bitfarm® ARCHIV
Dokumentenmanagement

# Schnittstellen/Datenaustausch/ Integrationsmöglichkeiten

bitfarm-Archiv – Enterprise V3.6.2

# Inhaltsverzeichnis

# I. Dokumentenimport

## 1. Drop in überwachten Ordner

In der Konfigurationsdatei `scripts.ini`, im bitfarm-Archiv-Programmverzeichnis auf dem Server, kann unter

`ScannerImportPath=`
und/oder
`ExtendedImport=`
der Pfad zu einem Ordner angegeben werden, der vom bitfarm-Archiv *Spooldienst* überwacht wird. Dateien die hier hineingelegt werden, werden in die Archiv-Warteschlange (queue) gestellt und dann vom *Archivierungsdienst* indiziert, nach Regelvorgaben sortiert, nach Vorgabe ggf. verschlagwortet und dann archiviert. Weitere Schritte wie z.B. *Workflows* können anschließend erfolgen. Falls keine Sortierregeln angegeben sind und auch kein Plugin hier Einfluss nimmt, werden die Dokumente in „unverteilt“ archiviert.

Es ist möglich, in den überwachten Ordnern Unterordner anzulegen, die genauso heißen müssen wie die Templates zu den Archiven, in die eine direkte Archivierung stattfinden soll. Werden Dokumente in diese Unterordner eingebracht, wird von der Software nach einem gleichnamigen Template in *%bitfarm-archiv%\templates*-Verzeichnis gesucht und wenn vorhanden, landet das Dokument in dem Archiv, welches im Template angegeben ist.

Auch über den Dateinamen kann die Ermittlung des Zielarchives bzw. des Benutzers erfolgen.

Übliche Dateitypen, die so verarbeitet werden können, sind `TIF`, `.PDF`, `.JPG`, `.GIF`, `.BMP`, `.DOC`, `.XLS`, `.PPT`, `.MSG`, `.EML`, `.DWG`, `.PLT`, `.DXF`, `.RTF`, `.HTM`, `.HTML`, `.ASC`, `.TXT`

## 2. Datenübergabe mit .job-Datei

Ein ähnliches Verfahren wie die Datenübergabe mit XML-Dateien besteht mit den .job-Dateien, die bitfarm-Archiv auch intern zum Informationstransport nutzt. Hier muss zu jeder Dokumentendatei die archiviert werden soll eine .job-Datei gleichen Namens existieren. Der Import erfolgt über das Einlegen der .job-Datei, und anschließendem Einlegen der Dokumentendatei in den Haupt-Übergabeordner, welcher in der Profildatei (Name der CON-Datei) als Clientspooler bzw. UNC-Clientspooler bezeichnet wird.

Die .job-Datei wird aus einer Vorlagedatei erzeugt, dem so genannten Template. Dieses Template wird vom DMS automatisch beim Erstellen des Archivs erzeugt. Es stellt ein leeres Gerüst dar, welches bereits angibt, welche Felder in einem Archiv gültig sind. Das Template muss dann von der importierenden Anwendung jeweils kopiert, ausgefüllt und in .job umbenannt werden. Ein Beispiel für eine solche Templatedatei welche im *%bitfarm-archiv%* Programmverzeichnis auf dem Server im Unterornder templates (Archivname.tpl) liegt:

```
[Version]
Version=36
[Archiv]
Profil=muster-gmbh
Name=Kreditoren
Tabelle=28052015172108
Arcid=10
[Document]
DOCID=
gdoc_id=
Titel=
Orginal=
Kopie=
Quelle=
Benutzer=
userid=
Volltext=
Status=
StatStr=
OCR_QF=
OCR_Typ=OMP
Schlagworte=
Datum=
Filter=
Pages=
[Hash]
SHA1=
[Referenz]
Anzahl=0
[SVN]
SVN_Link=
SVN_Version=
SVN_Revision=
SVN_Datum=
SVN_Info=
[Tasks]
Termin=
Alarm=
Bearbeiter=
Notiz=
TimerOptionen=0
[Zusatzfelder]
Felder=17
ZusTitel1=Auftragsnummer
ZusFeld1=add_5
ZusWert1=
ZusTitel2=Rechnungsnummer
ZusFeld2=add_3
```

```
ZusWert2=
ZusTitel3=Rechnungsdatum
ZusFeld3=add_2
ZusWert3=
ZusTitel4=Rechnungsbetrag
ZusFeld4=add_14
ZusWert4=
ZusTitel5=Rechnungsprüfer
…
```

