Installationsvoraussetzungen AutoOCR Server:

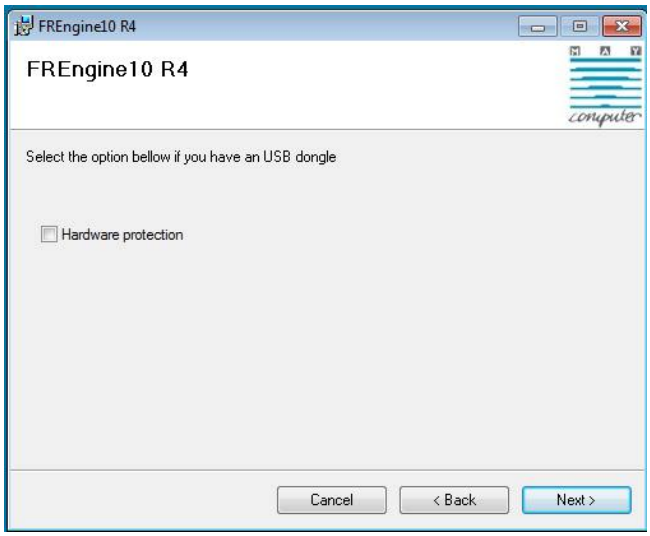
Empfohlen:

- Windows 7 - 64bit
- mind. 4GB RAM
- MultiCore und/oder Multi Prozessor Architektur wird unterstützt und beschleunigt die OCR Verarbeitung.

Schritt 1: Installation Abbyy OCR Engine

Download Setup:
http://www.may.co.at/webrr/respfile.php?file=MAYComp/Current/RO/FREngine10_x86.zip

ZIP extrahieren und Setup.exe als Administrator ausführen.



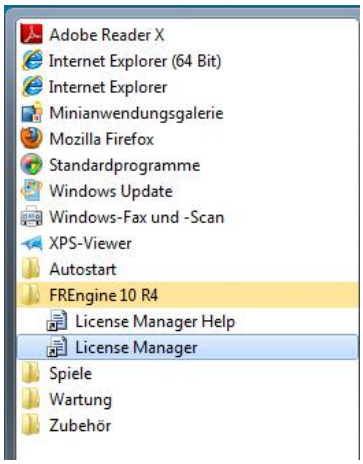
Je nachdem ob ein USB Dongle oder eine Softwarelizenz verwendet wird – ist diese Option bei der Installation anzuhaken oder nicht. Bei der Hardware Protection Installation den USB Stecker erst einstecken wenn die Installationsroutine dazu auffordert. Standardmäßig ist diese Option nicht ausgewählt da meist eine Softwarelizenz verwendet wird.

Achtung: Bitte beachten Sie dass es für physischen Maschinen und Installationen in virtuellen Umgebungen unterschiedliche Lizenz-Keys gibt – bitte dies bei der Bestellung angeben.

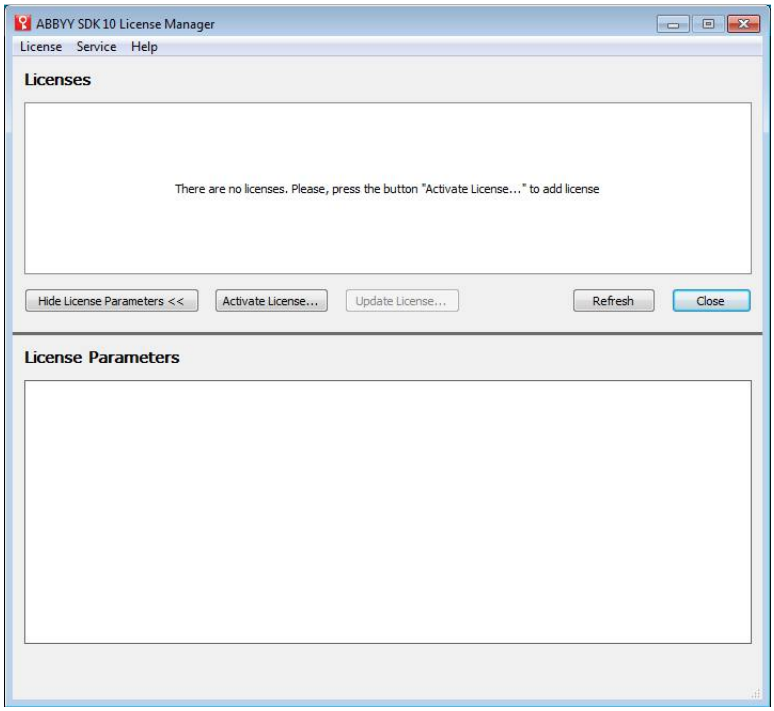
Schritt 2: Lizensierung Abbyy OCR Engine - Softwarelizenz

Nach der Installation muss die Abbyy OCR Engine Softwarelizenz aktiviert werden. Sie erhalten von uns dafür einen 24 stelligen Seriennummer (6 Gruppen zu je 4 Zeichen). Normalerweise erfolgt die Aktivierung per Internetverbindung über Port 80 oder 8080. Diese Ports müssen für die Onlineaktivierung freigeschalten sein. Alternativ gibt es auch andere Möglichkeiten der Aktivierung per E-Mail bzw. per Telefon.

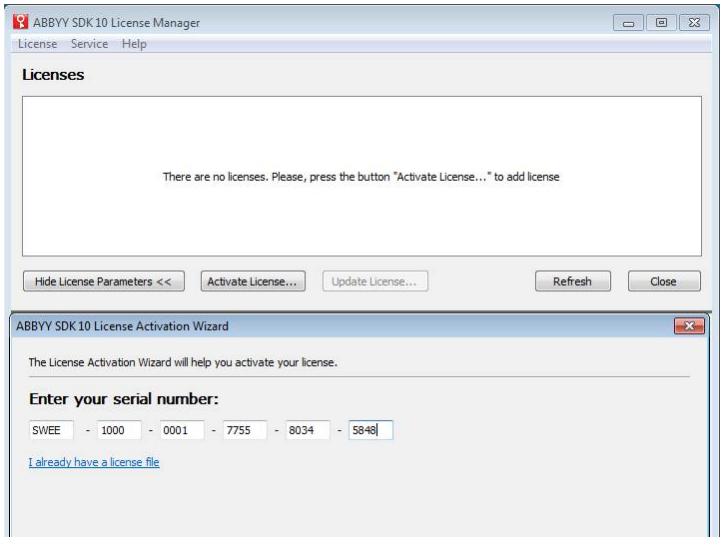
Aufruf – Abbyy License Manager:



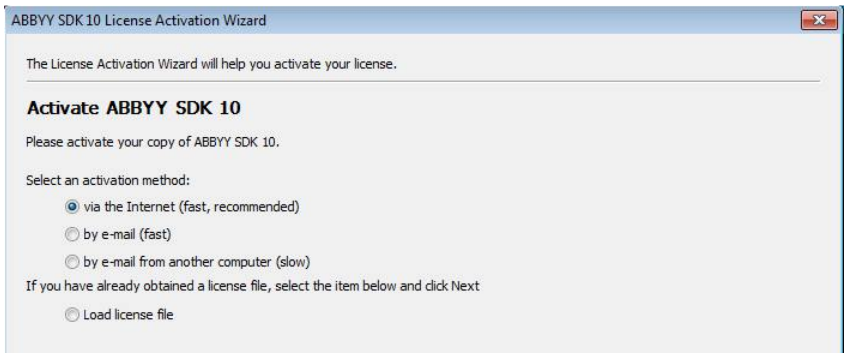
„Activate License“



Eingabe der Seriennummer:



Aktivierung der Lizenz:

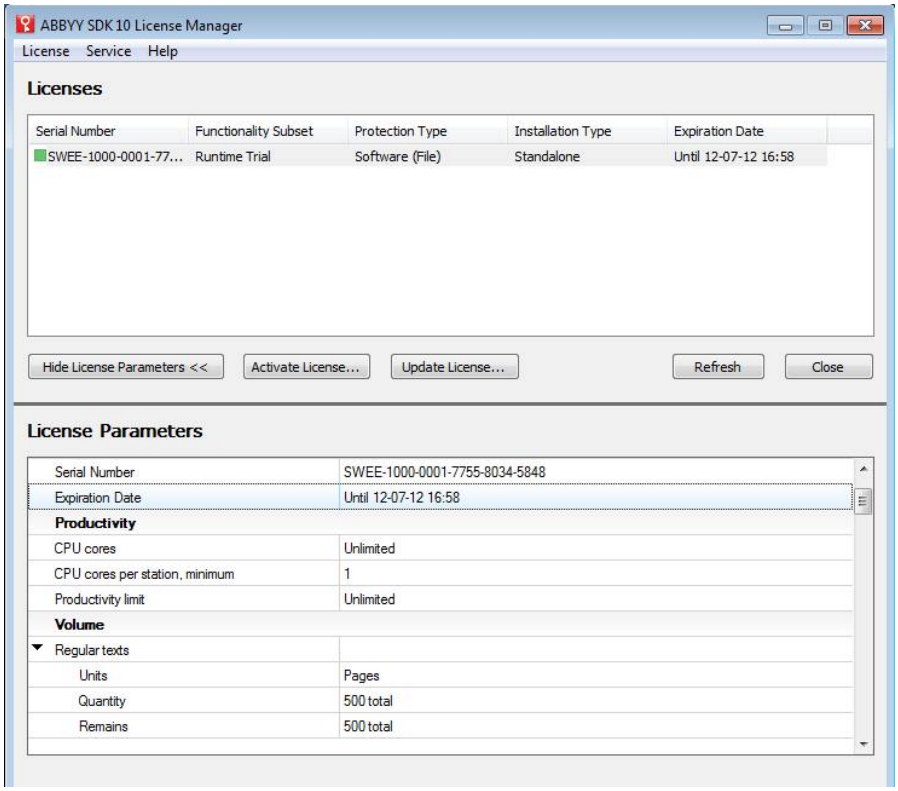


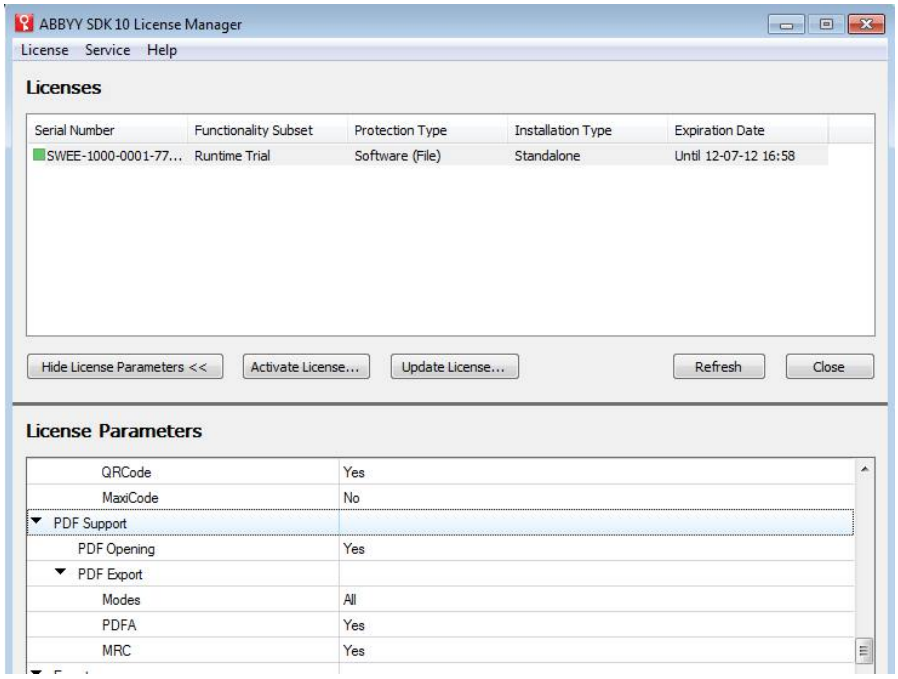
Erfolgreiche Aktivierung:

Überprüfen der Lizenz Parameter:

Wählen Sie die aktivierte Lizenz an und überprüfen Sie die Lizenzparameter ob diese mit der bestellten Lizenz übereinstimmen – Anzahl der CPU´s, Seitenzahl pro Monat, PDF Export Option.

Stimmen diese Informationen so ist die Installation der Abbyy OCR Engine und die Aktivierung der Lizenz erfolgreich verlaufen.



