifft. Zur Auswertung der Prüfbedingungen vgl. Kapitel Auswertung von Prüfbedingungen.

- Angabe eines externen Dispatchers

	`<!-- ... XML-Quellabchnitt mit Platzhaltern -->` `<dispatcher name="DispatcherKlasse" parameter="xyz" />`
	Der zu verwendende Dispatcher wird in Form seines Klassennamens in das Attribut *name* des Elements *dispatcher* eingetragen. Der Parameter wird an den Dispatcher übergeben. Dabei muss sichergestellt sein, dass der entsprechende Dispatcher (DLL-Datei) in den Unterordner Dispatchers des Installationsverzeichnisses abgelegt wurde. Die vorhandenen Dispatcher und deren ggf. vorhandene Konfiguration werden beim Start des Importprogramms eingelesen.

Verwenden und Weiterbearbeiten von ausgelesenen Variablen

Sie können die aus dem zu archivierenden Dokument bzw. aus Data-Lookups ausgelesenen Werte mit Hilfe von Variablen in Ihrer Importstruktur verwenden. Hierzu müssen Sie den Namen der Variablen in doppelten @-Zeichen eingeben (siehe auch Office Variablen und E-Mail Variablen). Beispiele:

`@@Belegnummer@@` wird beim Import durch die jeweils aus dem Feld Belegnummer ausgelesenen Werte ersetzt.

Alternativ können die ausgelesenen Werte auch vor der Verwendung bearbeitet werden. So kann z.B. aus einer Kundenbezeichnung „Müller GmbH 331“ nur die Mandantennummer „331“ extrahiert werden. Dazu wird an die Feldverwendungskennzeichen – getrennt durch zwei Doppelpunkte – ein Suchmuster (regulärer Ausdruck) angehängt (siehe auch Spezial Variablen):

`@@Mandant::\s*\d{3}\s*$@@` wendet das Suchmuster `\s*\d{3}\s*$` auf den Rückgabewert des Feldes Mandant an.
Ist ein Feldwert nicht vorhanden, wird dessen Variable beim Import durch eine leere Zeichenkette ersetzt.

Standardfeld für den Dateinamen

Die Systemvariable `@@Filename@@` wird unabhängig von der Festlegung des Feld-Suchmusters immer für den Dateinamen der zu importierenden Dokumentdatei verwendet. `@@Filename@@` kann im Feld `doc.FileName` angegeben werden:

```
doc.FileName="@@Filename@@"
```

doc.FileName

doc.FileName muss immer angegeben werden, wenn ein Dokument-Objekttyp angelegt oder versioniert werden soll.

Sollen lediglich die Eigenschaftsfelder aktualisiert werden und keine neue Version des Dokumentes erzeugt werden, so muss dieses über den obj.UpdateMode entsprechend entsprechend eingestellt werden (siehe Felder für sämtliche Objekttypen).

Wenn eine Steuerdatei (XML-Datei oder Dateiname einer ansonsten leeren Datei) für die Verarbeitung verwendet wird muss das Attribut doc.Filename nicht mit die Systemvariablen `@@Filename@@` sondern der feste Pfad zur Datei bzw. die aus der Steuerdatei ausgelesene Variable verwendet werden.

Wir nur ein Dateiname und kein Pfad angegeben, so geht der AutoProfiler davon aus, dass sich die Datei im Quell-Verzeichnis des jeweiligen AutoProfilers befindet. Im Fall von Steuerdateien wird folglich vom selben Verzeichnis ausgegangen aus dem die Steuerdatei abgeholt wurde.

Weitere Informationen finden Sie im Kapitel Standard-Felder für Dokument-Objekttypen.

Weitere Standardvariablen finden Sie unter Immer verfügbare Variablen.
Weitere Informationen zu den Systemvariablen finden Sie unter Systemvariablen.
Weitere Informationen zur Ausgestaltung von Templates finden Sie im folgenden Kapitel Aufbau der dvImport-Datei.

Sie möchten das Quell-Dokumente nicht importieren

Definieren Sie ein leeres Template ohne *Bedingung zur Ausführung* als letztes Template und geben dort lediglich `DELETE` an. Die Datei wird in diesem Fall einfach verworfen.

Aufbau der dvImport-Datei

5 Aufbau der dvImport-Datei

Der automatisierte Import von Dokumenten und Informationen in docuvita erfolgt über die Abarbeitung von Importdateien, die Informationen über zu importierende bzw. synchronisierende Daten und die zugehörigen Dokumentdateien beinhalten. Importdateien können von Hand oder externen Programmen erstellt werden, im Normallfall handelt es sich jedoch um automatisch vom docuvita.AutoProfiler erzeugte Dateien. In diesem Fall entspricht eine dvImport-Datei einem im AutoProfiler definierten Import-Template, bei dem die Variablen bereits durch die aus dem Dokument ausgelesenen Werte ersetzt wurden. In diesem Abschnitt wird beschrieben, welchen Anforderungen die Importdatei, also in gewissem Maße auch ein Import-Template, genügen muss. Sowohl dvImport-Dateien, als auch die Import-Templates werden im XML-Format geschrieben.

XML-Konformität

dvImport-Dateien müssen XML konform gestaltet werden. Aus diesem Grund müssen gewisse Sonderzeichen entsprechend ersetzt werden. Hier eine Übersicht:

& → `&`
< → `<`
> → `>`
" → `&quot;`
' → `&apos;`

5.1 Definition der zu importierenden Objekttypen / Schlüsselfelder

Die Definition der Schlüsselfelder geschieht über die Masken *Administration -> Objecttypes* -> Auswahl des *Objekttyps* per Doppelklick-> Pflege der Felder *Updatekeyfileds* und *Updateloctype* im docuvita.Client. Zur Konfiguration der Objekttypen siehe auch Administrationshandbuch docuvita.Server - Customizing.

Ein Objekttyp kann auch über einen aus mehreren Feldern zusammengesetzten Schlüssel verfügen. In diesem Fall trennen Sie die Eigenschaftsfelder im Feld *Updatekeyfileds* mit Hilfe eines Kommas oder Semikolons; bitte nutzen Sie keine Leerzeichen zwischen den einzelnen Eigenschaftsfeldern.

Änderung zur Version 1.9

Der bisher in der dvImport.exe.config definierte Bereich <objecttypes> entfällt. Bei einem Update müssen die Schlüsselfelder zwingend über

den docuvita.Client zentral eingepflegt werden.

Lassen Sie die Werte für Updatekeyfileds und Updateloctype leer, so wird in jedem Fall ein neues Objekt angelegt. Eine Versionierung, ein Aktualisieren der Eigenschaftsfelder oder ein Umhängen des Objektes werden so vermieden.

Schlüsselfelder müssen eindeutig sein

Es darf keine Schlüsselfelder geben, in denen zwei Mal der identische Wert gespeichert ist. Dabei wird zwischen lokal (also in aktuellen der Verzeichnisstruktur) und global (im gesamten System) unterschieden wird.

Ist dieses dennoch der Fall und handelt es sich bei dem doppelten Schlüsselfeld um ein zu importierendes Dokument so wird ein Fehler mit sämtlichen gefundenen Objekten protokolliert und der Import bricht ab. Handelt es sich um einen Ordner oder eine Akte, so wird eine Warnung mit sämtlichen gefundenen Objekten protokolliert, das zu importierende Objekt wird jedoch unter dem ersten gefundenen Objekt angelegt bzw. das erste gefundene Objekt aktualisiert.

5.1.1 Versionierung

Navigieren Sie im docuvita.Client in den Bereich *Administration* → *Objecttypes* und wählen dort den gewünschten Objekttypen aus. Öffnen Sie diesen per Doppelklick zur Bearbeitung. Tragen Sie im Bereich *Import Configuration* das Eigenschaftsfeld (entweder ein selbst angelegtes oder ein im Standard verfügbares – siehe Mögliche Felder im XML-Tag „object“) als Schlüssel in das Feld *Updatekeyfields* ein.

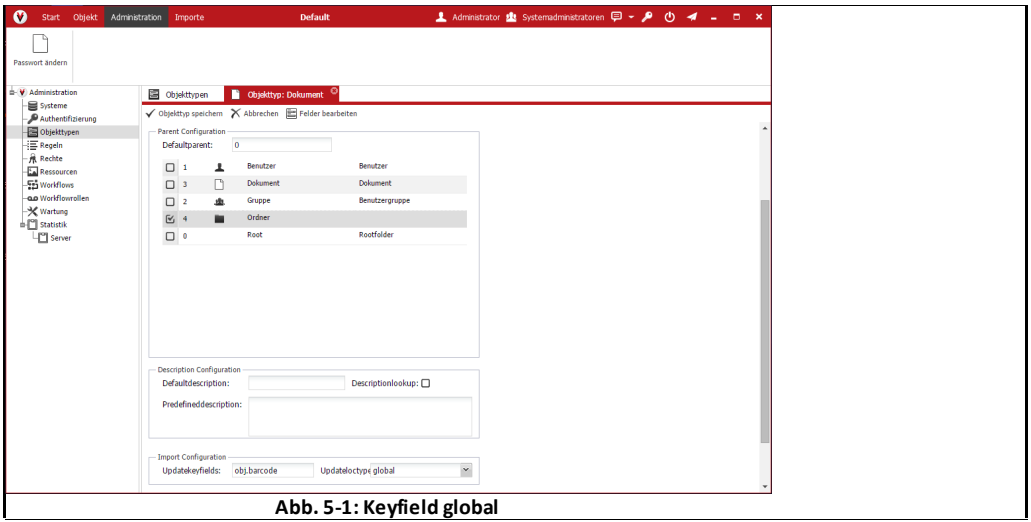


Abb. 5-1: Keyfield global

Wenn bei *Updateloctype* „global“ ausgewählt wird, dann wird im gesamten System auf das Vorhandensein des Schlüsselfeldes geprüft. Sowohl bei manueller Verarbeitung als auch beim automatischen Import wird somit kein doppeltes Anlegen zugelassen.