Durch das übergebende System können folgende Felder gefüllt werden:

- Titel

- Benutzer

- Zusatzfelder

Das übergebende System kann zusätzlich eine .ftx-Datei gleichen Namens mitliefern, die für die Volltextsuche verwendet wird. Dann ist bei OCR-Typ TXT einzutragen, damit keine OCR mehr erfolgt.

Es kann auch eine .slw-Datei mitgegeben werden, deren Inhalt dann in die globalen Schlagworte gespeichert wird.

Achtung: Es gibt Daten, welche nicht vom übergebenden System mitgeliefert werden können. Diese sind:

- Datum

- Referenzen

- Termin

- Alarm

- Bearbeiter

- Notiz

## 3. Drucken ins Archiv

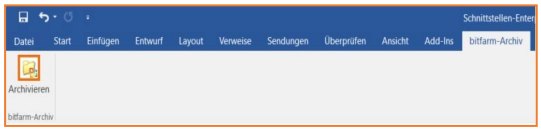
Es gibt die Möglichkeit, virtuelle Softwaredrucker für die Archivierungsfunktion einzurichten. Diese können auf dem Client eingerichtet werden. (siehe entsprechenden Abschnitt im Systemhandbuch). Bei der Einrichtung der Druckers kann das Zielarchiv und ein realer Drucker (dieser muss auf dem Server installiert sein) für eine zusätzlich Papierausgabe definiert werden.

Druckdaten können über Regeln zur Verschlagwortung in der .wfd-Datei mit sehr hoher Genauigkeit ausgelesen und indiziert werden. Der Nachteil dieses Verfahrens gegenüber einem XML-Datenimport oder dem Import über die .job-Datei, liegt darin, dass nur Informationen in das DMS übernommen werden können, die auf dem Dokument selbst dargestellt sind bzw. aus der Darstellung abgeleitet werden können (siehe Plugins).

## 4. Import von ASCII-Daten (COLD)

Für Systeme (meist UNIX-Host-basierte ERP-Anwendungen), die weder PDF/TIF/DOC exportieren können, noch auf Windows-Drucker drucken können, gibt es die Möglichkeit des Druckdatenimportes via COLD. Hierzu müssen die Daten, die an den Drucker geschickt werden, in eine Datei geschrieben und in einen der o.g. überwachten Ordner kopiert werden. Die Dateiendung kann weggelassen werden oder es wird ASC angegeben. Derzeit ist eine Epson-Druckeremulation fest einprogrammiert. Wenden Sie sich für andere Steuerkommandos bitte an den bitfarm-Archiv Softwaresupport.

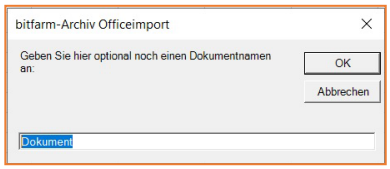
## 5. MS-Office Add-Ins




Für das Microsoft-Office Paket, genauer für Word, Excel und Outlook, stehen sogenannte Add-Ins, bis einschließlich Office 365, für das schnelle Archivieren der Originaldatei aus der jeweiligen Anwendung in das DMS, zur Verfügung. Die Add-Ins werden mit der Client-Installation automatisch installiert, können aber auch manuell nachinstalliert werden. Zur manuellen Installation, rufen Sie bitte die `bfaOfficeSetup.msi` auf Ihrem DMS-Server auf.

*%bitfarm-archiv%\install\Bitfarm-Tools\Office-Add-In*

Wählen Sie in Word bzw. Excel das Add-In zur Archivierung der Originaldatei.