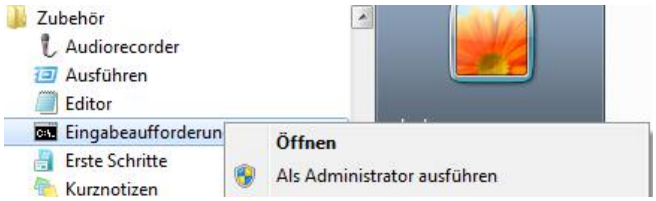


Installation USB Dongle Treiber – Hardware Protection (Optional)

Vor der Installation bitte unbedingt diese Information durchlesen: http://www.pdfblog.at/?p=1652

Download - **iKey1000 32bit** Treiber >>>
Download – **iKey1000 64bit** Treiber >>>

Unter Windows 7 – 64bit kopieren Sie die Datei „iKeyDrvr.msi“ z.b. in den Folder „C:\Program Files\Common Files\MAYComputer“ – öffnen Sie die Kommandozeile als Administrator



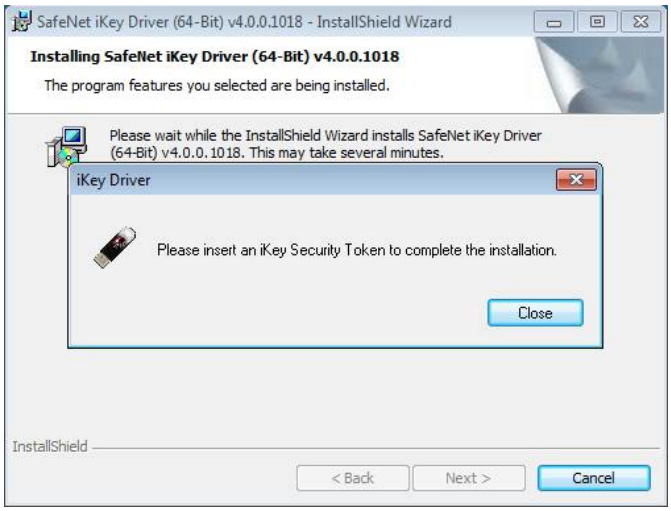
Wechseln sie in das Verzeichnis „C:\Program Files\Common Files\MAYComputer“

Führen Sie das Setup mit dem entsprechenden Commandline Paramter aus:

msiexec /i ikeydrvr.msi VR=OFF

```
C:\Windows>cd C:\Program Files\Common Files\MAYComputer
C:\Program Files\Common Files\MAYComputer>msiexec /i ikeydrvr.msi VR=OFF
```

Stecken Sie den den Dongle erst ein wenn sie dazu aufgefordert werden.



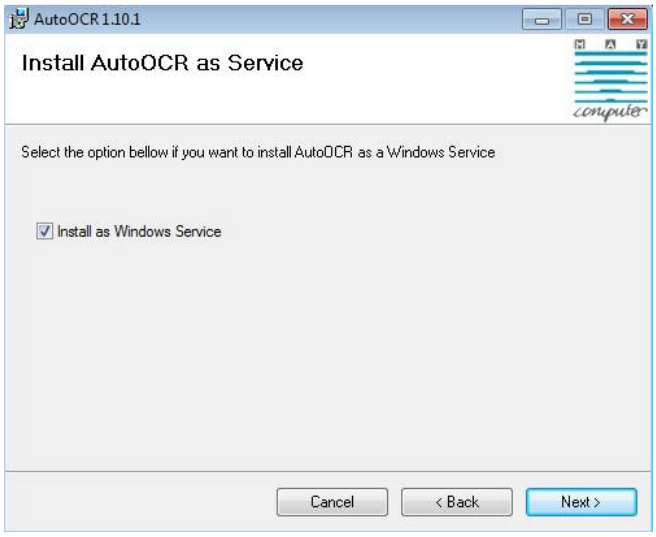
Schritt 2: Installation des AutoOCR Servers

Download Setup:
http://www.may.co.at/webrr/respfile.php?file=MAYComp/Current/RO/AutoOCR_Setup.zip

Achtung: Erfolgt die Installation von AutoOCR auf einem virtuellen Rechner so wird dafür eine spezielle Setup Version benötigt. Bitte kontaktieren Sie uns diesbezüglich unter support@xkey.at

ZIP extrahieren und Setup.exe als Administrator ausführen.

„Install as Service“



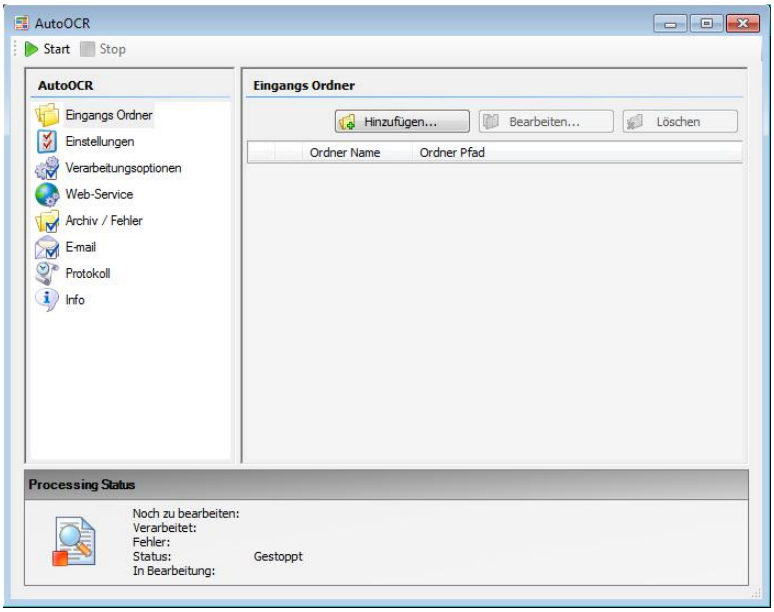
AutoOCR kann entweder als „normale Anwendung“ oder als „Service“ installiert werden. Üblicherweise wird AutoOCR als Service installiert – dafür ist bei der Installation die hier abgebildete Option anzuhaken.

Bei Installation als Service sind bestimmte Dinge zu beachten:

- Abbyy FineReader OCR Engine – Dongeltreiber – 32 / 64bit- http://www.pdfblog.at/?p=1652
- Windows Dienst – Zugriff auf Netzwerkressourcen – was ist zu beachten? http://www.pdfblog.at/?p=1688
- AutoOCR – als Service – was ist zu beachten ? http://www.pdfblog.at/?p=1641

Nach der Installation erscheint ein AutoOCR Icon am Desktop.

Rebooten Sie den Rechner und Starten danach „AutoOCR“ um das Userinterface für die Konfiguration anzuzeigen.

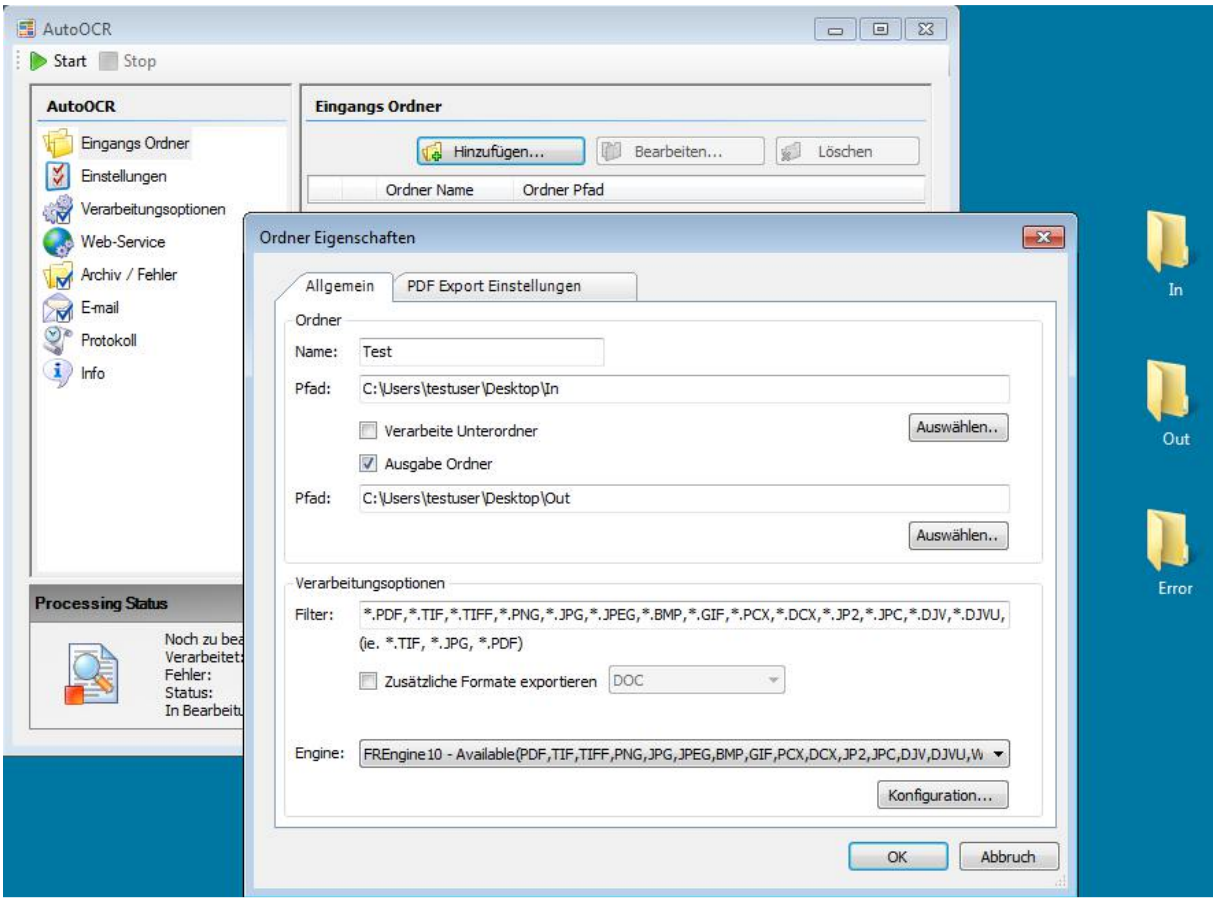


AutoOCR ist in der Lage Images oder PDF aus einem oder mehreren „In“ Foldern zu verarbeiten oder aber auch über eine Web-Service Schnittstelle angesprochen zu werden. Die Integration z.B. mit Alfresco erfolgt über die REST – Web-Service Schnittstelle.

Als erster Schritt um zu überprüfen ob die Installation der Abbyy Engine sowie die AutoOCR Installation erfolgreich waren macht es Sinn einen „In“ Folder zu definieren und eine Test Datei zu verarbeiten um zu sehen ob die AutoOCR Basisfunktionen problemlos funktionieren.

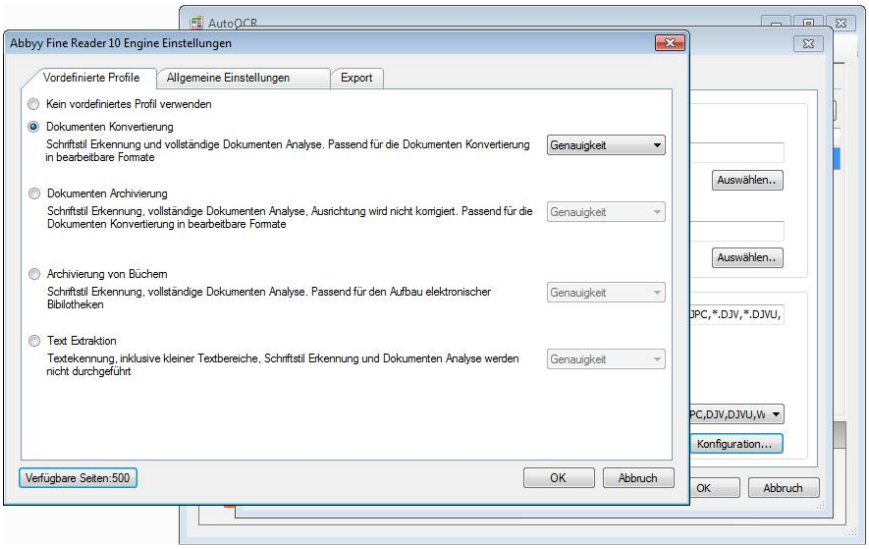
Test-Folder anlegen:

Legen Sie für diesen Zweck z.B. 3 Folder am Desktop an – einen „In“ einen „Out“ und einen „Error“ Folder. Fügen Sie in AutoOCR einen neuen „Eingangs Ordner“ hinzu.

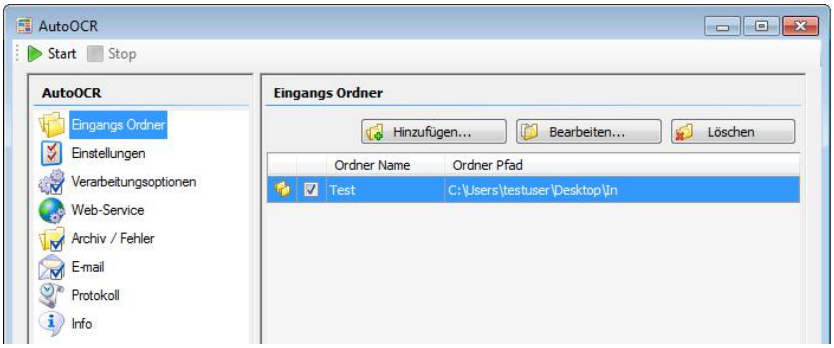


Wählen Sie **„Konfiguration“** um die OCR Engine Konfiguration zu anzuzeigen und die Anzahl der verfügbaren Seiten zu überprüfen. Durch die Funktion **„Verfügbare Seiten:“** werden diese angezeigt. 500 z.B. für die Demolizenz – aber normalerweise z.B. 10.000. Das ist z.B. die pro Monat verfügbare Seitenzahl – Am Beginn jedes Monats wird der Zähler dann wieder zurückgesetzt und es steht die volle Seitenzahl wieder zur Verfügung. Die Zahl bezieht sich auf A4 Seiten – werden z.B. größere Formate verarbeitet so wird die entsprechende Zahl an A4 Seiten abgezogen – bei A3 z.B. 2 statt 1. Es wird auch dann eine Seite abgezogen falls eine „sehr kleine Seite“ z.B. ein Thumbnail mit 70 x 100 Pixel (was natürlich für OCR sinnlos ist) verarbeitet wird.