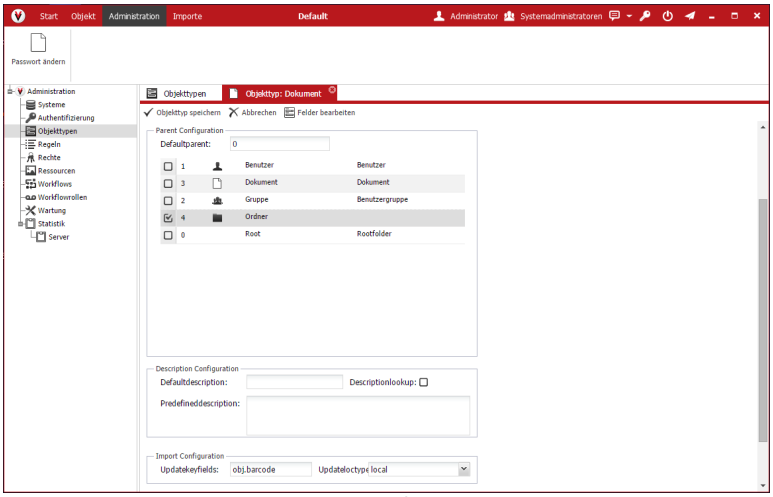


Abb. 5-2: Keyfield local

Wenn bei *Updateloctype* „local“ ausgewählt wird, dann wird in der aktuellen Struktur auf das Vorhandensein des Schlüsselfeldes geprüft. Sowohl bei manueller Verarbeitung als auch beim automatischen Import wird somit kein doppeltes Anlegen zugelassen.

5.2 Rumpf der dvImport-Datei

```
<?xml version="1.0" encoding="utf-8"?>
<import>
  <data>

.. Hier die zu importierenden Objekte und Dokumente
anführen

  </data>
</import>
```

Anstelle von *Hier die zu importierenden Objekte und Dokumente anführen* werden hier die zu importierenden Objekte in `<object />` Notation angegeben.

5.3 Importieren von Objekten

Zu Importierende Objekte müssen in der hier gezeigten XML-Syntax definiert werden. Das XML-Tag eines Objektes muss dabei immer mindestens das Feld „type“ sowie alle Pflichtfeldangaben (z.B. „obj.name“) enthalten. Soll ein

Objekt lediglich aktualisiert werden, so müssen nur die zu ändernden Attribute angegeben werden. Dokumente benötigen darüber hinaus das Feld „doc.Filename“. Weitere mögliche Standardfelder entnehmen Sie bitte der unten Tabelle in Kapitel Mögliche Felder im XML-Tag „object“, es können außerdem sämtliche selbst definierte Felder genutzt werden.

```
   <object type="Kundenakte" obj.Name="Mustermann GmbH" /
>
   <object type="Kundenakte" obj.Name="Frohsinn AG" />
   ...
```

Der Teil vor einer Zuweisung *xyz*="…" entspricht, bis auf den Sonderfall *type, doc.Filename, docDeleteFile* und *versioncomment* immer einem Eigenschaftsfeld des zu importierenden Objekttyps. Der Teil hinter der Zuweisung *xyz*=**“…“** kann entweder ein Freitext sein oder es handelt sich um eine Variable bzw. deren Inhalt. Im Fall des Import-Templates wird an dieser Stelle die automatisch zu ersetzende Variable, also z.B. `@@Kundennummer@@` eingetragen. In der dvImport-Datei ist dieses immer ein fester Wert. Wenn die dvImport-Datei automatisch aus einem Import-Templates erstellt wurde finden Sie hier den Wert der Variablen wieder (sofern ein Wert für die Variable gefunden wurde, ansonsten ist der Wert leer). Ein leerer Wert überschreibt keinen bereits gefüllten Wert eines Eigenschaftsfeldes eines bestehenden Objektes.

5.4 Mögliche Felder im XML-Tag „object“

5.4.1 Felder für sämtliche Objekttypen

	Feld	Beschreibung
	Type **obj.Objecttype** (erforderlich)	Gibt den Typ des zu importierenden Objekts an. Verwenden Sie hier den Namen (nicht den internen Namen), den Sie in der Objekttypkonfiguration für den gewünschten Objekttyp festgelegt haben. **Beispiel** `<object type="Kundenakte"… />` Erzeugt ein Objekt vom Typ Kundenakte.
	obj.Name (erforderlich)	Legt den Namen fest, den das zu importierende Objekt erhalten soll. **Beispiel** `<object … obj.name="Johann Müller`

		`GmbH" … />`
	obj.ID (optional)	Mit Hilfe von obj.ID können Sie direkt auf ein bestehendes Objekt (Dokument, Akte, etc.) zugreifen und dieses aktualisieren. Das Feld kann z.B. für die Aktualisierung von Dokumenten aus Fremdsystemen heraus verwendet werden. Das Attribut obj.ID kann auch in einer Verschachtelung von Objekten verwendet werden, um bestehende Objekte/ Dokumente in eine neue Struktur umzuhängen, oder in Verbindung mit /type zu einem Wechsel des Objekttyps bestehender Objekte eingesetzt werden. obj.ID ist, wenn angegeben, immer das vorrangige/eindeutige Schlüsselfeld. Ist für einen Objekttypen zusätzlich ein anderes Schlüsselfeld definiert wird dieses zusätzlich auf Eindeutigkeit geprüft, jedoch nicht für eine Identifizierung genutzt.
	obj.UpdateMode (optional, nur bei erneutem Import / Versionierung relevant)	Folgende Werte sind möglich: `UpdateProperties` → aktualisiert die Objekteigenschaften `UpdateParent` → setzt die obj.Parentid neu, kann zum Umhängen eines bestehenden Objektes genutzt werden `UpdateDocNewVersion` → importiert die Dokumentdatei als neue Version `StartWorkflowOnUpdate` → startet den Workflow auch bei Objektaktualisierungen / Versionierungen `NoUpdate` → startet führt kein Update aus. Dieses kann z.B. beim setzen von Verknüpfungen sinnvoll sein (es wird nur ein Objekt gesucht) Es können mehrere Werte kombiniert werden (Komma-getrennt, ohne Leerzeichen). So kann ein Dokument z.B. umgehängt und als neue Version importiert

		werden, ohne das die Eigenschaften des Dokumentes aktualisiert/überschrieben werden. **Beispiel** `<object … obj.UpdateMode="UpdateParent, UpdateDocNewVersion" … />` Ist kein Wert angegeben, so werden sämtliche Werte als gesetzt angesehen, es werden also Eigenschaften aktualisiert, Dokumenten umgehängt und versioniert sowie der Workkflow gestartet.
	obj.Parentobject / obj.Parentid (optional)	ID des Parentobjects unter dem dieses Element erstellt (oder verschoben) werden soll. Wenn obj.Parentobject innerhalb einer Verschachtelung angegeben wird das Element dennoch in der Struktur angelegt, der Wert für obj.Parentobject verworfen und eine Warnung ins Log ausgegeben.
	obj.Description (optional)	Hier geben Sie die Beschreibung an, die das zu importierende Objekt erhalten soll. **Beispiel** `<object … obj.description="ist ein guter Kunde" … />` Trägt beim Import „ist ein guter Kunde“ in das Beschreibungsfeld ein.
	obj.usercreated (optional)	Über obj.usercreated kann der Benutzername angegeben werden, der für den Import in der Ersteller-Angabe in dem Objekt eingetragen wird. Es muss ein entsprechendes docuvita Benutzerkonto geben (kein Import-Benutzer). Wenn obj.usercreated im ersten Objekt des Importtemplates angegeben wird, dann wird er auch für alle weiteren Objekte in diesem Template verwendet.
	obj.Link (optional)	Über diesen Parameter kann nach dem eigentlichen Dokumentenimport noch ein Link zu dem Objekt an einer anderen Stelle / in einer anderen Struktur angelegt werden.