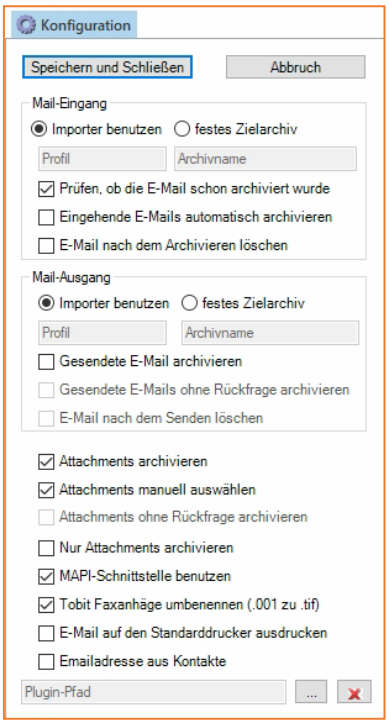

Sollte das Dokument noch unbenannt sein, so können Sie dem Dokument einen Namen geben, bevor es in das DMS importiert wird. Natürlich kann der Name auch im Nachhinein geändert bzw. festgelegt werden.

Hinweis: Im bitfarm-Archiv DMS können Dokumente auch über Ihren Namen gefunden werden.


Bei dem Addin für Outlook haben Sie neben der Archivierung der Originaldatei (msg-Datei) auch die Möglichkeit, lediglich die Anhänge zu archivieren. Klicken Sie auf *manuelle Archivierung* um hier ggf. Einstellungen zu treffen, die Ihrer Arbeitsweise entspricht.

Das automatische Archivieren von E-Mails macht bei Benutzer-Postfächern üblicherweise keinen Sinn, weil man schlecht wissen kann, um welche Art von Dokument es sich handelt.

Tipp: Postfächer, wie Rechnungen@IhrUnternehmen.de, können Sie bequem und komfortabel über den bfaPop3-Connector oder den imap4-Service, wie im nächsten Kapitel beschrieben, archivieren.




Sie haben die Möglichkeit einige Einstellungen in dem Add-In für MS-Outlook zu treffen, die sich auf Ihren persönlichen Mailein- und Mailausgang beziehen. Wie zuvor schon erwähnt macht es in den meisten Fällen keinen Sinn ein Benutzer-postfach automatisiert ohne Rückfrage zu zu archivieren. Spam möchte man üblicherweise nicht archivieren und das System kann schlecht feststellen, was von den Mails ein reiner Schriftverkehr, ein Angebot, eine Anfrage… etc. ist, weswegen man den Importer wählen sollte (default) und somit selbst festlegen muss, in welches Archiv die Mail und/oder der Anhang archiviert werden soll. Wenn Sie die Option *Attachments manuell auswählen* aktivieren, so werden die Anhänge einer Mail in einem extra Fenster angezeigt und nur die Anhänge archiviert, die sie dafür bestimmen.

## 6. Automatischer Mail-Import (POP3)

Mit dem bitfarm-E-Mail-Dienst und dem bfaPop3-Connector können Sie wichtige Mailkonten automatisiert abrufen und zielgenau archivieren lassen.

Hinweis: bitfarm rät davon ab, alle Unternehmens-Mails oder Sammelpostfächer wie info@ihrunternehmen.de, auf diesem Weg archivieren zu wollen, sondern gezielt Postfächer wie rechnungen@ihrunternehmen.de, oder bewerbungen@ihrunternehmen.de, wo schon anhand der Postfachs klar ist, um welche Art von Dokumenten es geht und in welchen Archiven diese dann automatisiert abgelegt werden können.

### POP3-Service als Dienst

Achtung: Läuft der POP3 Service im Dienste-Kontext, wird das Postfach, sobald die Mails zu bitfarm übertragen wurden, geleert.

Erstellen Sie im ersten Schritt ein Verzeichnis *bfaEMail* im *bitfarm-Archiv*-Verzeichnis und fügen Sie alle Dateien aus dem Verzeichnis *...\bitfarm-Archiv\install\Bitfarm-Tools\bfaEMail* hinzu.