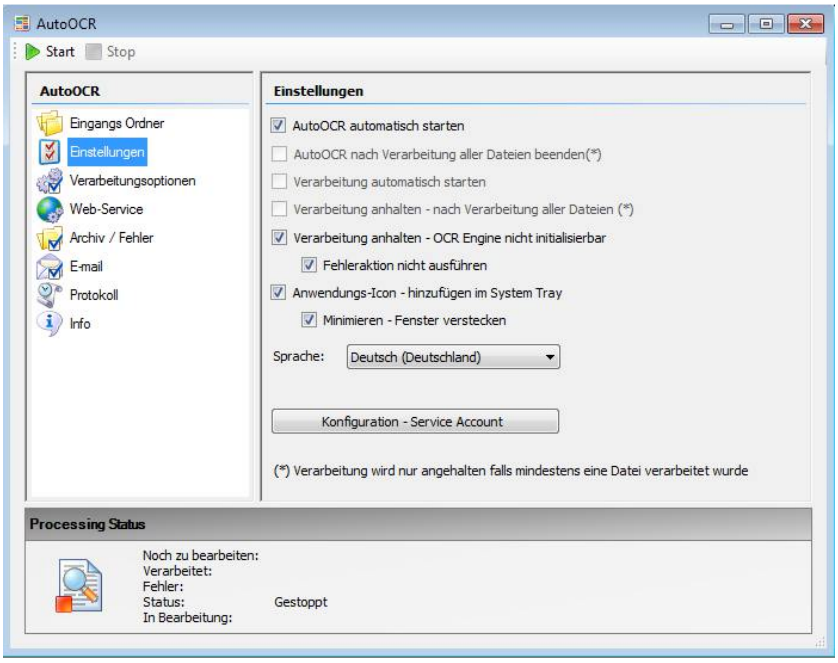
Wird hier „0“ angezeigt so stehen entweder keine Seiten mehr zur Verfügung oder die Abbyy OCR Engine kann nicht angesprochen werden. In dem Fall können keine Dokumente verarbeitet werden.



Für den ersten Test können die Standard-Einstellungen der Abbyy OCR Engine belassen werden.



Öffnen und Setzen Sie nun die **„Einstellungen“**

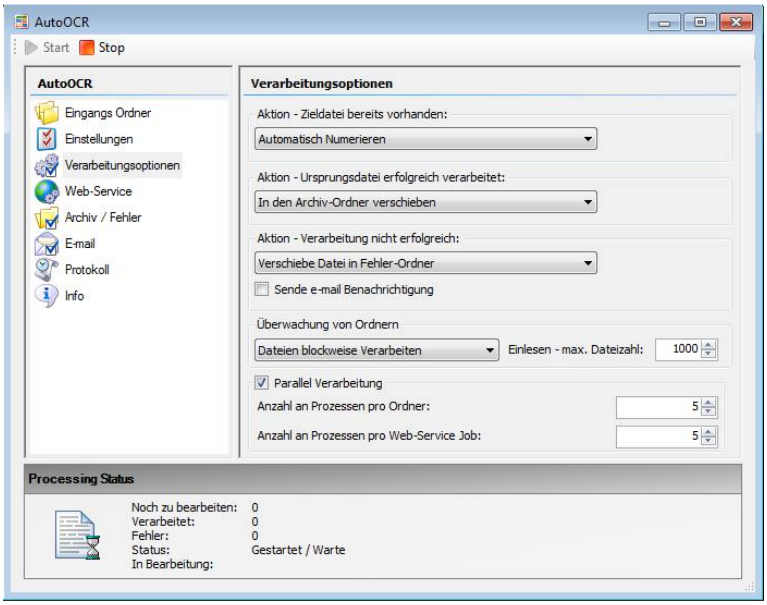


„AutoOCR automatisch starten“ – dabei wird mit dem Rechnerstart auch die Verarbeitung von AutoOCR aktiviert. Die anderen Optionen sind ausgegraut und nur verfügbar falls die Anwendung nicht als Service installiert wurde.

„Anwendungs Icon im System Tray“ – Damit kann das UI versteckt werden und nur ein kleines Icon im System Tray bleibt erhalten – Das UI kann jedoch jederzeit wieder aktiviert und angezeigt werden – Doppelklick auf das Icon im System Tray oder rechte Maustaste „Öffnen“

„Konfiguration – Service Account“ ist dann wichtig falls der AutoOCR Service unter einem bestimmten Service Account User laufen soll. Standardmäßig wird der AutoOCR Service unter dem lokalen System-Account betrieben. Weitere Informationen dazu finden sie hier: http://www.pdfblog.at/?p=1688

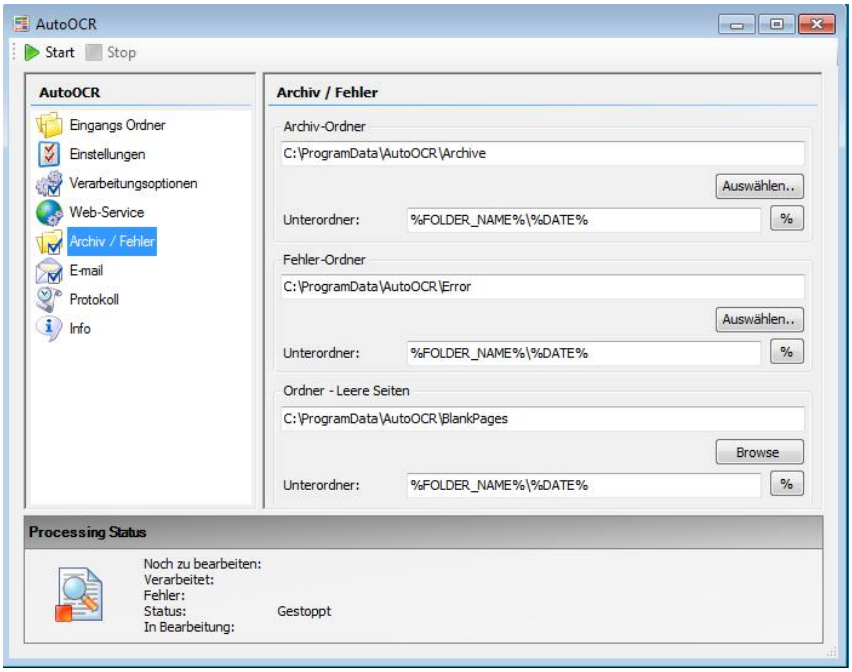
Öffnen und Setzen Sie nun die **„Verarbeitungsoptionen“**



„Parallel Verarbeitung“ – aktivieren Sie diese Option falls der Rechner über entsprechende Ressourcen verfügt. Ein Multicore Prozessor und mind. 4GB RAM – ermöglichen es mehrere OCR Verarbeitungstasks parallel zu starten und Dokumente parallel zu verarbeiten. Eine parallele Verarbeitung kann eine wesentliche Steigerung des Durchsatzes bringen aber nur falls die entsprechenden Ressourcen dafür zur Verfügung stehen. „5“ Prozesse sind die Standardeinstellung – je nach Anzahl der CPU – Cores und verfügbarem Hauptspeicher kann die Zahl auch erhöht werden. Eine Optimale Nutzung der vorhandenen Ressourcen muss aber immer erst evaluiert werden.

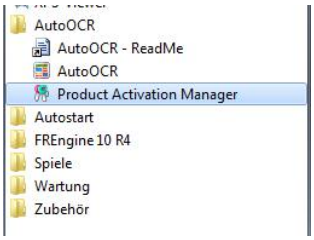
Einstellungen für Archiv und Error Folder:

Für einen ersten Test können diese Einstellungen so beibehalten oder auf andere von Ihnen angelegte Folder gesetzt werden.

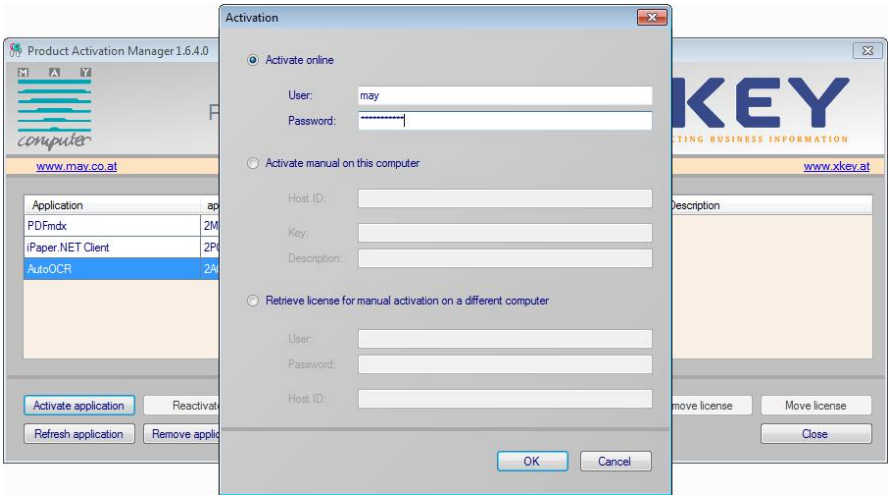


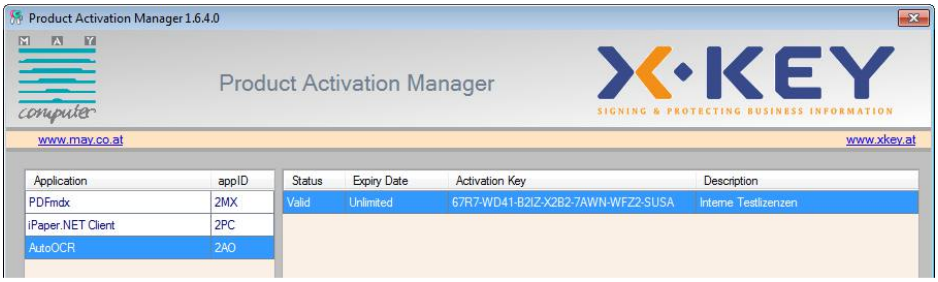
Aktivieren der AutoOCR Lizenz

Aktivieren Sie die AutoOCR Lizenz über den Menüpunkt „Info“ und „Aktivieren“ oder durch Start der Anwendung PAM – Product Activation Manager. Sie erhalten dafür von uns einen Usernamen und ein Password. Zur Aktivierung ist eine aktive Internetverbindung erforderlich.

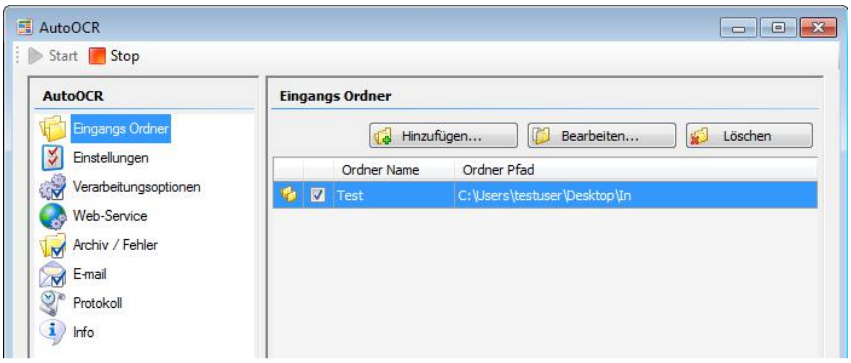


Wählen Sie „AutoOCR“ in der Liste der Anwendungen aus und führen Sie die Funktion „Activate application“ aus. Danach geben Sie den „Usernamen“ und das „Password“ ein. Danach kann die für Sie angelegte Lizenz auswählen.





„Starten“ Sie den AutoOCR Server



Um ein optimales OCR Ergebnis zu erzielen sollten die Images folgende Merkmale aufweisen:

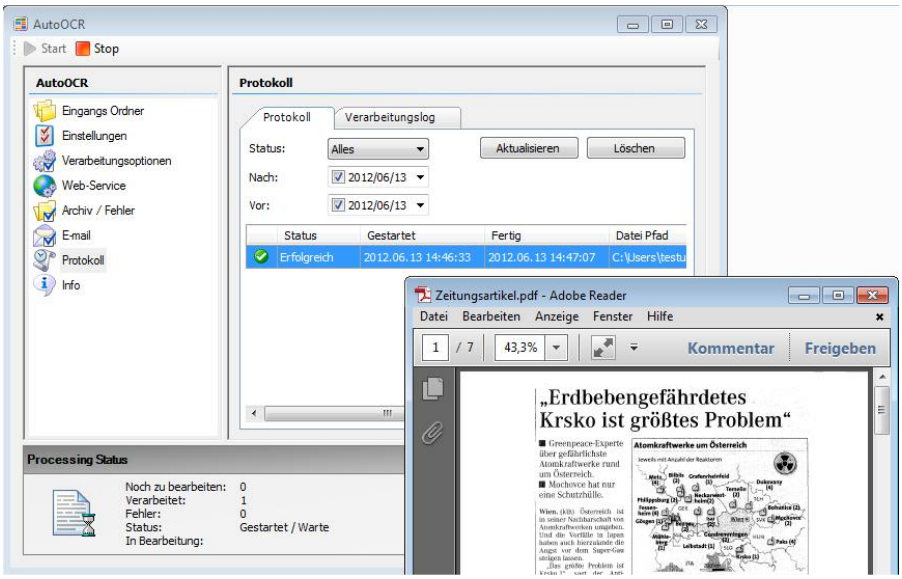
Schwarz-Weiß: TIFF Gruppe 4 Komprimierung, 300dpi
Farbe / Graustufen: 200 bis max. 300dpi, JPEG Komprimierung

Mehr als 300dpi machen keinen Sinn und bei SW sollte die Auflösung nicht unter 300dpi sein. Bei Farbscans reichen in den meisten Fällen auch 200dpi aus.

Eine 7 Seiten Testdatei – TIFF Gr4, 300dpi, Multipage können Sie von hier herunterladen:
http://www.may.co.at/webrr/respfile.php?file=MAYComp/Current/RO/Zeitungsartikel.tif

Kopieren Sie eine Testdatei in den zuvor angelegten Eingangsfolder – die OCR Verarbeitung startet automatisch und zeigt den aktuellen Verarbeitungsstatus im unteren Bereich des Benutzerinterfaces entsprechend an.