		Beispiel `<object … obj.Link="True" … />` Siehe auch Erzeugen von Verknüpfungen.
	[Feldname] (optional)	Sie können benutzerdefinierte Felder mit Informationen füllen, indem Sie den Feldnamen im Importsatz angeben. Verwenden Sie hier den Feldnamen (nicht die „Eingabeaufforderung“), den Sie für das gewünschte Feld festgelegt haben. Der Feldname darf keine Leerzeichen enthalten. **Beispiel** `<object … meinFeld="Abcde" deinFeld="12345" … />` Füllt die Felder *meinFeld* und *deinFeld* mit den entsprechenden Werten.
	loctype **locvalue** (optional)	Mit Hilfe dieser Optionen ist es möglich, Objekte unterhalb von bestimmten bestehenden Objekten anzulegen. Dabei führt der Import eine Suche nach einem bestimmten Objekt durch, unterhalb dessen das aktuelle Objekt angelegt wird. Dabei legen Sie den Typ (*loctype*) und den Suchwert (*locvalue*) fest. In welchem Feld der in locvalue angegebene Suchwert gesucht wird, bestimmt sich nach dem Schlüsselfeld des jeweiligen Objekttypen. Ein Beispiel zum Anlegen eines Objekts unterhalb einer bestehenden Kundenakte finden Sie im Kapitel 5.5 Importieren von Hierarchien. Falls obj.Parentobject (siehe oben) angegeben ist geht dieser Wert vor.
	preImportScript (optional)	Script, welches direkt vor der Verarbeitung des entsprechenden Objekts/Dokuments ausgeführt wird. Mit Hilfe dieses Skripts ist z.B. eine kundenspezifische Anpassung an

		den Dokumentdateien durchführbar. Bei der Ausführung wird anhand der Endung der Scriptdatei automatisch in IronPython (Endung .py) oder C# (Endung .cs) Scripts unterschieden. Im Rahmen des preImportScripts werden Variablen zur Verfügung gestellt, mit denen ein Zugriff auf die Dokumentdatei und die bereits ermittelten Daten zum Objekt/ Dokument möglich ist. Das Script wird vom docuvita.Server ausgeführt. Das Dienstkonto muss Zugriff auf das Skript (Pfad im Dateisystem) haben.
	postImportScript (optional)	Script, welches direkt nach der Verarbeitung des entsprechenden Objekts/Dokuments ausgeführt wird. Bei der Ausführung wird anhand der Endung der Scriptdatei automatisch in IronPython (Endung .py) oder C# (Endung .cs) Scripts unterschieden. Zusätzlich zu den im preImportScript vorhandenen Daten kann im postImportScript auch auf die Objekt-ID des angelegten Objekts sowie auf die Versions-ID des importierten Dokuments (sofern es sich beim aktuell durchgeführten Import um einen Dokumentimport und nicht um ein Strukturelement handelt) zugegriffen werden. Das Script wird vom docuvita.Server ausgeführt. Das Dienstkonto muss Zugriff auf das Skript (Pfad im Dateisystem) haben.
	workflowName (optional) und **workflowValues** (optional)	Workflows können direkt aus dem AutoProfiler heraus bzw. beim Import gestartet werden. Zum automatischen Starten des Workflows muss folgender Abschnitt in das Template des zu importierenden Objekts eingefügt werden: `workflowName="Name_des_Workflows"`

		optional: `workflowValues="Feld1:Wert1;Feld2:Wert2;usw"` In workflowValues ist die Feld:Wert-Trennung auch per = möglich, und die Gruppentrennung statt ; auch per , -> also Feld1=Wert1,Feld2=Wert2,usw.

5.4.2 Standard-Felder für Dokument-Objekttypen

	Feld	Beschreibung

	doc.FileName
	(erforderlich beim Import von Dokumenten)
		Gibt den Pfad zu der zu importierenden Datei an, wenn es sich bei dem anzulegenden Objekt um ein Dokument handelt.
		
		Beispiel
		
		`<object … doc.FileName="C:\import\test.tif" … />`
		
		Veranlasst, dass das Dokument *C:\import\test.tif* importiert wird.
		
		Bei der Nachverschlagwortung von Dokumenten mit zusätzlichen/anderen Eigenschaften muss das Attribut nicht angegeben werden.
		
		Wird `doc.FileName` im Template des AutoProfilers verwendet bietet es sich an die Variable `@@Filename@@` zu verwenden. `@@Filename@@` enthält immer den Pfad zur Quelldatei und wird zur Laufzeit des AutoProfilers automatisch ersetzt:
		`<object … doc.FileName="@@Filename@@" … />`

	doc.FileDelete	Löscht die eigentliche Quelldatei und kann nur verwendet werden, wenn dass Feld `doc.FileName` nicht verwendet wird oder ein anderer Wert oder Variable als `@@Filename@@` in `doc.FileName` verwendet wird. `<object … doc.FileDelete="@@FileDelete@@" … />`

		Dieses Feld kann nur in einem Template des docuvita.AutoProfilers verwendet werden und führt seine Funktion zur Laufzeit des AutoProfilers aus. Eine Verwendung durch eine extern erstellte dvImport-Datei ist nicht vorgesehen und wird auch nicht benötigt, da in diesem Fall immer direkt die korrekte Datei in doc.FileName angegeben werden kann. Weitere Informationen finden Sie auch im Kapitel Systemvariablen.
	ver.Comment (optional)	Bietet bei Dokumentimporten die Möglichkeit, einen Versionskommentar für die neu angelegte Dokumentversion anzugeben.
	obj.fileextension (optional)	Wird im Normalfall automatisch ausgewertet und sollte nicht verändert werden.
	obj.Documenttype (optional)	Art des Dokumentes (z.B. Rechnung oder Lieferschein)
	obj.Barcode (optional)	Barcode (bei Barcodescanszenarien als Schlüsselfeld empfohlen)
	obj.Voucherdate (optional)	Belegdatum
	obj.Vouchertype (optional)	Belegart (z.B. Eingangs oder Ausgangsrechnung)
	obj.Vouchernumber (optional)	Belegnummer (z.B. interne Rechnungsnummer des führenden Systems)
	obj.Externalvouchernumber (optional)	Externe Belegnummer (z.B. Rechnungsnummer des externen Erstellers der Rechnung)
	obj.Transactionkey (optional)	Vorgangsnummer
	obj.Transactiontype (optional)	Vorgangstyp
	obj.Validfrom (optional)	Gültig ab
	obj.Validuntil (optional)	Gültig bis

	obj.Mailfrom (optional)	Absender (E-Mail – wird auch automatisch durch die docuvita.DocumentServices gefüllt)
	obj.Mailto (optional)	Empfänger (E-Mail – wird auch automatisch durch die docuvita.DocumentServices gefüllt)
	obj.Mailcc (optional)	CC-Empfänger (E-Mail – wird auch automatisch durch die docuvita.DocumentServices gefüllt)
	obj.Mailsubject (optional)	Betreff (E-Mail – wird auch automatisch durch die docuvita.DocumentServices gefüllt)
	obj.Mailsenddate (optional)	Versanddatum (E-Mail – wird auch automatisch durch die docuvita.DocumentServices gefüllt)
	obj.Mailmessageid (optional)	MessageID (E-Mail – wird auch automatisch durch die docuvita.DocumentServices gefüllt)

E-Mailinformationen

Die *obj.Email...* Felder werden von den docuvita.DocumentServices bei der Erstellung des Indexes automatisch befüllt; sie müssen also nicht zwangsläufig über das Template im AutoProfiler am E-Mail-Objekt gefüllt werden.

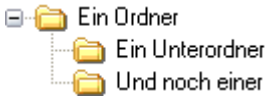
5.5 Importieren von Hierarchien

Das Importprogramm ist in der Lage, Objekte in verschachtelter Form zu importieren und so ganze Strukturen anzulegen.

Beispiel „Anlegen von Hierarchien“

```
<object type="Ordner" obj.Name="Ein Ordner">
  <object type="Ordner" obj.Name="Ein Unterordner" />
  <object type="Ordner" obj.Name="Und noch einer" />
</object>
```

Beachten Sie, dass der XML-Tag des ersten Ordners nach den beiden untergeordneten Ordnern wieder geschlossen wird. Das obige Beispiel erzeugt folgende Hierarchie:



Darüber hinaus können Sie Objekte unterhalb von bestehenden Objekten anlegen, indem Sie Suchbegriffe für das übergeordnete Objekt definieren. So können Sie beispielsweise Dokumente unterhalb von Kundenakten anlegen, wenn die Kundenstammdaten von einem Datenprovider bereitgestellt werden.

Beispiel „Dokument in einem Ordner einer Kundenakte importieren“
In diesem Beispiel wird ein Dokument im Ordner Rechnungen in der Kundenakte mit der Kundennummer 12345 angelegt.

```
 <object type="Ordner" obj.Name="Rechnungen"
loctype="Kundenakte" locvalue="12345">

  <object type="Dokument" obj.Name="Rechnung"
Belegnummer="99991" doc.Filename="C:\import\test.tif" />

</object>
```

Kundenakte
Für den Objekttyp *Kundenakte* wurde im docuvita.Server das Feld Kundennummer als global eindeutiges Identifizierungsmerkmal angegeben. Siehe dazu auch Definition der zu importierenden Objekttypen / Schlüsselfelder. Dadurch kann die korrekte Kundenakte gefunden werden.

Ordner
Für den Objekttyp *Ordner* wurde im docuvita.Server der Objektname als lokal eindeutiges Identifizierungsmerkmal angegeben

Belegnummer
Für den Objekttyp „Dokument“ wurde im docuvita.Server das Feld Belegnummer als lokal/gloabl (je nach Anforderung) eindeutiges Identifizierungsmerkmal angegeben. Wenn bereits ein Dokument mit dieser Belegnummer existiert, wird eine neue Dokumentversion angelegt (bei global wird es zusätzlich in die neue Struktur verschoben). Andernfalls wird ein neues Dokument angelegt.