Wählen Sie den Punkt „Protokoll“ an bzw. öffnen Sie den Out Folder um die PDF-OCR Datei anzuzeigen.



Nun ist sichergestellt dass der AutoOCR Server richtig installiert ist und Dateien problemlos verarbeitet werden können.

Falls Sie in weitere Folge keine folderbasierende Verarbeitung benötigen können Sie den angelegten Eingangsfolder wieder deaktivieren in den Sie die Checkbox bei der Eingangsfolder Definition entfernen – AutoOCR Server „Stop“ – Checkbox „Eingangsordner“ in der Liste entfernen.

Der nächste Schritt ist die Konfiguration der Web-Service Schnittstelle für die Anbindung und Integration mit dem ifresco AutoOCR Transformer für Alfresco.

AutoOCR – Web-Service Konfiguration

„Stoppen“ Sie dazu zuerst den AutoOCR Server und wählen Sie den Punkt „Web-Service“

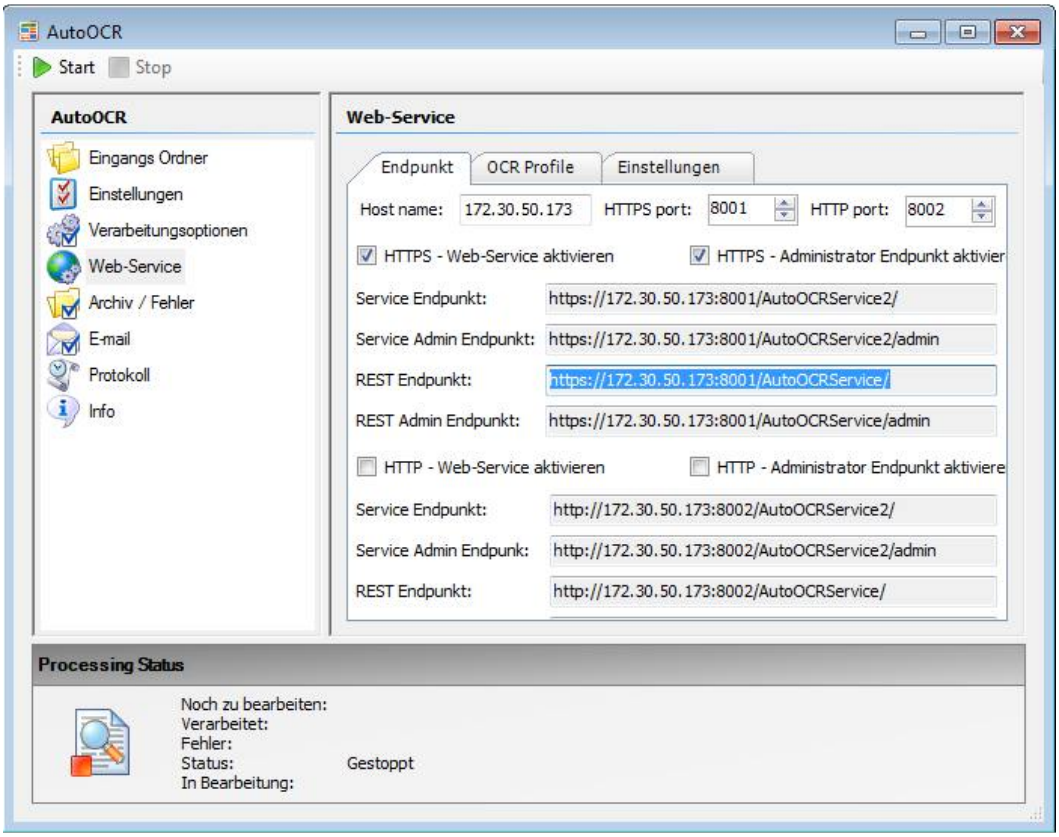
Um die Web-Service Schnittstelle von AutoOCR ansprechen zu können müssen Sie die „Service Endpunkte“ konfigurieren – das sind Adressen über den der AutoOCR Server angesprochen wird. Die Kommunikation erfolgt entweder über HTTP bzw. gesichert über HTTPS. Standardmäßig werden die **Ports 8001 für HTTPS** bzw. **8002 für HTTP** verwendet welche entsprechend freigeschalten sein müssen damit der Rechner über diese erreichbar ist. Konfigurieren bzw. Deaktivieren Sie die Firewall entsprechend auf dem Rechner.

Ermitteln Sie die IP Adresse des Rechners auf dem der AutoOCR Server installiert wurde (ipconfig). Alternativ kann auch ein Name verwendet werden – es muss jedoch sichergestellt sein dass der Rechner unter diesem Namen auch erreichbar ist. Verwenden Sie für die ersten Tests sicherheitshalber die IP Adresse des Rechners.

Tragen Sie die IP Adresse im Feld „Host Namen:“ ein – der Wert wird automatisch in die andere Felder übernommen. Alle anderen Einstellungen können unverändert belassen werden.

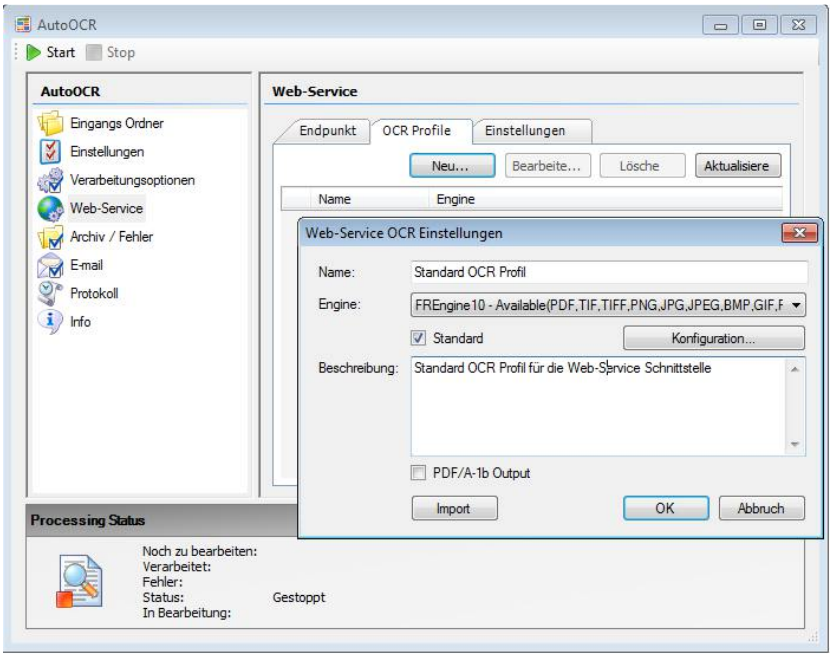
Für die spätere Konfiguration des ifresco AutoOCR Transformers benötigen Sie die URL Adresse des REST Endpunktes:

z.b.: https://172.30.50.173:8001/AutoOCRService/

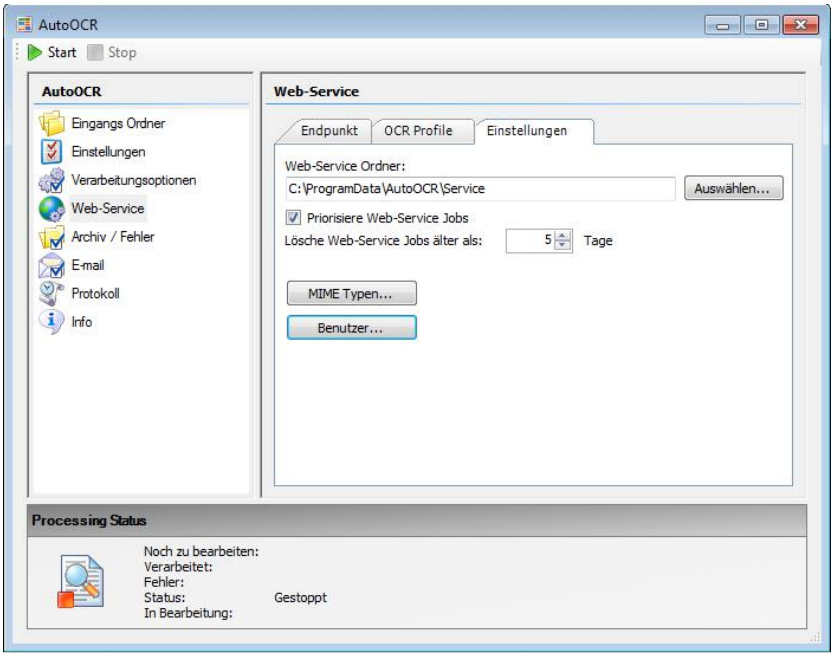


OCR Profil anlegen:

Als nächster Schritt muss eine OCR Profil angelegt werden – Ein OCR Profil fasst alle möglichen Einstellungen und Optionen der OCR Engine unter einem Profilname zusammen. Für den Anfang können die Standardeinstellungen „Konfiguration“ unverändert belassen werden. Wichtig – Es muss jedenfalls ein OCR Profil angelegt werden und vorhanden sein sonst ist eine nachfolgende Konfiguration des ifresco AutoOCR Transformers nicht möglich.



Web-Service – Einstellungen:

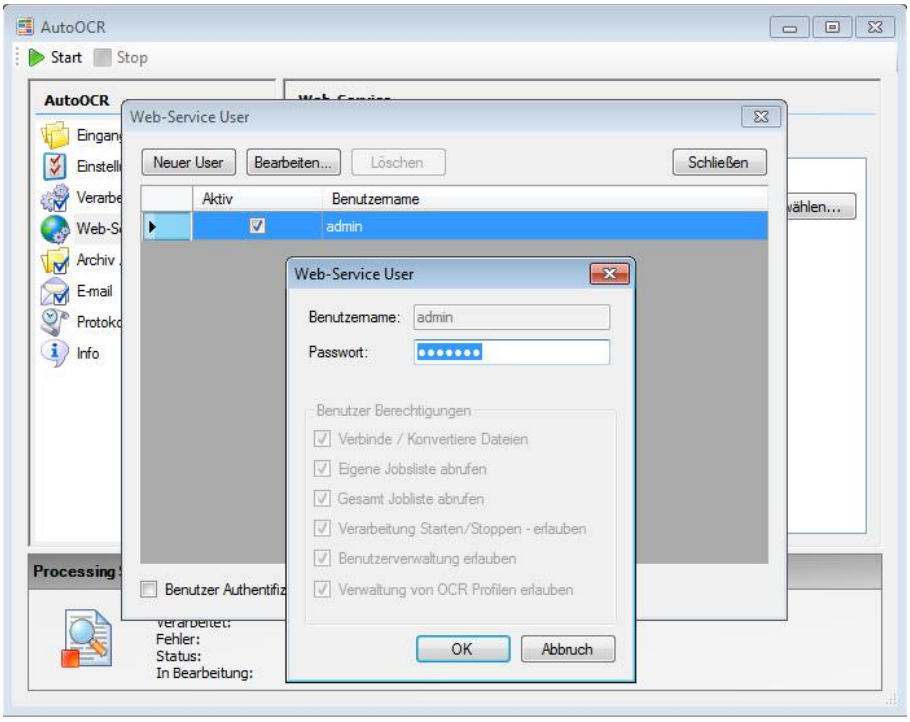


AutoOCR ist in der Lage sowohl Dateien aus Foldern zu verarbeiten als auch Aufträge über die Web-Service Schnittstelle entgegenzunehmen. Mit der Option „Priorisiere Web-Service Jobs“ kann festgelegt werden dass die Web-Service Verarbeitung höher priorisiert wird. Der Web-Service Folder wird für die Zwischenspeicherung der Web-Service Jobs verwendet, wobei hier eine automatische Bereinigung des Folders nach der eingestellten Anzahl an Tagen erfolgt.

Web-Service - Benutzer:

Für die HTTPS Kommunikation ist jedenfalls immer ein Benutzer erforderlich – Standardmäßig ist bereits ein „admin“ User angelegt der über alle Rechte verfügt. Ein solcher User ist in weitere Folge auch für die Konfiguration des ifresco Transformers erforderlich. Es kann entweder der Standarduser verwendet oder ein eigener User angelegt werden.

User: admin
Password: autoocr



Damit ist die Web-Service Konfiguration abgeschlossen und der Server kann wieder mit „Start“ gestartet werden.

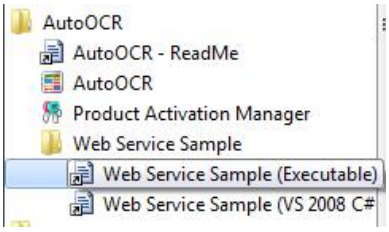
Schritt 3: Installation – Web-Service Test Client (Optional)

Um die Web-Service Kommunikation testen zu können steht ein Setup mit einem Microsoft - Windows Testclient zur Verfügung. Damit kann von jedem beliebigen Windows Rechner aus die Web-Service Kommunikation mit dem AutoOCR Server überprüft und ausgetestet werden.

Web-Service Testclient - Download Setup:
http://www.may.co.at/webrr/respfile.php?file=MAYComp/Current/RO/AutoOCRWebServiceSample.zip

Setup.exe als „Administrator“ ausführen.

Web-Service Sample (Executable) aufrufen



Geben Sie bei „AutoOCR Service“ die „Service Endpunkt“ Adresse des AutoOCR Web-Servers (nicht die REST Endpunkt Adresse) ein und drücken Sie „Connect“

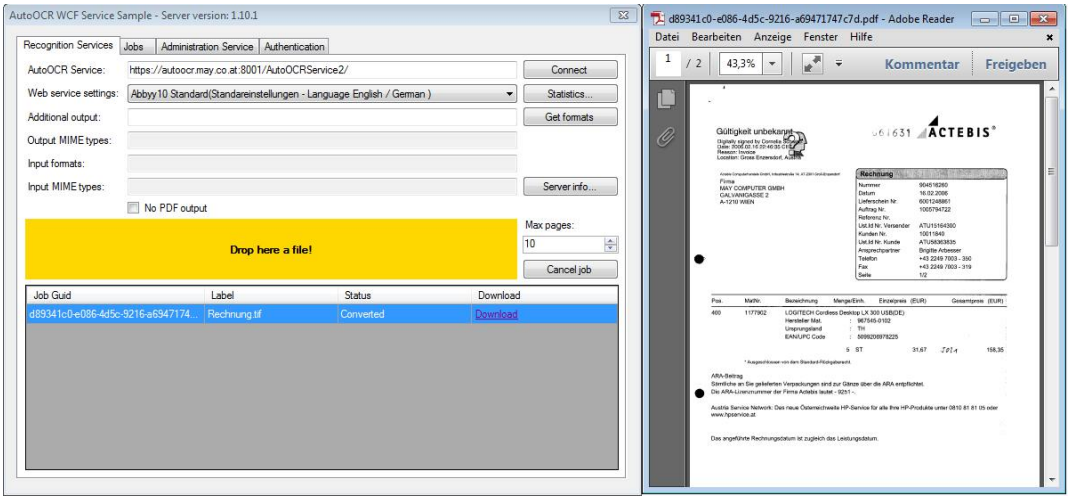
z.b. : http://172.30.50.174:8002/AutoOCRService2/
oder http://localhost:8002/AutoOCRService2/
oder http://autoocr.may.co.at:8002/AutoOCRService2/

Unter „Web service settings“ können Sie die am AutoOCR Server angelegten OCR Profile auswählen. Alle konfigurierten Profile werden hier als drop down Liste angezeigt. Das aktuell ausgewählte wird für die Verarbeitung der übertragenen Datei verwendet.

Um nun die Web-Service Verarbeitung mit dem AutoOCR Server testen zu können ziehen Sie eine Testdatei z.B. eine TIFF Datei in den gelben Bereich „Drop here a file“

Beispieldatei:
http://www.may.co.at/webrr/respfile.php?file=MAYComp/Current/RO/Zeitungsartikel.tif

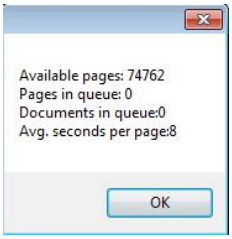
Die Datei wird an den AutoOCR Server übertragen und in ein durchsuchbares PDF konvertiert. In der Statusspalte wird der aktuelle Status der Verarbeitung angezeigt – Bei „Converted“ erscheint in der Spalte daneben ein „Download Link“. Über den Link kann die PDF Datei vom Server abgerufen und angezeigt werden.



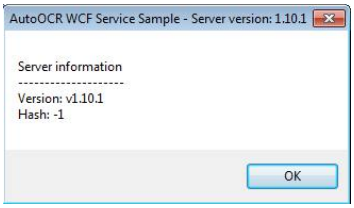
Damit ist der Test der Web-Service Verarbeitung abgeschlossen – uns es ist sichergestellt dass die für den ifresco OCR Transformer erforderliche Schnittstelle funktioniert.

Weitere Funktionen:

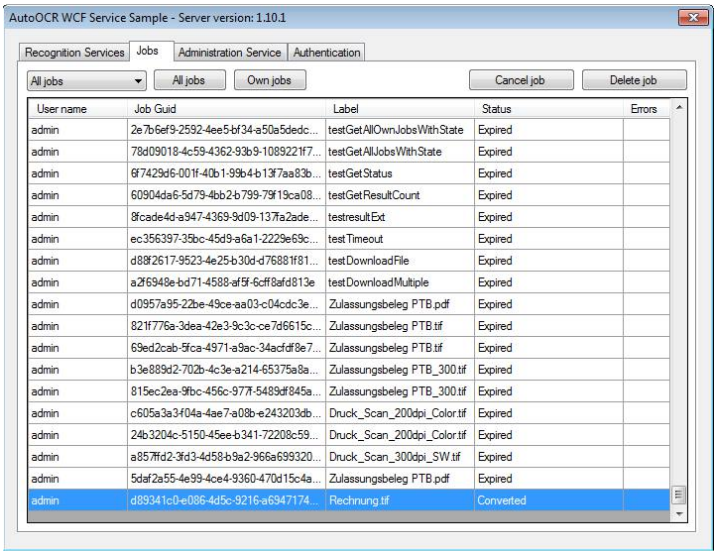
Statistik – z.B. Zeigt die noch verfügbaren Seiten der OCR Lizenz



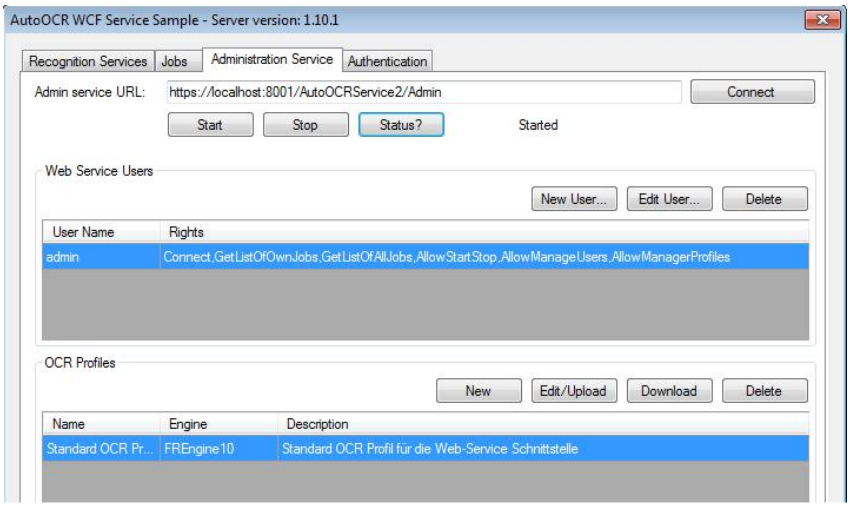
Server-Info – Zeigt die Versionsnummer und den Hash-Code des Servers:



Jobs-Liste – Zeigt die Jobs-Liste des AutoOCR Servers und den Status



Admin – Über eine eigene Endpunkt Adresse können Admin Funktionen ausgeführt werden – Server Start / Stop, Status, User, OCR Profile



Authentification – Username und Password für die HTTPS Authentifikation – Standardmäßig wird auch hier der **User: „admin“** und das **Password: „autoocr“** verwendet.

Schritt 4: Installation – ifresco AutoOCR Transformer

Allgemein

Die wichtigsten Features und Funktionen:

1. Dokument Aktion in Alfresco Share zur Transformation, mit und ohne AutoOCR
2. AutoOCR Alfresco Transformer, zur Einbindung der von AutoOCR angebotenen Transformation in Alfresco. Der Transformer kann zur Laufzeit dynamisch über eine Share-Admin Oberfläche konfiguriert werden.
3. JavaScript-Client für den AutoOCR Service, nutzbar in Alfresco Repository-Scripts (WebScripts, Actions, usw.)
4. Java-Client für den AutoOCR Service, zur Verwendung in Java Code (Alfresco oder sonstige Umgebungen). Der Java Client selbst hat keine Abhängigkeiten gegen Alfresco.

ifresco AutoOCR Transformer Konfiguration

Der Transformer greift auf eine Konfigurationsdatei im Alfresco Data-Dictionary zurück (/Data Dictionary/May Computer/AutoOCR/autoocr.config). Sie ist im JSON Format hinterlegt und kann über das Share-Admin GUI verändert werden.

Damit das Modul auch grundsätzlich ohne Share funktionieren kann, ist es jedoch genauso möglich die Konfigurationsdatei manuell zu editieren. Die Änderungen sind nach dem speichern sofort aktiv.

Beispiel Konfigurationsdatei:

```
{
  "enabled": true,
  "endpoint": "https://123.123.123.123:8001/AutoOCRService",
  "username": "OCR",
  "password": "****",
  "apiKey": 12345,
  "sleeptime": 1000,
  "timeout": 240000,
  "connectiontimeout": 10000,
  "transformations": [
    {
      "autoOCRExtension": "PDF",
      "settingsName": "AbbyyFR10",
      "sourceMimetype": "application/pdf",
      "targetMimetype": "application/pdf"
    },
    {
      "autoOCRExtension": "PDF",
      "settingsName": "Image2PDF",
      "sourceMimetype": "image/png",
      "targetMimetype": "application/pdf"
    }
  ]
}
```

Installation und Konfiguration

Aktuell unterstützte Alfresco Versionen:

- Alfresco CE – Version 4.0d
- Alfresco EE – Version 4.0 und 4.0.1

Die Installation erfolgt nach dem AMP Modell.

Es gibt ein AMP für das alfresco.war (Repository) und ein AMP für das share.war.

Bitte beachten Sie dass es je nach verwendeter Alfresco Version – CE – Community Edition sowie EE-Enterprise Edition – sowie auch je nach Versionsnummer unterschiedliche AMP Packages gibt. Bitte stellen Sie sicher dass Sie die richtigen AMP Packages für Ihre Alfresco Version für die Installation verwenden.

- Alfresco herunterfahren, **Backup** nach Bedarf.
- Ablage des *-repo.amp in ALF_HOME/amps
- Ablage des *-share.amp in ALF_HOME/amps_share
- Kontrolle der Verzeichnisse, ob nicht alte Versionen der amps noch dort liegen.
- Aufruf von bin/apply_amps.sh installiert die Module.
- Alfresco starten
- Die Konfiguration unter http://ALF_HOST:ALF_PORT/share/page/console/admin-console/autoocradmin vornehmen.

Beispiel:

Das *-repo.amp in das AMPS Verzeichnis kopieren:

```
/opt/alfresco/amps/
```

Das *-share.amp in das SHARE AMPS Verzeichnis kopieren:

```
/opt/alfresco/amps_share/
```

Anschließend mit der Command Line in das Alfresco BIN Verzeichnis wechseln:

```
cd /opt/alfresco/bin/
```

Ausführen des AMP Installer Scripts

```
./apply_amps.sh
```

Es sollte eine ähnliche Ausgabe kommen die man einfach mit einer beliebigen Taste bestätigt:

```
This script will apply all the AMPs in amps and amps-share to the
alfresco.war and share.war files in /opt/alfresco/tomcat/webapps
Press control-c to stop this script . . .
Press any other key to continue . . .
```

```
Press control-c to stop this script . . .
Press any other key to continue . . .

Module 'ifresco_autoocr' installed in '/opt/alfresco/tomcat/webapps/alfresco.war
'
   -    Title:        Alfresco Module ifresco_autoocr
   -    Version:      1.12
   -    Install Date: Thu Jun 14 13:34:41 CEST 2012
   -    Description:   Alfresco Module ifresco_autoocr
Module 'org.alfresco.module.vti' installed in '/opt/alfresco/tomcat/webapps/alfr
esco.war'
   -    Title:        Vti
   -    Version:      1.2
   -    Install Date: Thu May 24 11:50:23 CEST 2012
   -    Description:   Alfresco Vti Extension
Module 'ifresco_autoocr' installed in '/opt/alfresco/tomcat/webapps/share.war'
   -    Title:        Alfresco Module ifresco_autoocr
   -    Version:      1.12
   -    Install Date: Thu Jun 14 13:34:55 CEST 2012
   -    Description:   Alfresco Module ifresco_autoocr
About to clean out /opt/alfresco/tomcat/webapps/alfresco and share directories a
nd temporary files...
Press control-c to stop this script . . .
Press any other key to continue . . .
```

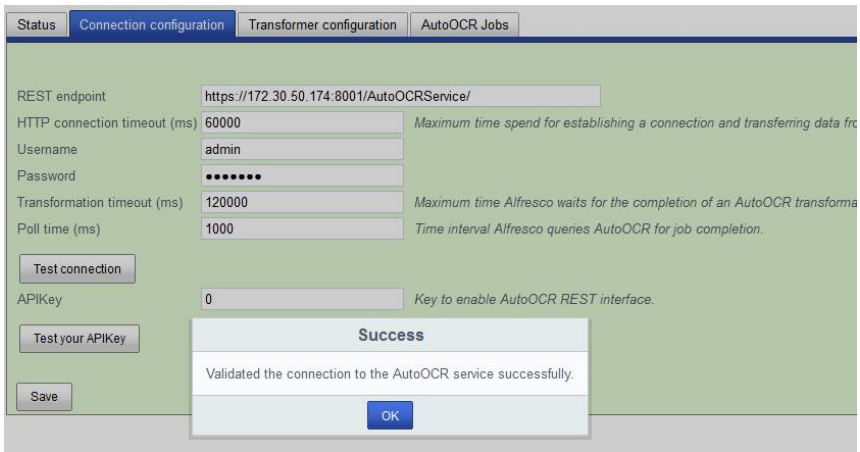
Alfresco neu starten

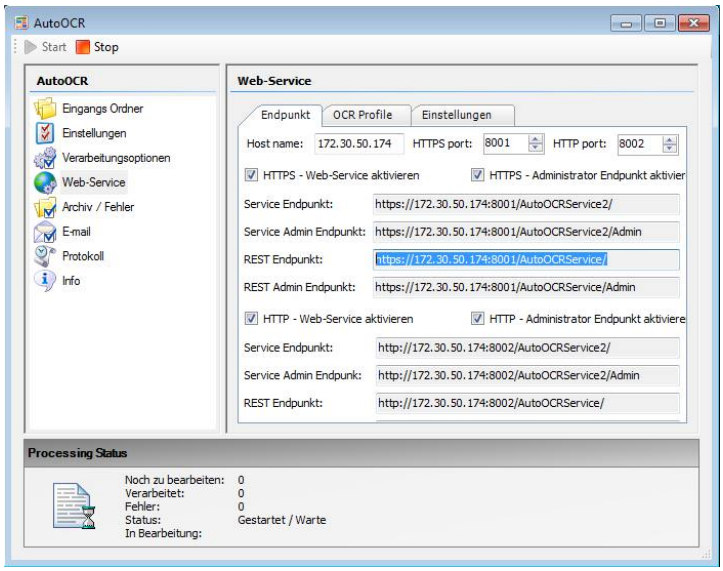
```
cd /opt/alfresco/
./alfresco.sh restart
```