<object type="Ordner" loctype="Kundenakte" locvalue="12345" obj.Name="Rechnungen">
Dieser Schlüssel legt ein neues Objekt vom Typ Ordner an (`type="Ordner"`). Dieser wird unterhalb der Kundenakte (`loctype="Kundenakte"`) mit dem Suchwert 12345 (`locvalue="12345"`) angelegt. Dass es sich bei dem Suchwert um die Kundennummer handelt, wurde im docuvita.Server festgelegt. Der Import sucht nur unterhalb der angegeben Kundenakte (`loctype="local"`) nach einem Ordner Namens

„Rechnungen“ (`obj.Name="Rechnungen"`). Sollte dieser Ordner dort noch nicht existieren, wird er angelegt.

<object type="Dokument" obj.Name="Rechnung" Belegnummer="99991" doc.Filename="C:\import\test.tif" />
Dieser Schlüssel legt ein neues Dokument oder eine neue Dokumentenversion unterhalb des Ordners Rechnungen an.

5.6 Erzeugen von Verknüpfungen

Um einen Verknüpfung zu einem soeben angelegten Objekt zu setzen, muss in dem Objektdas Attribut: `obj.Link="True"` angegeben werden.

Im Anschluss an das zu importierende Objekt wird direkt eine weitere, ganz normale Struktur angelegt und der Link mit dem Attribut: <link/> an beliebiger Stelle eingefügt.

Beispiel eines Import-Templates mit Anlage einer Verknüpfung:

```
<template0>
  <object type="Beleg"
    obj.ParentID="20"
    obj.Name="Testdokument"
    doc.Filename="@@Filename@@"
    ver.Comment="Automatischer Belegimport"
    obj.Link="True"  />

  <object type="Ordner"
    obj.Name="Belege"
    loctype="Kunde"
    locvalue="@@kundennummer@@" >
    <link/>
  </object>
</template0>
```

Skripte

6 Skripte

Im Rahmen der Ausführung von AutoProfilern sowie jeweils vor und nach dem Import von Strukturelementen und Dokumenten ist docuvita in der Lage, installationsspezifische Skripte in der Programmiersprache IronPython bzw. C# auszuführen. Zusätzlich werden für die Logik im docuvita Server auch .NET Assemblies als Plugins unterstützt. In diesem Abschnitt werden die gemeinsamen Grundlagen und Anforderungen für diese Funktionalität dargestellt.

6.1 Überblick

Mit Hilfe der in den AutoProfiler sowie docuvita.Server integrierten Skriptfunktionalität kann eine installationsspezifische Logik im Rahmen des Imports von Strukturelementen und Dokumenten ausgeführt werden. Dabei sind während der Ausführung eines AutoProfilers beliebig viele Skripte definierbar, die jeweils abhängig von der ggf. festgelegten Bedingungsprüfung starten oder übersprungen werden.

	Skript-Art	**Programmier-sprache**	**Konfiguration**
	AutoProfiler-Skript (AutoProfiler)	IronPython	Codeblock oder Skriptdatei im docuvita AutoProfiler-Konfigurationsprogramm angeben Mehrere Skripte möglich
	preImportScript (docuvita Server)	IronPython	Angabe der Skriptdatei über das Import-Template als XML-Attribut: preImportScript="Pfad-Zur-Skriptdatei.py"
		C#	Angabe der Skriptdatei über das Import-Template als XML-Attribut: preImportScript="Pfad-Zur-Skriptdatei.cs"
		.NET Assembly	• .NET Assembly erstellen, die IBatchImportScript aus der docuvita.Contract.DLL implementiert • Name der Assembly: <Hersteller>.BatchImportScript.<Titel>.dll • DLL muss beim Start des docuvita Servers im Ordner "plugins" liegen • Aufruf per XML-Attribut: postImportScript="KlassennameAusAssembly"
	postImportScript	IronPython	Angabe der Skriptdatei über das Import-Template als XML-Attribut:

	(docuvita Server)		postImportScript="Pfad-Zur-Skriptdatei.py"
		C#	Angabe der Skriptdatei über das Import-Template als XML-Attribut: postImportScript="Pfad-Zur-Skriptdatei.cs"
		.NET Assembly	• .NET Assembly erstellen, die IBatchImportScript aus der docuvita.Contract.DLL implementiert • Name der Assembly: <Hersteller>.BatchImportScript.<Titel>.dll • DLL muss beim Start des docuvita Servers im Ordner "plugins" liegen • Aufruf per XML-Attribut: postImportScript="KlassennameAusAssembly"

Bei der Verwendung von Skriptdateien ist sicherzustellen, dass der jeweilige Aufrufkontext (also der docuvita.AutoProfiler bzw. das ausführende Dienstkonto oder der docuvita.Server und das ausführende Dienstkonto) mindestens Lesezugriff auf die Skriptdateien haben.

Abhängig vom Kontext der Ausführung stehen im Skript unterschiedliche Variablen zur Verfügung. So kann z.B. ein Skript im Rahmen des AutoProfilers die Verschlagwortungswerte zu einem Dokument ergänzen. Ein Skript nach dem Import eines Dokuments kann beispielsweise die Dokument-ID aus dem docuvita-System an ein anderes System melden.

6.2 Programmiersprache IronPython

Als Programmiersprache für die Ausführung von Skripten kann IronPython genutzt werden. Die Skripte werden im Microsoft .NET Framework von der Dynamic Language Runtime ausgeführt. Weitere Informationen zu dieser Programmiersprache finden Sie auf der zugehörigen Internetseite unter http://www.ironpython.org.

IronPython leitet sich grundsätzlich von der Programmiersprache Python ab und ist in das Microsoft .NET Framework integriert. Daher sind der Grundablauf und die Syntax der Sprache (Befehle/Schlüsselwörter, Ablaufkontrolle, Variablen, Blockbehandlung) analog zu Python, während IronPython den Zugriff auf die Elemente des .NET Framework ermöglicht.

6.3 Programmiersprache C#

Als Programmiersprache für die Ausführung von Skripten während des Imports am docuvita.Servers kann alternativ auch C# als Skriptsprache genutzt werden.

Dabei kann die volle Funktionalität des Microsoft .NET Framework verwendet werden.

6.4 Ausführung eines Skripts

6.4.1 Skript im Rahmen des AutoProfilers

Soll ein installationsspezifisches Skript im Rahmen der Verarbeitung eines AutoProfilers ausgeführt werden, so erfolgt die Ausführung nach den Datenlookups und vor der Bestimmung des Import-Templates. Im AutoProfiler-Konfigurationsprogramm können mehrere Skripte mit entsprechenden Ausführungsbedingungen hinterlegt werden.

Jedes aufgerufene Skript erhält eine Reihe von Variablen, mit der der Bezug auf das aktuell in der Verarbeitung befindliche Objekt/Dokument sowie die Verfügbarkeit aller im Rahmen der bisherigen Ausführung des AutoProfilers ermittelten Verschlagwortungsdaten hergestellt wird. Zusätzlich ist der Zugriff auf die Logging-Funktionalität des AutoProfilers möglich.

Beispiel: Rumpfskript für die Ausführung im AutoProfiler

```
import clr
from System import Convert
from log4net import *
from System.Collections import Hashtable

# Aufrufparameter
# "log":             log4net ILog-Objekt zum Schreiben
#                    von Logeintraegen
# "fieldValues":     Hashtable mit Feldern und Werten
#                    zum aktuellen Objekt
# "docFilename":     string mit Pfad zur aktuellen
#                    Dokumentdatei, oder null, falls
keine
#                    Dateioperation vorliegt
# "sourceFilename":  string mit Pfad zur Quelldatei der
aktuellen
#                    Dokumentdatei, oder null, falls
keine
#                    Dateioperation vorliegt
#
# Rueckgabewert
# "fieldValues": kann zurueckgegeben werden, um die
Werte
# beim Import zu veraendern

#####################################################
### MAIN SCRIPT
#####################################################
log.Info("Hello World")
```

Im Beispielskript werden zunächst einige Deklarationszeilen abgearbeitet.

Sofern die Nutzung weiterer Klassen des Microsoft.NET Framework gewünscht wird, können diese in den Kopf eingefügt werden.

Im Skript wird dann das übergebene Objekt **log** genutzt, um in das aktuelle Protokoll des XML Imports einen Eintrag zu schreiben. Hierbei handelt es sich um ein ILog-Objekt aus der für die Protokollierung verwendeten **log4net** Assembly.

Beispiel: Wichtige Methoden des log-Objekts

Das log-Objekt unterscheidet beim Schreiben der Protokolleinträge bezüglich des Charakters der Meldung nach Information, Warnung, Fehler oder Debug-Meldung.

```
log.Info("Meldung")
log.InfoFormat("Meldung {0}", meldungsParameter)
log.Warn("Warnung")
log.WarnFormat("Warnung {0}", warnungsParameter)
log.Error("Fehler")
log.ErrorFormat("Fehler {0}", fehlerParameter)
log.Debug("Debugmeldung")
log.DebugFormat("Debugmeldung {0}", debugParameter)
```

6.4.2 Ausführung direkt vor dem Dokument-/Objektimport

Im Unterschied zu den Skripten im AutoProfiler können die mit Hilfe der Attribute preImportScript und postImportScript deklarierten Skripte sowohl in IronPython als auch in C# implementiert werden. Alternativ ist auch die Bereitstellung einer .NET DLL möglich, in der eine Klasse das von docuvita.Contract bereitgestellte Interface IBatchImportScript implementiert ist.