Konfiguration - ifresco AutoOCR Transformer über das Share UI

Aufruf der Konfiguration über die Share Oberfläche als Admin unter „More, More, AutoOCR“ oder über die direkte URL http://ALF_HOST:ALF_PORT/share/page/console/admin-console/autoocradmin z.b.: http://172.30.50.152:8080/share/page/console/admin-console/autoocradmin

Tab – „Connection Configuration“ auswählen



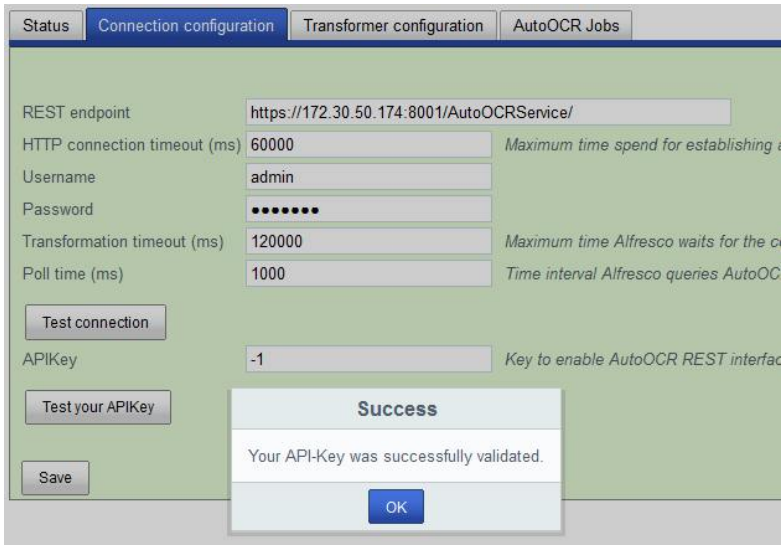


Tragen Sie die REST Endpunktadresse des AutoOCR Servers unter „REST Endpoint“ ein.

Tragen Sie unter „Username“ und „Password“ den User und das Passwort für den User ein über den die Kommunikation mit dem AutoOCR Server laufen soll. Standardmäßig ist das der **User „admin“** mit dem **Passwort: „autoocr“**

Über die Funktion „Test connection“ kann die Kommunikation zwischen den beiden Servern überprüft werden.

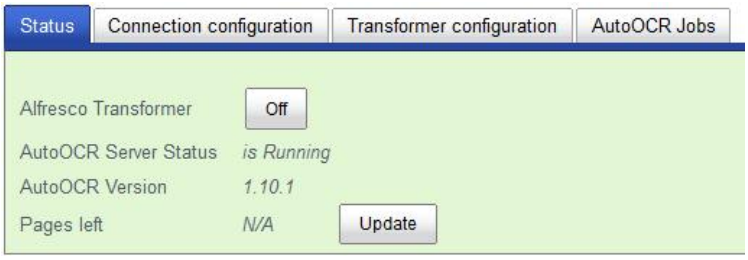
Tragen Sie danach unter „APIKey“ entweder „-1“ oder den von uns zur Verfügung gestellten API Schlüssel hier ein. Testen Sie anschließend mit der Funktion „Test your APIKey“ ob der von Ihnen eingegebene Wert vom Server akzeptiert wird.



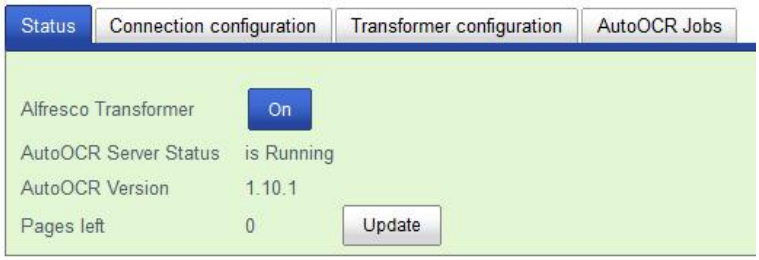
Speichern Sie anschließend die Einstellungen mit der Funktion „Save“

Anschließend loggen Sie sich aus Alfresco Share aus und wieder ein

Rufen Sie jetzt wieder die Konfiguration auf - der Status Tab sollte nun folgende Information zeigen:



Aktivieren Sie nun den Alfresco Transformer – Button Label wechselt von „Off“ auf „On“. Drücken Sie den Button „Pages left“ – Die Anzeige wird nun „0“ sein. Das ist normal und wird erst aktualisiert sobald das erste Dokument verarbeitet wurde.



Alfresco Transformer Konfiguration:

Alfresco arbeitet intern mit Transformern – damit ist es möglich ein Format in ein anderes Format zu konvertieren. Standardmäßig sind bei der Installation bestimmte Transformer bereits installiert und für die Funktion von Alfresco auch erforderlich.

Alfresco verwendet z.b. Transformer auch intern um aus einem Dokument den Text für die Volltextindexierung zu extrahieren – z.b. aus einem DOC, RTF oder TXT. Der daraus gewonnene Text wird in den Indexer eingefügt und dient dazu die Dokumente bei der Volltextsuche finden zu können.

Zu diesem Zweck „sucht“ Alfresco beim Hochladen eines neuen Dokuments zuerst einen passenden Text-Transformer – ist ein solcher definiert und vorhanden so wird er verwendet. Das funktioniert „automatisch“ im Hintergrund ohne dass der Anwender eine Funktion ausführen muss.

Alfresco verwendet aber auch Transformer um z.b. die Dokumentenvoransicht zu erzeugen. Für Thumbnails werden JPEG und für die normale Dokumentenvoransicht SWF Dateien erzeugt. Bei SWF wird in vielen Fällen der Zwischenschritt über PDF gemacht. Z.b. DOC > PDF > SWF

AutoOCR unterstützt verschiedene Input und Output Formate

- **Input:** PDF, TIFF, PNG, JPEG, BMP, GIF, PCX, DCX, JP2, JPC, DJV, DJVU, WPD
- **Output:** PDF, DOC, DOCX, RTF, XLS, XLSX, TXT, HTM, HTML, XML, PPT

Für jedes dieser Input Formate kann eine Transformer in ein Ausgabeformat konfiguriert werden.

Achtung: Bestimmte Transformer Konfigurationen werden durch Alfresco automatisch verwendet sobald vorhanden – Konfigurieren sie also einen **PDF > TXT Transformer** über AutoOCR so wird jedes PDF Dokument das ins Alfresco kommt oder intern in Alfresco in eine PDF konvertiert wird durch diesen Transformer geschickt um den Volltext für die Suchengine daraus zu gewinnen. Egal ob das PDF bereits einen Text enthält oder nicht. Das gleiche gilt für alle anderen Transformer Konfigurationen nach TXT. Ist ein Transformer für ein Format „nach TXT“ vorhanden so wird ein Dokument automatisch über diesen Transformer geschickt sobald ein solches Dokument im Alfresco abgelegt wird. Damit werden sowohl OCR-Seitenlizenzen verbraucht, als auch Last und Traffic am OCR Server erzeugt. Das kann bewusst gewollt sein (z.B. TIFF > TXT) oder aber auch nicht.

Wichtig: Bitte konfigurieren Sie mit der aktuellen Version keinen Transformer PNG nach TXT. Alfresco erzeugt für jedes Dokument das ins Repository kommt ein Thumbnail mit 77 x 100 Pixel und falls ein PNG > TXT Transformer konfiguriert wurde wird dieser benutzt – d.h. jedes Thumbnail wird dann durch den Transformer geschickt. Daraus kann kein OCR Ergebnis erzeugt werden – da das Image zu klein und nieder auflösend ist, außerdem macht es keinen Sinn für ein Thumbnail nochmals eine Texterkennung zu machen wo diese bereits für das Dokument erzeugt wurde. Die Konfiguration PNG > TXT erzeugt als nur Last am OCR Server und es werden „teure“ Abbyy OCR Seitenlizenzen verschwendet.

Ebenso wichtig zu wissen: Wir ein Dokument im Alfresco gelöscht so wird dieses Dokument (falls dafür ein TXT Transformer definiert wurde) ebenfalls wieder über diesen Transformer verarbeitet. Das Dokument wird aus dem Arbeitsbereich in den „Mistkübel“ Bereich verschoben und wird dort ebenfalls neu indexiert und daher extrahiert Alfresco nochmals den Textinhalt. D.h. Sie dürfen sich nicht wundern dass falls ein TXT Transformer für diese Dokumente konfiguriert wurde beim Löschen die OCR Verarbeitung nochmals durchgeführt wird.

Mit der Konfiguration eines Transformers nach TXT muss man also bewusst und überlegt vorgehen.

Die Integration des AutoOCR Servers mit Alfresco kann als entweder dazu verwendet werden um aus Imagedokumenten die TXT Information für die Volltextsuche zu erzeugen, dabei bleibt das Dokument in seinem Ursprungsformat, oder man konvertiert die Dokumente in durchsuchbare PDF´s. Die damit erzeugten PDF Dokumente enthalten sowohl das Image als auch die dem Image hinterlegte Volltextinformation. Aus solchen PDF Dokumenten extrahiert dann Alfresco über den intern vorhandenen PDF>TXT Transformer den Volltext für die Suche.

Die „OCR“ - Transformation nach PDF (TIF>PDF, JPEG>PDF oder PDF>PDF) – kann über Regeln auf Folder, über JavaScripts bzw. auch über das Share User Interface angestoßen werden.

Mit der „Transformer configuration“ wird die Default Transformer Konfiguration von Alfresco zur Laufzeit dynamisch überschrieben. Die hier konfigurierten MIME-Typen für Quelle und Ziel werden über den AutoOCR Server mit dem ausgewählten Profil verarbeitet. Je nach Profil können mehr oder weniger MIME-Typen zur Auswahl stehen – abhängig welche von der dahinterliegenden Engine unterstützt werden.

Status | Connection configuration | Transformer configuration | AutoOCR Jobs

Current registered transformations for AutoOCR:

AutoOCR Profile	Source mimetype	Target mimetype	
Standard OCR Profil	application/pdf	application/pdf	Delete
Standard OCR Profil	image/tiff	application/pdf	Delete
Standard OCR Profil	image/png	application/pdf	Delete
Standard OCR Profil	image/jpeg	application/pdf	Delete
Standard OCR Profil	image/gif	application/pdf	Delete
Standard OCR Profil	image/pcx	application/pdf	Delete
Standard OCR Profil	image/dcx	application/pdf	Delete
Standard OCR Profil	image/jp2	application/pdf	Delete
Standard OCR Profil	image/vnd.djvu	application/pdf	Delete
Standard OCR Profil	image/tiff	text/plain	Delete
Standard OCR Profil	image/tiff	application/msword	Delete

Register a new transformation for AutoOCR:

AutoOCR Profile: Standard OCR Profil - Standa | Source mimetype: image/tiff | Target mimetype: application/msword | Add transformation

Save

Transformer – Job Liste:

Die Jobliste zeigt mit der Filtermöglichkeit über den Status die Jobs des AutoOCR Servers an.

Status | Connection configuration | Transformer configuration | AutoOCR Jobs

Please select a state and click "Refresh Table" Converted Refresh Table

First Previous 1 Next Last

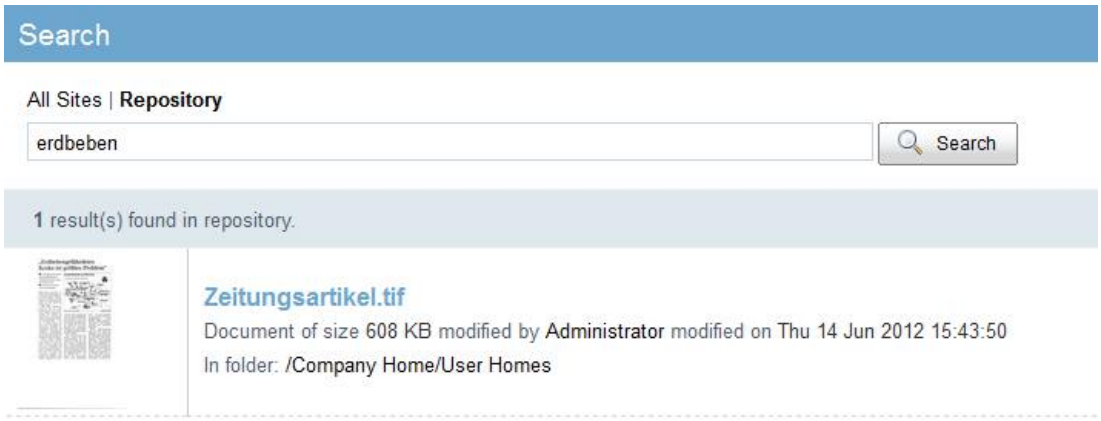
Label	Job GUID	Status	Page count	Owner	Error	Actions
Zeitungsartikel.tif	5222a1ac-098b-47a5-9c0e-b5f25dc2d376	CONVERTED	7			
Rechnung.tif	925de507-1d7e-422d-a100-e0efa57d19f9	CONVERTED	2			

First Previous 1 Next Last

Alfresco und AutoOCR – die verschiedenen Einsatzmöglichkeiten

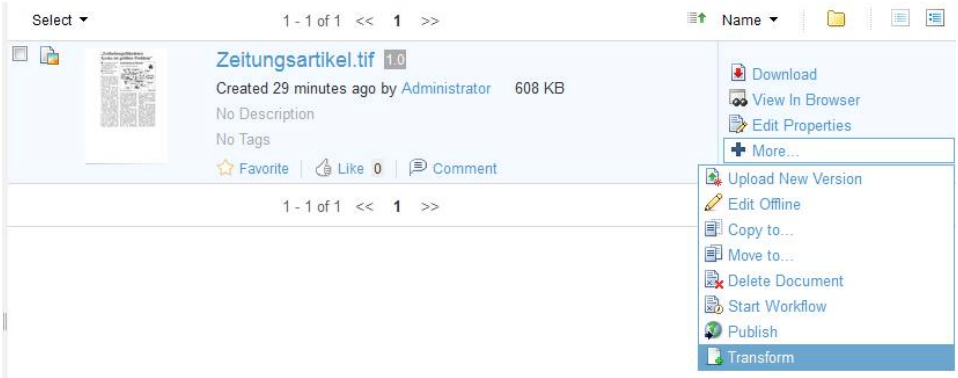
TIFF-Scan - Volltextsuche

Wie in der oben gezeigten Konfiguration haben wir einen AutoOCR Transformer von TIFF nach TXT konfiguriert. Dabei wird aus jedem TIFF Dokument das in das Alfresco Repository geladen wird über den AutoOCR Server eine TXT Datei erzeugt und diese in den Volltext von Alfresco eingefügt. Damit ist kann das TIFF Dokument unmittelbar nach dem Upload bereits per Suche über den Textinhalt gefunden werden. Dabei hat der Anwender selbst keine Aktion auszuführen.