Dateipfade und Aufrufkontext

Stellen Sie sicher, dass die Dateipfade für preImportScript und postImportScript im Kontext des docuvita Servers gültig sind. Auch die Dateizugriffsrechte auf die Skriptdatei und evtl. notwendige Zusatzmodule müssen für das Dienstkonto des docuvita.Servers ausreichend sein.

Skripte können durch die Deklaration im <object>-Element des zugehörigen Dokument- oder Strukturelement-Imports direkt vor dem Anlegen des Elements auf dem Server ausgeführt werden. Dadurch ist der direkte Eingriff auf die Verschlagwortungsdaten und die Ziel-Dokumentdatei möglich. Bei der Ziel-Dokumentdatei handelt es sich um die Dokumentdatei, die z.B. aus dem AutoProfiler nach einer barcode- oder feldgesteuerten Belegtrennung erzeugt

worden ist.

IronPython-Skript:
```
<object type="Dokumenttyp"
preImportScript="PathToScriptfile.py"… />
```

C#-Skript:
```
<object type="Dokumenttyp"
preImportScript="PathToScriptfile.cs"… />
```

.NET DLL:
```
<object type="Dokumenttyp"
preImportScript="KlassennameAusPlugin"… />
```

Bei dieser Art der Skriptausführung kann nur eine Skriptdatei angesteuert werden.

Die im Skript zur Verfügung stehenden Daten entsprechen denen der der Skriptausführung im AutoProfiler.

6.4.3 Ausführung direkt nach dem Dokument-/Objektimport

Im Unterschied zur Ausführung vor einem Dokumentimport stehen in einem Skript nach der Durchführung des eigentlichen Imports die docuvita-internen Zuordnungsinformationen zur Verfügung. Hier handelt es sich um die Objekt-ID (eindeutige Referenz auf das docuvita-Objekt) sowie um die Versions-ID (eindeutige Referenz auf die angelegte Dokumentversion; steht zur Verfügung, sofern es sich beim Import um einen Dokumentimport handelt).

IronPython-Skript:
```
<object type="Dokumenttyp"
postImportScript="PathToScriptfile.py"… />
```

C#-Skript:
```
<object type="Dokumenttyp"
postImportScript="PathToScriptfile.cs"… />
```

.NET DLL:
```
<object type="Dokumenttyp"
postImportScript="KlassennameAusPlugin"… />
```

Bei dieser Art der Skriptausführung kann wie beim preImportScript nur eine Skriptdatei angesteuert werden.

6.4.4 Besonderheiten für preImportScript mit IronPython

Der Skript-Rumpf für IronPython-Skripte als preImportScript entspricht dem des AutoProfilers. Im Unterschied zur Ausführung im AutoProfiler wird in der Variable fieldValues die Verschlagwortungsdeklaration für das anzulegende Objekt übergeben.

In den fieldValues werden zwei mögliche Feldarten übergeben:

- docuvita Standardfelder im Format OBJ.<Feldname>. Die Liste der verfügbaren Felder entspricht der des normalen Imports, vgl. Abschnitt 5.4 Mögliche Felder im XML-Tag „object“ dieses Handbuchs.
- Objekttyp-spezifische Felder. Hier wird einfach der Feldname angegeben.

Bitte beachten Sie, dass Zugriff und Rückschreibung von Feldern aus den fieldValues strikt in Großschreibung erfolgt.

Variablen im preImportScript

	Variablenname	Typ	Beschreibung
	log	ILog	log4net-Objekt für das Schreiben in das Importprotokoll
	fieldValues	Dictionary <**string**,**string**>	Feldwerte der Objekt-/ Dokumentenverschlagwortung
	docFilename	**string**	Dateiname der zu importierenden Dokumentdatei. Null, falls kein Dokumentenimport vorliegt.
	pageCount	**int**	Seitenzahl des zu importierenden Dokuments. Nur bei PDF- und TIF-Dokumenten.
	log	ILog	log4net-Objekt für das Schreiben in das Importprotokoll
	fieldValues	Dictionary <**string**,**string**>	Feldwerte der Objekt-/ Dokumentenverschlagwortung
	docFilename	**string**	Dateiname der zu importierenden Dokumentdatei. Null, falls kein Dokumentenimport vorliegt.

Bitte beachten Sie, dass die Ausführung der IronPython-Skripte im Kontext des docuvita Servers erfolgt. Es wird deshalb empfohlen, Erwägungen zum Ressourcenverbrauch und zur Stabilität zu beachten.

6.4.5 Besonderheiten für postImportScript mit IronPython

Bei postImportScript stehen einige weitere Variablen zur Verfügung. Da der docuvita Server zum Zeitpunkt der Ausführung dieses Skripts bereits die Anlage des entsprechend zugrundeliegenden Objekts durchgeführt hat, stehen die vergebenen Objekt- und Versions-IDs hier zusätzlich bereit.

Zusätzliche Variablen im postImportScript

	Variablenname	Typ	Beschreibung
	`obj_objectid`	**int**	Objekt-ID des aktuell verarbeiteten Dokuments bzw. Strukturelements
	alternativ: `obj_id`	**int**	analog obj_objectid
	`ver_versionid`	**int**	Versions-ID der aktuell importierten Dokumentversion (nur bei Dokument-Typen)
	alternativ: `ver_id`	**int**	analog ver_versionid

6.4.6 Besonderheiten für C#-Skripte

C#-Skripte müssen das IBatchImportScript-Interface aus der docuvita.Contract.dll implementieren. Legen Sie dazu ein neues Visual C# Projekt als „Klassenbibliothek“ an und referenzieren Sie in diesem Projekt die docuvita.Contract.dll aus dem Programmverzeichnis des docuvita Servers.

Beispielskript C#

```
using System;
using System.Collections.Generic;
using docuvita.Contract.Interfaces;

namespace docuvita.Sample {
    class SamplePreOrPostImportScript :
IBatchImportScript {

        public IBatchImportScriptHost
BatchImportScriptHost { get; set; }

        public void RunScript( int objectId, int
versionId, ref string docFilename, int pageCount,
Dictionary<string, string> fieldValues ) {

            //Skript-Logik
        }
    }
```

```
}
```

Die Methode RunScript erhält direkten Zugriff auf die bekannten Objektdaten. Hierbei gilt der gleiche Unterschied wie bei IronPython-Skripten:

- fieldValues ist sowohl bei preImportScript als auch bei postImportScript verfügbar.
- Die Werte für objectId, versionId stehen bei postImportScript zur Verfügung.
- Ein Sonderfall ist die objectId. Sofern der anstehende Import eine Aktualisierung eines bereits bestehenden Objekts (identifiziert anhand von Schlüsselfeldern) durchführt, so steht die Variable objectId auch in der Skriptlogik zur Verfügung.

Bei der C#-Skriptausführung kann auf Elemente des .NET Framework zugegriffen werden. Weitere Skriptdateien können zur Auslagerung von Logik mit Hilfe der folgenden Compilerdirektive angegeben werden:
//css_import Pfad-Zur-Ausgelagerten-Skriptdatei.cs

Zusätzlich können auch externe .NET Assemblies (DLLs) referenziert werden. Für Assemblies im Global Assembly Cache (GAC) ist nur die Angabe des eindeutigen Assemblynamens notwendig. Andere DLLs können über den Dateinamen angegeben werden.
//css_reference Pfad-Zur-Assembly.dll

Sämtliche Compilerdirektiven müssen am Anfang der Skriptdatei stehen.

Bitte beachten Sie, dass die Ausführung der C#-Skripte im Kontext des docuvita Servers erfolgt. Es wird deshalb empfohlen, Erwägungen zum Ressourcenverbrauch und zur Stabilität zu beachten.

6.4.7 Besonderheiten für .NET DLLs

Die als BatchImportScript zu verwendende Logik kann neben der Bereitstellung in einer Skriptdatei auch in Form einer DLL implementiert werden. Während prinzipiell jede .NET-Programmiersprache verwendet werden kann, empfiehlt sich ein Vorgehen ausgehend von C# Skripten. Wird ein C#-Skript wie im o.g. Beispiel erstellt, so kann es in Visual Studio auch zu einer .NET DLL kompiliert werden. Der einzige Unterschied bei der Verwendung ist die Angabe des Klassennamens statt des Skript-Dateinamens.

Die .NET DLL muss einige Anforderungen erfüllen, damit sie erfolgreich als Plugin vom docuvita Server geladen werden kann:

- Der Assembly- und Dateiname muss dem Schema <Hersteller>.BatchImportScript.<Titel> entsprechen.
- Der Aufruf des Skripts erfolgt dann mit dem eindeutigen Klassennamen, also z.B. postImportScript="SamplePostImportScript".

Bitte beachten Sie, dass die Ausführung der .NET DLLs im Kontext des docuvita

Servers erfolgt. Es wird deshalb empfohlen, Erwägungen zum Ressourcenverbrauch und zur Stabilität zu beachten.

Variablen

7 Variablen

Das Importprogramm unterstützt die Verwendung bestimmter Variablen in den dvImport-Dateien und auch bei Bedingungsprüfungen sowie Daten-Lookups und in Skripten. Der Aufbau ist dabei für ausgelesene oder durch Daten-Lookups und Skripte gewonnene Variablen wie folgt: `@@Name der Variablen@@`. Diese Variablen werden bei der Umwandlung des Import-Templates in dvImport-Dateien durch die entsprechenden Werte gefüllt.

Es gibt außerdem noch Variablen, die nicht vom AutoProfiler verarbeitet werden sondern erst bei der Verarbeitung der dvImport-Datei am docuvita.Server. Diese Variablen können auch in selbst erstellten dvImport-Dateien verwendet werden. Der Aufbau dieser Variablen ist wie folgt: `%Name der Variablen%`.

Um diese Variablen zu verwenden, tragen Sie sie einfach in die Wert-Felder der Importdatei ein. Während des Imports werden diese dann in entsprechende Werte umgesetzt.

Beispiel zur Verwendung von Variablen

```
obj.description="Objekt vom %ds%, %ts%"
```

wird während des Importvorgangs in

```
obj.description="Objekt vom 22.04.2016, 12:12"
```

umgewandelt

7.1 Systemvariablen

Diese Variablen werden vom System verwendet und haben eine feste Funktion. Sie können daher nicht vom Benutzer anderweitig verwendet werden bzw. neue Werte zugewiesen bekommen.

	Variable	Funktion
	@@Filename@@	Liefert den Pfad zur Quelldatei und wird automatisch zur Laufzeit des AutoProfilers durch den vollqualifizierten Pfad & Dateinamen ersetzt. `@@Filename@@` wird für gewöhnlich mit dem Standard-Feld zur Angabe des zu importierenden Dokumentes `doc.Filename` verwendet: `<object … doc.FileName="@@Filename@@" … />`
	@@FileDelete@@	Löscht die Quelldatei.

		`<object … doc.FileDelete="@@FileDelete@@" … />` Diese Funktion ist sinnvoll wenn keine Datei importiert (z.B. wenn eine leere Datei zum Starten eines AutoProfilers verwendet wird - siehe Quellordner) werden soll, oder bei der Verwendung einer Steuerdatei (siehe XmlContentParser) wenn gleichzeitig der Wert für `doc.Filename` von einer anderen, z.B. aus der Steuerdatei ausgelesenen, Variablen gefüllt wird. Eine gleichzeitige Verwendeung von `doc.FileName="@@Filename@@"` und `doc.FileDelete="@@FileDelete@@"` unter Verwendung von `@@Filename@@` ist nicht möglich.

7.2 Immer verfügbare Variablen

Die immer verfügbaren Variablen werden in der Notation `%Name der Variablen%` angegeben.

	Variable	Funktion

	%fp%
	%fn%
	%fx%
		Liefert den Pfad (%fp%), den Namen (%fn%) und die Dateinamenserweiterung (%fx%) der zu importierenden Dokumentdatei, sofern vorhanden. Falls ein Objekt (z.B. eine Kundenakte) importiert wird, liefern diese Variablen keinen Wert zurück.
		
		Beispiel
		
		`%fp%%fn%%fx%`
		
		liefert beispielsweise
		
		`C:\import\test\test0001.doc`

	%ip% **%in%** **%ix%**	Liefert den Pfad (%ip%), den Namen (%in%) und die Dateinamenserweiterung (%ix%) der verwendeten Importdatei. **Beispiel** `%ip%%in%%ix%` liefert beispielsweise

		`C:\import\beispiel.dvImport`

	%ds%
	%dl%
		Liefert das Datum zum Zeitpunkt des Imports in kurzer (%ds%) oder in langer (%dl%) Form.
		
		Beispiele
		
		`%ds%` liefert z.B. `22.04.2016`
		`%dl%` liefert z.B. `Freitag, 22. April 2016`
		
		Die Formatierung der Ausgabe ist von den Einstellungen des lokalen Systems abhängig, auf dem das Importprogramm ausgeführt wird.

	%ts%
	%tl%
		Liefert die Uhrzeit zum Zeitpunkt des Imports in kurzer (%ts%) oder in langer (%tl%) Form.
		
		Beispiele
		
		`%ts%` liefert z.B. `12:15`
		`%tl%` liefert z.B. `12:15:29`
		
		Die Formatierung der Ausgabe ist von den Einstellungen des lokalen Systems abhängig, auf dem das Importprogramm ausgeführt wird.

	%un%
	%ui%
		Liefert den Benutzernamen (%un%) und die eindeutige ID (%ui%) des Benutzers, mit dessen Anmeldeinformationen der Import durchgeführt wird.
		
		Beispiele
		
		`%un%` liefert z.B. `Administrator`
		`%ui%` liefert z.B. `2`

	%gn%
	%gi%
		Liefert den Namen (%gn%) und die eindeutige ID (%gi %) der Benutzergruppe des Benutzers, mit dessen Anmeldeinformationen der Import durchgeführt wird.
		
		Beispiele
		
		`%gn%` liefert z.B. `Systemadministratoren`
		`%gi%` liefert z.B. `1`

	%ot%	Liefert den Objekttyp des aktuell zu importierenden Objekts **Beispiel** `%ot%` liefert z.B. `KUNDENAKTE`

		Die Ausgabe des Objekttyps erfolgt stets in Großschreibung.

7.3 E-Mail Variablen

Die E-Mail-Variablen sind lediglich bei der korrekten Verwendung des *MailContentParser* (siehe Kapitel MailContentParser) verfügbar. Diese Variablen werden vom AutoProfiler automatisch bei der Erzeugung der dvImport-Datei umgewandelt. Es handelt sich bei den verfügbaren Platzhaltern um folgende mit festen Bezeichnern versehenen Werte:

	Variable	Funktion
	@@MAILFULLADDR@@	Volle Mailadresse des externen Empfängers/ Absenders (vgl. Text)
	@@MAILFULLDOMAIN@@	Volle Domainangabe der o.g. Adresse (zum Beispiel docuvita.de für info@docuvita.de)
	@@MAILPRETLD@@	Domainangabe ohne Toplevel-Domain (zum Beispiel docuvita für docuvita.de)
	@@MAILTLD@@	Toplevel-Domain der o.g. Adresse (zum Beispiel de für docuvita.de)
	@@MAILTYPE@@	INTERNAL für interne, INCOMING für eingehende, OUTGOING für ausgehende Mails
	@@MAILUSERNAME@@	Benutzername der o.g. Adresse (vor @-Zeichen)
	@@MAILSUBJECT@@	Betreffzeile der E-Mail
	@@MAILBODY@@	Text der E-Mail (erste 800 Zeichen)
	@@MAILSENDDATE@@	Sendedatum
	@@MAILFROM@@	Absender
	@@MAILTO@@	Empfänger
	@@MAILLOCALBOX@@	lokale Mailbox (wird anhand der Voreinstellung für die lokalen Maildomains ermittelt)
	@@MAILATTCOUNT@@	Anzahl Attachments
	@@MAILMESSAGEID@@	Eindeutige MessageID der E-Mail.
	@@MAILSENSITIVITY@@	Vertraulichkeitsstatus der E-Mail, z.B. *company confidential*. Falls kein Vertraulichkeitsstatus beim Versand der E-Mail eingestellt wurde, ist dieser Wert leer.

7.4 Office Variablen

Auch aus Word Dokumenten können Variablen ausgelesen werden. Die Word-Variablen sind lediglich bei der korrekten Verwendung des *WordContentParsers* verfügbar. Diese Variablen werden vom AutoProfiler automatisch bei der Erzeugung der dvImport-Datei umgewandelt. Es handelt sich bei den verfügbaren Platzhaltern um folgende mit festen Bezeichnern versehenen Werte:

Feste Eigenschaftsfelder von Word 97 (.doc und .docx) und neuer:

	Variable	**Funktion**
	`@@CATEGORY@@`	Kategorien
	`@@KEYWORDS@@`	Markierungen
	`@@TITLE@@`	Titel
	`@@SUBJECT@@`	Thema

Zusätzlich Eigenschaftsfelder von Word 97 bis 2003 (.doc):

	Variable	**Funktion**
	`@@AUTHOR@@`	Autor
	`@@COMMENTS@@`	Kommentare
	`@@COMPANY@@`	Firma
	`@@KEYWORDS@@`	Stichwörter
	`@@DATECREATED@@`	Inhalt erstellt
	`@@DATELASTSAVED@@`	Letzte Speicherung

Diese Werte können über Word im Reiter *Zusammenfassung* eingesehen und bearbeitet werden. Öffnen Sie dazu ein Word-Dokument und wechseln auf *Datei → Eigenschaften → Erweiterte Eigenschaften*.