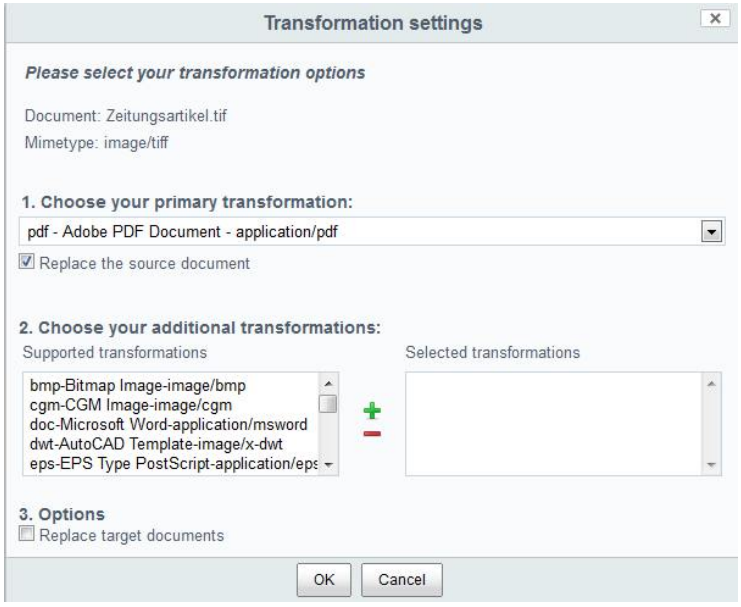
Share – Dokumentenaktion „Transform“

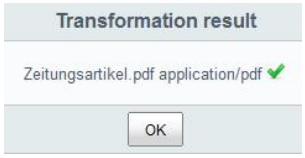
Über die Share Integration des ifresco AutoOCR Transformers steht auch eine interaktive Möglichkeit zur Dokumentenkonvertierung zur Verfügung. Dabei können nicht nur die AutoOCR Transformer sondern auch alle anderen in Alfresco zur Verfügung stehenden Transformer genutzt werden. Share wird damit um eine allgemeine Dokumenten Transformer Funktion erweitert.

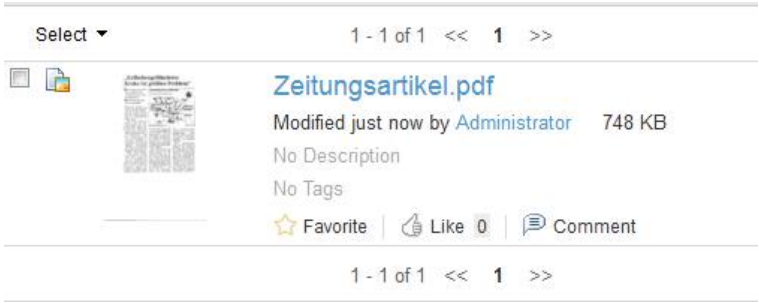


TIFF Scan Image durch PDF-OCR ersetzen

Dabei wird das vorhandene TIFF Image Dokument durch das OCRte PDF ersetzt.

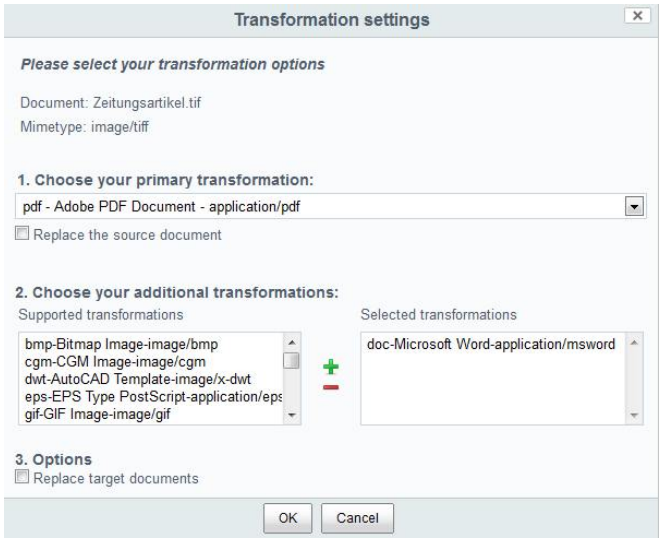


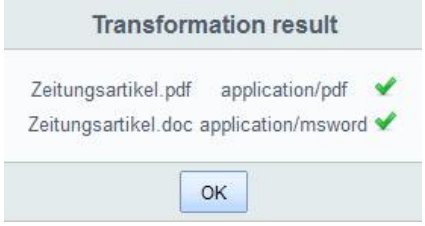


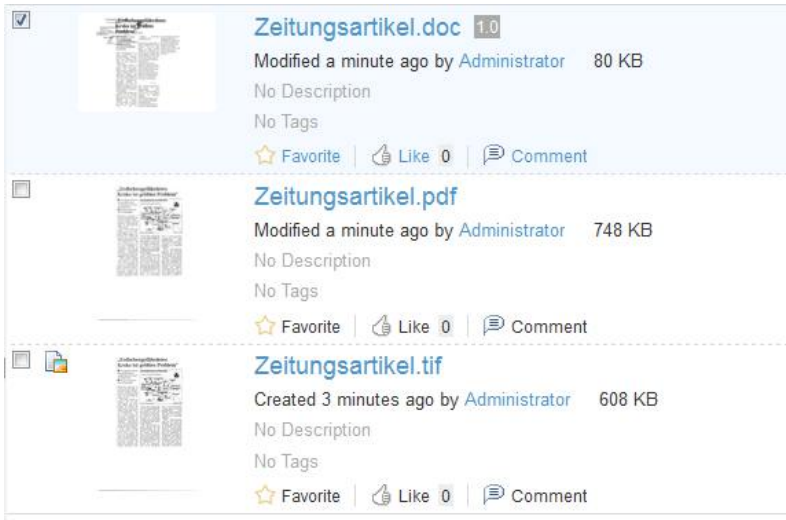


TIFF Scan Image zusätzliche Formate (PDF, DOC) erzeugen

Images können durch das OCR Ergebnis ersetzt bzw. es können auch zusätzliche Formate erzeugt werden – in dem Fall wird zusätzlich zum dem Image TIFF auch eine PDF und eine DOC Datei angelegt.

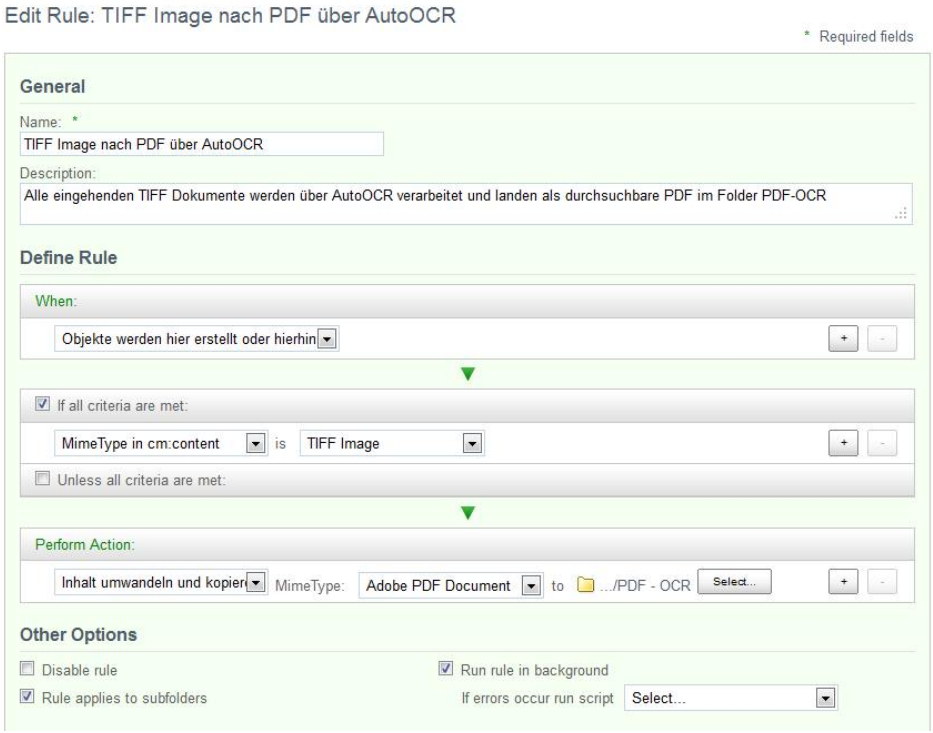




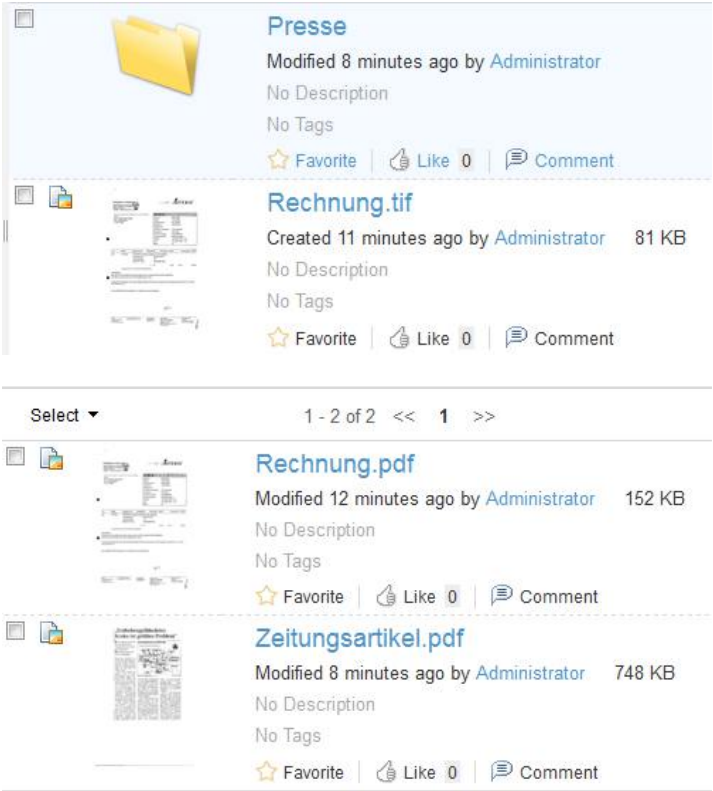


OCR Verarbeitung über Regeln auf Foldern

Alfresco kann Transformer auch als Regeln auf Ordner anwenden. Als Beispiel haben wir 2 Folder definiert – in den „Scan – Eingang“ werden die TIFF Scans abgelegt. Auf diesem Ordner wird eine Regel hinterlegt die besagt dass für jede TIFF Datei die in diesem Ordner abgelegt wird automatisch eine Konvertierung der TIFF Datei per AutoOCR Verarbeitung nach PDF –stattfindet und das PDF im Ordner „PDF – OCR“ landen soll.



TIFF Dokumente aus dem Eingangsfolder werden über die Regel automatisch in PDF Dokumente im Ausgangsfolder konvertiert.



AutoOCR JavaScript Binding für Alfresco

Das JavaScript API ermöglicht den direkten Zugriff auf den AutoOCR Service von Alfresco Scripts aus. In Repository JavaScripts (WebScript-Controller Scripts, Scripted Actions) können alle Funktionen des AutoOCR APIs aufgerufen werden. Dieses API ist vollständig unabhängig von der Einbindung der AutoOCR-Services als Alfresco-Transformer.

Skript Objekt und Konfiguration

Der Name des AutoOCR Skript Clients ist autoocr. Das Objekt steht nach der Installation allen Skripten automatisch zur Verfügung. Die Konfiguration erfolgt über ein JavaScript Object-Literal '{ ... }' wie es von vielen JavaScript Frameworks bekannt ist, und ist immer der erster Parameter der einzelnen Methoden.