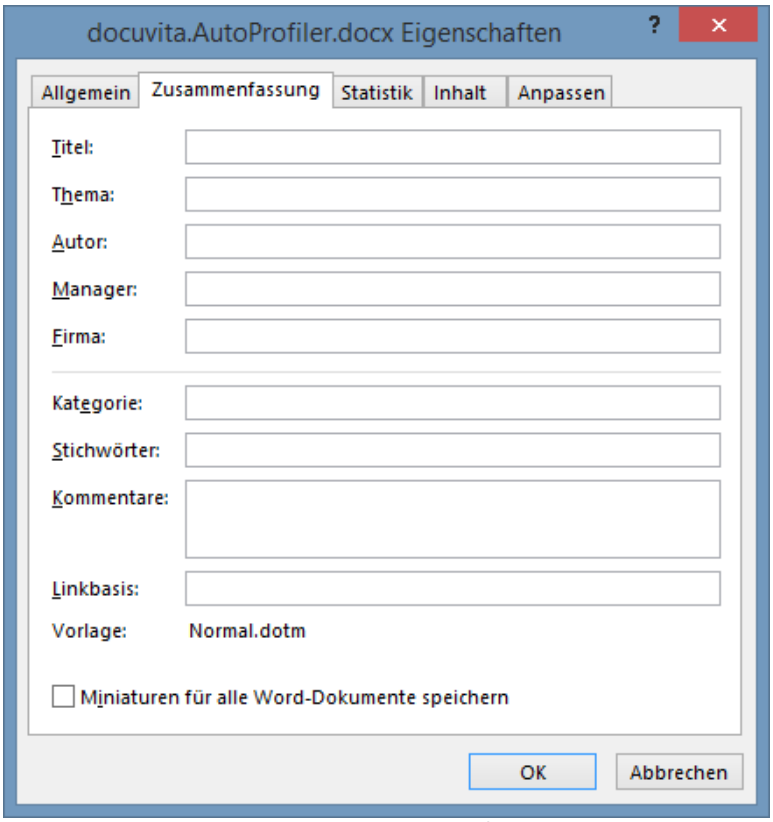


Abb. 7-1: Word Eigenschaften

Außerdem kann bei beiden Versionen auf sämtliche selbst hinzugefügte Eigenschaften zugegriffen werden. Diese Eigenschaften können über den Reiter *Anpassen* erstellt und angepasst werden. Der Variablenname entspricht der Spalte *Name* und der Inhalt der Spalte *Wert*. Auch diese Variablen können in der @@-Notation genutzt werden.

Zusätzlich wird der komplette Pagetext geliefert (analog PdfContentParser). Siehe dazu auch Kapitel Pagetext.

Für Excel und PowerPoint Dokumente stehen keine speziellen Variablen zur Verfügung.

7.5 Ausgelesene Variablen

Die in einem AutoProfiler über Parser, Daten-Lookups und Skripte gesammelten Variablen werden pro verarbeitetem Dokument in Feld-Wert-Listen gespeichert. Wird eine Variable im Verlauf der Verarbeitung eines Dokumentes mehrfach mit Daten gefüllt bleibt immer der letzte zugewiesene Wert erhalten. Bei einem Dokument handelt es sich in diesem Zusammenhang um ein Zieldokument (PDF-Dokumente können beim Einlesen in mehrere Zieldokumente umgewandelt werden).

7.6 Spezial Variablen

7.6.1 Teile von bestehenden Variablen

Es können reguläre Ausdrücke auf ausgelesen Variablen angewendet werden, um nur einen Teil der Variablen weiterzuverwenden. Der reguläre Ausdruck wird mit einem doppelten Doppelpunkt von der Variablen abgetrennt:
`@@Variablenname::RegulärerAusdruck@@`

Der reguläre Ausdruck soll wird auf eine Variable angewendet; das Ergebnis wird in die Feld-Wert-Listen eingetragen.

Beispiel:

- Variable *Belegnummer* ist vorhanden mit dem Wert RE12345
- Syntax: `@@Belegnummer::RE(\d{5})@@`
- Als Wert wird zurückgegeben: 12345

Der verwendete reguläre Ausdruck muss exakt einen Ergebniswert zurückgeben, im angegebenen Beispiel durch den geklammerten Wert kenntlich.

7.6.2 Pagetext

Diese Möglichkeit ist beim PDF- und WordContentParser verfügbar.

Der kompletter Text steht als Variable zur Verfügung und kann über Regular Expressions nach beliebigen Kriterien ausgewertet werden um einzelne Bestandteile in Import-Templates (siehe Aufbau der dvImport-Datei) zu verwenden. Für die Auswertung wird keine Variable angegeben, sondern lediglich eine RegEx als Variable in @@ verwendet.

Der reguläre Ausdruck soll auf den ausgelesenen Seitentext angewendet werden, das Ergebnis wird in die Feld-Wert-Listen eingetragen.

Beispiel:

- Im Text kommt vor: Mandant: 833
- Syntax: `@@Mandant:\s(\d{3})\s+@@`
- Als Wert wird zurückgegeben: 833

Der Pagetext steht nur in Import-Templates zur Verfügung

Die Variable Pagetext steht nur in den Import-Templates zur Verfügung und nicht bei den Bedingungen zur Ausführung, Daten-Lookup und Skripten zur Verfügung.

Auswertung von Prüfbedingungen

8 Auswertung von Prüfbedingungen

Jeder AutoProfiler kann bei Zutreffen bestimmter Bedingungen eine ODBC-Datenquelle nach Zusatzinformationen befragen oder mit einer Fallunterscheidung zwischen mehreren Import-Templates auswählen. Auch die Ausführung von Skripten kann an eine Bedingung geknüpft sein. Grundlage der Bedingungsprüfung ist die Auswertung logischer Ausdrücke im Feld „Bedingung zur Ausführung“ entsprechenden Konfigurationsabschnitts.

8.1 Grundlagen

Jede Prüfbedingung, die im Feld *Bedingung zur Ausführung* eingetragen wird, muss sich beim Betrieb des AutoProfilers zu einem gültigen logischen Wert (Wahr/Falsch) evaluieren lassen. Eine Prüfbedingung besteht dabei aus mehreren Komponenten:

- **Variablen**. Die Verwendung von Variablen erfolgt wie in den Import-Templates des AutoProfilers (siehe auch Variablen).
  - **Standardvariablen** werden in der Form @@Variablenname@@ verwendet (siehe auch Kapitel E-Mail Variablen bis Ausgelesene Variablen).
  - **Spezialvariablen** sind Variablen, denen ein regulärer Ausdruck nachgeschaltet ist. Der Zweck ist in der Regel, nur einen Teilbereich des eigentlichen Variablenwertes zurückzugeben. **Spezialvariablen** werden wie in den Import-Templates in der Form @@Variablenname::RegulärerAusdruck@@ verwendet (siehe auch Spezial Variablen).
  - **Literale.** Als Literale werden feste Werte bezeichnet. In der Regel kommen Zeichenketten (in einfachen Hochkomma), Datumswerte sowie Zahlen (ohne Kennzeichnung) zum Einsatz. Ein besonderes Literal ist „null“.
- **Operatoren**. Mit Hilfe von Operatoren werden Literale und Variablen bzw. Variablen untereinander verglichen.
- **Methoden**. Methoden werden auf Literale, Operatoren oder logische (Teil-) Ausdrücke angewendet.

Beispiel: Prüfbedingung „Belegart=Rechnung“

```
@@Belegart@@=='Rechnung'
```

Das aus dem Dokument zurückgegebene Feld Belegart wird in Standardschreibweise für Variablen verwendet (Umschlossen mit @@).

Der einfachste Operator ist der Vergleichsoperator ==.

Die Prüfung erfolgt hier gegen einen festen Wert, einer Zeichenkette in einfachen Hochkommata.

Das Literal „null“

Wird in einer Prüfbedingung der Wert einer Variablen abgefragt, die aus einem bestimmten Dokument nicht geliefert wird, so würde ein ungültiger Verweis entstehen und ein Fehler protokolliert werden.

Um dies zu verhindern, müssen entsprechende Variablen zuvor auf Ungleichheit zu null geprüft werden:

```
@@Feld@@!=null and ...
```

Erfolgt lediglich eine Prüfung auf einen bestimmten Wert (Operator ==), so kann die null-Prüfung entfallen:

`@@Feld@@=='A'` (ist „falsch“, auch wenn @@Feld@@ nicht vorhanden ist)

8.2 Operatoren

8.2.1 Standardoperatoren

Der AutoProfiler unterstützt die logischen Standardoperatoren bei der Auswertung von Ausdrücken in den Bedingungsprüfungen.

Operator	Beispiel	Bedeutung
==	`@@Belegart@@=='Rechnung'`	Gleichheit (auch mit null!)
!=	`@@Barcode@@!=null`	Ungleichheit (auch mit null!)
>, >=	`@@Kdnr@@>10000`	Größer, Größer oder gleich
<, <=	`@@Kdnr@@<=10000`	Kleiner, Kleiner oder gleich

8.2.2 Spezialoperatoren

Neben den Standardoperatoren wird eine Reihe von Spezialoperatoren unterstützt, die die Prüfung anhand regulärer Ausdrücke oder SQL-konformer Suchmuster erlauben.

Operator	Beispiel	Bedeutung
like	`'Abc' like '[A-Z]b*'` (ergibt wahr) `'Abc' like`	Musterprüfung mit Wildcard

		`'?'` (ergibt falsch)	
	matches	`@@Barcode@@ matches '\d{7}'`	Musterprüfung mit regulären Ausdrücken
	in	`@@Documenttype@@ in ('BELEG', 'SONSTIGE')`	Prüft, ob Element in einer Auflistung enthalten ist
	between	`'efg' between {'abc', 'xyz'}` (ergibt wahr)	Prüft, ob ein Element zwischen zwei Elementen einer Liste enthalten ist
	is	`'xyz' is int`	Typprüfung Gültige Typen sind die .NET Basis-Werttypen (int, string, float, double, …)

8.2.3 Mathematische Operatoren

Der AutoProfiler unterstützt die mathematische Standardoperatoren bei der Auswertung von Ausdrücken in den Bedingungsprüfungen.

	Operator	Beispiel	Bedeutung
	+	`1 + 1` `'a' + 'b'` (ergibt 'ab')	Addition (bei Zeichenketten: Verkettung)
	-	`5 - 1` (ergibt 4)	Subtraktion (nur bei Zahlen zulässig)
	*****	`-2 * -3` (ergibt 6)	Multiplikation
	/	`6 / -3` (ergibt -2)	Division
	% (Modulo-Divsion)	`7 % 4` (ergibt 3)	Modulo-Division. Gibt den ganzzahligen Rest einer Division zurück

8.3 Methoden und Eigenschaften

Zusätzlich zu den verfügbaren Operatoren sind Vorgänge möglich, die mit Hilfe der Ausführung von Objektmethoden oder dem Abfragen von Eigenschaften (Properties) durchgeführt werden.

Es sind alle Methoden und Eigenschaften zulässig, die von den jeweiligen Standardtypen (string, int, …) des .NET Framework unterstützt werden. Die folgende Tabelle stellt einige Beispiele dar:

	Operator	Beispiel	Bedeutung
	StartsWith(string)	`@@Belegnummer@@.StartsWith('RE')`	Prüft, ob eine Zeichenkette mit einem bestimmten Wert

			beginnt
	EndsWith(string)	`@@Belegnummer@@.EndsWith('_P')`	Prüft, ob eine Zeichenkette mit einem bestimmten Wert endet
	IndexOf(string)	`@@Belegnummer@@.IndexOf('RE')==1`	Prüft, an welcher Stelle die angegebene Zeichenkette vorkommt: -1: kommt nicht vor 0 oder größer: kommt vor
	ToUpper(), ToLower()	`'_P'.ToLower()` (ergibt '_p')	Wandelt eine Zeichenkette in Groß-/Kleinschreibung um
	Trim()	`'  a  ' (ergibt 'a')`	Entfernt vorangestellte und nachgestellte Leerzeichen aus einer Zeichenkette

Die Methoden können zur Auswertung auch verkettet werden z.B.:
`@@Belegnummer@@.ToUpper().Trim() == 'RECHNUNG'`

Spring Framework

Es sind weitere Methoden aus dem Spring Framework nutzbar:
http://www.springframework.net/doc-latest/reference/html/expressions.html

Der auszuwertende Ausdruck wird im Spring Framework mit Hilfe der Methode ExpressionEvaluator.GetValue geprüft. Dabei muss zwingend ein Boolean-Wert zurückgegeben werden.

8.4 Verkettung von Bedingungen

Verschiedene Teile von Ausdrücken können mit den Schlüsselwörtern *and* sowie *or* verknüpft werden (Kleinschreibung zwingend erforderlich). Es gelten die logischen Verkettungs- und Klammerregeln.

	Beispiel für die Verkettung von Bedingungen

Hier ein Beispiel für eine Verkettung von Bedingungen:

```
@@vorgangsnr@@!=null and (@@belegart@@=='Angebot' or
@@belegart@@=='Rechnung'
```

Damit diese Bedingung erfüllt ist darf die Variable @@vorgangsnr@@ nicht leer sein und die Variable @@belegart@@ muss entweder den Wert Angebot oder Rechnung enthalten.

```
@@vorgangsnr@@!=null and @@belegart@@=='Angebot' or
@@belegart@@=='Rechnung'
```

Ohne die Klammer darf die Variable @@vorgangsnr@@ nicht leer sein und die Variable @@belegart@@ muss den Wert Angebot enthalten. Alternativ reicht es aus, dass die Variable @@belegart@@ den Wert Rechnung enthält. Die erste Bedingung hat in diesem Fall dann keinen Einfluss mehr.

Überwachung von Imports

9 Überwachung von Imports

Die einzelnen Schritte der Verarbeitung können über den docuvita.Client detailliert nachverfolgt werden. Melden Sie sich hierzu mit einem berechtigten Benutzer (Mitglied der Importadmingroup). Wechseln Sie nun in den *Import*-Ribbon. Weitere Informationen, dazu wie Sie den *Import*-Ribbon angezeigt bekommen finden Sie im Administrationshandbuch - docuvita.Server - Customizing.

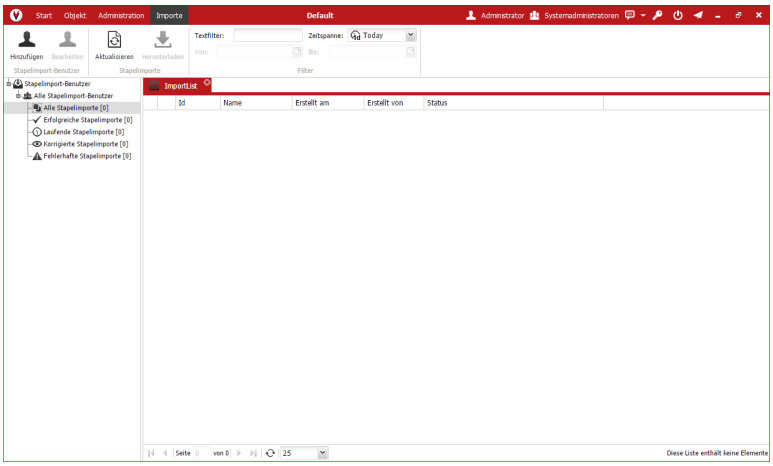


Abb. 9-1: Import-Ribbon

Weiter Informationen zur Überprüfung der Imports und zum Vorgehen im Fehlerfall erhalten Sie im Benutzerhandbuch - docuvita.Client.

AutoProfiler Testprogramme

10 AutoProfiler Testprogramme

Das docuvita.AutoProfiler Konfigurationsprogramm enthält einige Testprogramme, mit denen Sie einen Großteil der Funktionen überprüfen können, ohne einen tatsächlichen Import nach docuvita durchführen zu müssen.

Die einzelnen Testprogramme erreichen Sie direkt über das Konfigurationsprogramm.

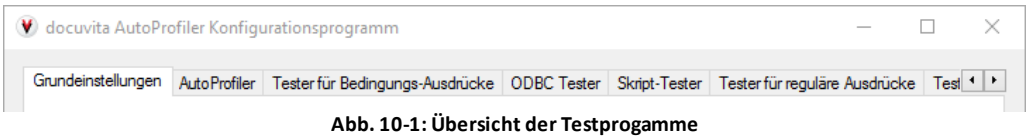


Abb. 10-1: Übersicht der Testprogamme

Mit dem Button können Sie weiter durch die einzelnen Testprogramme durchschalten.

Es gibt folgende Testprogramme:

- **Tester für Bedingungs-Ausdrücke**
  Testen Sie Ihre Bedingungen zur Ausführung z.B. zur Ausführung von Daten-Lookups,Skripten und Import-Templates.

- **ODBC Tester**
  Testen und debuggen Sie Ihre Daten-Lookups bevor Sie diese im laufenden Betrieb automatisiert einsetzen. Außerdem finden Sie hier Beispiele für Connection-Strings zu gängigen Datenbanksystemen.

- **Skript-Tester**
  Testen und debuggen Sie Ihre Skripten bevor Sie diese im laufenden Betrieb automatisiert einsetzen.

- **Tester für Reguläre Ausdrücke**
  Testen Sie Reguläre Ausdrücke, die Sie an verschiedenen Stellen auf Variablen im AutoProfiler anwenden können.

- **Tester für XSLTDispatche**
  Testen und debuggen Sie die XSLT-Transformation von XML-Dateien.

- **PDF-Tester**
  Testen Sie PDF-Dateien, bevor Sie mit dem PdfContentParser elektronische PDF-Dokumente verarbeiten. Es wird Ihnen der gesamte zur Verwendung stehende Textinhalt angezeigt. So können Sie überprüfen, ob Ihre gesuchten Werte auch tatsächlich enthalten und zugreifbar sind. So können Sie Fehler, die ggf. durch einen nicht optimalen Einsatz eines externen Belegdesigner entstehen können, leicht entdecken.

- **Tester für E-Mailabholung**

Testen Sie den Zugriff auf ein Postfach zur E-Mailabholung bevor Sie es im AutoProfiler konfigurieren.

- **Barcode-Tester**
  Testen Sie gescannte TIF- oder PDF-Dokumente auf die enthaltenen Barcodes. So können Sie einfach feststellen, ob das Dokument in dem Format lesbar/auswertbar ist (z.B. zum Erkennen von Problemen, die durch eine Komprimierung entstehen). Zusätzlich können auf einfache Art verschiedene Einstellung zur Optimierung der Erkennungsqualität getestet werden, ohne in laufende Prozesse eingreifen zu müssen. Der Barcodetester eignet sich hervorragend um fehlerhafte Scan-Imports (Überwachung von Imports) zu überprüfen.

Weiterverwenden der Tests

An machen Stellen im AutoProfiler Konfigurationsprogramm haben Sie die Möglichkeit Ihre getesteten Funktionen und Werte in eine Konfiguration zu übernehmen. Achten Sie hierfür auf den Button 

und *Aktuellen Wert aus ...-Tester übernehmen*.