Das Nutzungs-Pattern ist also beispielhaft:

```
1  var config = {
2      endpoint : "https://{AUTOOCRHOST UND PORT}/AutoOCRService",
3      username : "user",
4      password : "password",
5      apikey   : 12455
6  };
7
8  autoocr.{APIMETHODE1}(config, param1, param2, param2, ...);
9  autoocr.{APIMETHODE2}(config, param1, param2, param2, ...);
10
```

Abfrage von Eigenschaften

```
1  var config = {
2      endpoint : "https://{AUTOOCRHOST UND PORT}/AutoOCRService",
3      username : "user",
4      password : "password",
5      apikey   : 12455
6  };
7
8  var nrdocs = autoocr.getNrOfDocumentsInQueue(config);
9
10 var nrpages = autoocr.getNrOfPagesInQueue(config);
11
12 var availpages = autoocr.getAvailablePages(config);
13
14 var avgpages = autoocr.getAvgSecPerPage(config);
```

Upload und Download von Files (Transformationen)

```
1  var config = {
2      endpoint : "https://{AUTOOCRHOST UND PORT}/AutoOCRService",
3      username : "user",
4      password : "password",
5      apikey   : 12455
6  };
7
8  var nrdocs = autoocr.getNrOfDocumentsInQueue(config);
9
10 var nrpages = autoocr.getNrOfPagesInQueue(config);
11
12 var availpages = autoocr.getAvailablePages(config);
13
14 var avgpages = autoocr.getAvgSecPerPage(config);
```

Upload über Mimetypes

```
var sourceNode = companyhome.childByNamePath("tester.pdf");
var jobDescMimetype = {
    sourcenode : sourceNode,
    targetmimetype : 'application/pdf',
    settingsname : 'AbbyyFR10',
        label: "upload per mimetype"
};
...
```

Settings auslesen

```
var settings = autoocr.getSettingsCollection(config);
for(var i = 0; i < settings.length; i++) {
  var setting = settings[i];
  logger.log(setting.settingsname);
  logger.log(setting.description);
  logger.log(setting.enginename);
  for(var k = 0; k < setting.additionaloutputs.length; k++) {
    logger.log(setting.additionaloutputs[k]);
  }
  for(var k = 0; k < setting.inputformats.length; k++) {
    logger.log(setting.inputformats[k]);
  }
  for(var k = 0; k < setting.inputmimetypes.length; k++) {
    logger.log(setting.inputmimetypes[k]);
  }
  for(var k = 0; k < setting.outputformats.length; k++) {
    logger.log(setting.outputformats[k]);
  }
  for(var k = 0; k < setting.outputmimetypes.length; k++) {
    logger.log(setting.outputmimetypes[k]);
  }
}
```

Job- und Status-Abfrage

```
var job = autoocr.getJob(config, "b525042c-0789-4e67-ad65-991dea92fa35");

logger.log(job.guid);
logger.log(job.jobid);
logger.log(job.pagecount);
logger.log(job.status);
logger.log(job.error);
logger.log(job.owner);
logger.log(job.label);

var status = autoocr.getStatus(config, job.guid);

if(status == 1)
  logger.log("CREATED");

```

Mehrfache Transformationen

```
var jobDesc, job, resultCount, i, config, ext, targetNode, someContentNode, dlDesc;

jobDesc = {
  sourceNode: someContentNode,
  autoocrext: 'PDF', // alternativ kann hier auch targetmimetype verwenden.
  additionaloutputs: ['DOCX','PPT'], // alternativ können hier auch mimetypes verwendet werden
  settingsname: 'AbbyyFR10'
};

job = autoocr.upload(config, jobDesc);

autoocr.waitForCompletion(config, job.guid);

resultCount = autoocr.getResultCount(config, job.guid);

for(i = 0; i < resultCount; i++) {
  ext = autoocr.getResultExt(config, job.guid, i);
  targetNode = companyhome.createFile("autoocrtransformed-" + i + "." + ext);
  dlDesc = {
    guid : job.guid,
    index: i,
    targetnode : targetNode
  };
  autoocr.download(config, dlDesc);
}

```

AutoOCR Benutzer abfragen

```
var users = autoocr.getUsers(config);
for(var i = 0; i < users.length; i++) {
  logger.log("user #" + i + ": name=" + users[i].username + ", permissions=" + users[i].permissions);
}
```

AutoOCR Server starten und stoppen

```
autoocr.stopServer(config);
autoocr.startServer(config);
```

AutoOCR Server Verfügbarkeit prüfen

```
autoocr.isServerStarted(config);
```

AutoOCR Server Versionsabfrage

```
var version = autoocr.getServerVersion(config);
```

Abfrage der gesamten AutoOCR Jobliste

```
var allJobs = autoocr.getAllJobs(config);
for(var i = 0; i < allJobs.length; i++) {
    logger.log("job #" + i + ": label=" + allJobs[i].label);
}
```

Abfrage der AutoOCR Jobliste mit Statusfilter

```
var statusJobs = autoocr.getAllJobs(config, autoocr.STATUS_CONVERTED);
for(var i = 0; i < statusJobs.length; i++) {
    logger.log("job #" + i + ": label=" + statusJobs[i].label);
}
```

Abfrage der eigenen AutoOCR Jobs mit Statusfilter

```
var ownJobs = autoocr.getOwnJobs(config, autoocr.STATUS_CONVERTED);
for(var i = 0; i < ownJobs.length; i++) {
    logger.log("job #" + i + ": label=" + ownJobs[i].label);
}
```

Übersicht der gesamten Konfigurationsoptionen

```
var config = {
    endpoint : "https://212.41.224.95:8001/AutoOCRService", // required, this is the http or https endpoint of the Au
    username : "OCR", // optional, if set enables Basic Auth
    password : "****", // only used if username is set
        apikey: 12345, // required to talk to the AutoOCR Remote API
    credentialsencoding : "UTF-8", // optional defaults to UTF-8
    urlencoding : "UTF-8", // optional, defaults to UTF-8
    connectiontimeout: 10000, // optional, defaults to infinite
    jobtimeout : 60000, // optional, defaults to 120000msec (2min)
    sleeptime : 1000 // optional, defaults to 1000msec (1sec)
};
```

Übersicht der gesamten Uploadoptionen

```
1 var jobDesc = {
2     sourcenode : sourceNode, // required, the node which content is uploaded for transformation to AutoOCR
3     sourcecontentproperty : 'cm:content', // optional, if not the standard content, stored in the cm:content property,
4     autoocrext : 'PDF', // required if 'targetmimetype' is not given, the AutoOCR Extension names the target transforr
5         targetmimetype: 'application/pdf', // required if 'autoocrext' is not given, the mimetype of the requested tra
6     settingsname : 'AbbyyFR10', // required, the name of the AutoOCR setting which has to do the transformation
7     additionaloutputs : [] // optional, an empty list is the default. The AutoOCR service can produce additional outpu
8 };
```

Übersicht der gesamten Downloadoptionen

```
1 var dlDesc = {
2     guid : job.guid, // required, the guid of the uploaded job
3     index: 0, // optional, zero is the default value. If there where more than one transformations requested, the inde
4     targetnode : targetNode, // required, at which node the downloaded content should be stored.
5     targetcontentproperty : 'cm:content' // optional, if the content should not be stored in the default property cm:c
6 };
```

Fehlerbehandlung von sonstigen Netzwerk- und Laufzeitfehlern

```
1 try {
2   autoocr.upload(...);
3 } catch(ex) {
4   logger.log("network problems: " + ex);
5 }
```

Debugging Unterstützung

```
2011-12-27 14:26:58,781  DEBUG [impl.conn.DefaultClientConnection] [main] Sending request: GET /AutoOCRService/GetResult
2011-12-27 14:26:58,782  DEBUG [apache.http.headers] [main] >> GET /AutoOCRService/GetResultEx?jobID=b525042c-0789-4e67-
2011-12-27 14:26:58,783  DEBUG [apache.http.headers] [main] >> Authorization: Basic
2011-12-27 14:26:58,783  DEBUG [apache.http.headers] [main] >> Connection: close
2011-12-27 14:26:58,784  DEBUG [apache.http.headers] [main] >> Host: 212.41.224.95:8001
2011-12-27 14:26:58,784  DEBUG [apache.http.headers] [main] >> User-Agent: Apache-HttpClient/4.1.1 (java 1.5)
2011-12-27 14:26:59,209  DEBUG [impl.conn.DefaultClientConnection] [main] Receiving response: HTTP/1.1 200 OK
2011-12-27 14:26:59,209  DEBUG [apache.http.headers] [main] << HTTP/1.1 200 OK
2011-12-27 14:26:59,209  DEBUG [apache.http.headers] [main] << Content-Length: 95481
2011-12-27 14:26:59,209  DEBUG [apache.http.headers] [main] << Content-Type: application/octet-stream
2011-12-27 14:26:59,210  DEBUG [apache.http.headers] [main] << Server: Microsoft-HTTPAPI/1.0
2011-12-27 14:26:59,210  DEBUG [apache.http.headers] [main] << Date: Tue, 27 Dec 2011 13:26:55 GMT
2011-12-27 14:26:59,210  DEBUG [apache.http.headers] [main] << Connection: close
```

Logging des Transformers aktivieren

Die Log-Kategeorie ifresco.autoocr muss auf DEBUG gesetzt werden, um logging Ausgaben zum Verhalten des Transformer und der Interaktion mit dem AutoOCR Service zu erhalten. In der log4j.properties:

```
log4j.logger.ifresco.autoocr=DEBUG
```

AutoOCR Server API Version 1.9.9 - Job WebService

```
[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "PUT", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/UploadJob?ext={ext}")]
JobResult UploadJob(string ext, System.IO.Stream stm);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/Auth?userName={userName}&password={password}")]
string Auth(string userName, string password);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "PUT", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/UploadJobEx?ext={ext}&settingsName={settingsName}&maxPage={maxPage}")]
JobResult UploadJobEx(string ext, string settingsName, int maxPage, System.IO.Stream stm);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "PUT", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/UploadJobEx2?ext={ext}&settingsName={settingsName}&maxPage={maxPage}&outExts={outExts}")]
JobResult UploadJobEx2(string ext, string settingsName, int maxPage, string outExts,
System.IO.Stream stm);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "PUT", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/UploadJobEx3?ext={ext}&settingsName={settingsName}&maxPage={maxPage}&outExts={outExts}&label
={label}")]
JobResult UploadJobEx3(string ext, string settingsName, int maxPage, string outExts, string
label, System.IO.Stream stm);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/GetNrOfPages?jobID={jobID}")]
int GetNrOfPages(string jobID);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/GetStatus?jobID={jobID}")]
int GetStatus(string jobID);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/GetJob?jobID={jobID}")]
JobResult GetJob(string jobID);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/GetOwnJobs?status={status}")]
JobResult[] GetOwnJobs(int status);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/GetAllJobs?status={status}")]
JobResult[] GetAllJobs(int status);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/GetResultCount?jobID={jobID}")]
int GetResultCount(string jobID);
```

```
[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/GetResultExt?jobID={jobID}&index={index}")]
string GetResultExt(string jobID, int index);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/GetResultMimeType?jobID={jobID}&index={index}")]
string GetResultMimeType(string jobID, int index);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/GetResult?jobID={jobID}&index={index}&removeFile={removeFile}")]
byte[] GetResult(string jobID, int index, bool removeFile);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/RemoveJob?jobID={jobID}")]
void RemoveJob(string jobID);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, Method = "GET", BodyStyle =
WebMessageBodyStyle.Wrapped, UriTemplate =
"/GetResultEx?jobID={jobID}&index={index}&removeFile={removeFile}")]
System.IO.Stream GetResultEx(string jobID, int index, bool removeFile);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/GetSettingsCollection")]
EngineSettings[] GetSettingsCollection();

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate = "/GetAvailablePages")]
int GetAvailablePages();

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/GetNrOfPagesInQueue")]
int GetNrOfPagesInQueue();

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/GetNrOfDocumentsInQueue")]
int GetNrOfDocumentsInQueue();

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate = "/GetAvgSecPerPage")]
int GetAvgSecPerPage();

[OperationContract(IsOneWay = false)]
string CancelJob(string jobID);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate = "/GetServerVersion")]
string GetServerVersion();

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate = "/GetServerHash")]
int GetServerHash();
```

AutoOCR Server API Version 1.9.9 - Admin WebService

```
[ServiceContract]
public interface IAutoOCRAdmin {
[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate = "/IsStarted")]
bool IsStarted();

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate = "/Start")]
void Start();

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate = "/Stop")]
void Stop();

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/DownloadOCRProfile?profileName={profileName}")]
string DownloadOCRProfile(string profileName);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "PUT", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/UploadOCRProfile?profileName={profileName}&content={content}")]
bool UploadOCRProfile(string profileName, string content);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/DeleteOCRProfile?profileName={profileName}")]
bool DeleteOCRProfile(string profileName);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate = "/GetOCRProfiles")]
EngineSettings[] GetOCRProfiles();

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/AddUser?userName={userName}&password={password}&right={rights}")]
bool AddUser(string userName, string password, int rights);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/ChangeUserPassword?userName={userName}&newPassword={newPassword}")]
bool ChangeUserPassword(string userName, string newPassword);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/DeleteUser?userName={userName}")]
bool DeleteUser(string userName);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate =
"/ModifyUserRights?userName={userName}&rights={rights}")]
bool ModifyUserRights(string userName, int rights);

[OperationContract(IsOneWay = false)]
[WebInvoke(RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json,
Method = "GET", BodyStyle = WebMessageBodyStyle.Wrapped, UriTemplate = "/GetUsers")]
WebServiceUser[] GetUsers();
```