Ändern Sie in der beigefügten `bfapop3.ini` alle notwendigen Einträge Ihrer Umgebung entsprechend:

```
[Main]
POP3 Server=Mailservername oder IP
Username=Testmail@Mailserver.de
Password=test
Uebergabe=c:\bitfarm-archiv\import\Eingangsrechnungen
UseTLS=true
TLSUser=TLSBenutzer
```

Mit dem Befehl `POP3 Server` geben Sie den Servernamen oder die IP-Adresse des POP3 Servers an. Unter `Username` geben Sie den Accountnamen an. Der Befehl `Uebergabe` lässt Sie das Archiv wählen, in welches die Dokumente importiert werden soll. Achten Sie dabei darauf, dass der *Import* durch den Spooldienst überwacht wird, da die Emails sonst nicht in das DMS importiert werden. Mit `UseTLS` entscheiden Sie, ob die Anmeldung per TLS erfolgen muss. Die möglichen Werte sind hierbei `true` oder `false`. Der Befehl `TLSUser` gibt den (Windows-) Benutzer des POP3 Kontos an. Zudem kann, bei einem geänderten Port (Standard ist 110), noch der Befehl `Port=` hinzugefügt werden, hinter dem der genutzte Port definiert wird.

Führen Sie in einer Administratorkonsole den Befehl `bfaEMailService.exe -install` aus und der POP3 Service wird installiert. Im Dienste-Kontext muss diesem noch die Anmeldung per bitfarm-User zugewiesen werden.

Sobald Sie den Dienst starten führt dieser alle 5 Minuten die Kontrolle des konfigurierten Postfaches aus und sendet die gefundenen Mails direkt an den unter `uebergabe` angegebenen Pfad. Sie können als Startparameter noch `-t:<sec>` mitgeben und das Abholintervall ändern.

Hinweis: Im Dienste-Kontext schreibt der POP3 Service Logs, welche in das *…\bitfarm-Archiv\bfaEMail\logs*-Verzeichnis gelegt werden.

Achtung: Sollten Sie mehrere POP3-Konten abrufen wollen, so ist das nicht über diesen Dienstkontext möglich, da in der INI-Datei nur ein Postfach angegeben werden kann.

Tipp: Erstellen Sie eine Batch-Datei, welche in jeder Zeile den Abruf des Kontos hat, welches Sie abrufen wollen. Nachfolgend sind alle Parameter erklärt und aufgelistet und auch ein Beispielaufruf angegeben. Wenn Sie die .bat-Datei erstellt haben, binden Sie die bfapop3.exe über die Windows-Taskplanung an und übergeben Sie die Batch-Datei als Parameter.

## POP3-Service als Konsolen-Aufruf

Der POP3 Service kann auch als Konsolenaufruf gestartet werden. Die möglichen Parameter können mit dem Aufruf `bfapop3.exe -h` erfragt werden:

```
bfapop3.exe <POP3-Server|-auto><User><Pass><Übergabe-Pfad>
{-TLS}{-TLSUser:Name}{-p:Port}{-nodelete}{-log}
```

Nachfolgend die Bedeutung der einzelnen Parameter:

| | |
|---|---|
| `POP3-Server` | Name oder IP des POP3 Servers |
| `-auto` | Parameter zur automatischen Abholung. Nutzt die in der bfapop3.ini hinterlegte Verbindung. |
| `User` | Benutzer des POP3 Accounts |
| `Pass` | Passwort des POP3 Accounts |
| `Übergabe-Pfad` | Verzeichnis, in das die Emails gespeichert werden. |

Optional:

| | |
|---|---|
| `-TLS` | Aktivierung der Anmeldung per TLS |
| `-TLSUser` | (Windows-)Username des POP3-Kontos |
| `-p` | Port (default ist 110) |
| `-nodelete` | Emails werden im Postfach nicht gelöscht |
| `-log` | Schreibt ein Log in einem eigens angelegten Unterverzeichnis |

Beispielaufruf:

```
bfapop3.exe 127.0.0.1 bitfarm passwort c:\bitfarm-archiv\import\ -TLS -p 1337 -log
```

## 7. E-Mail-Archivierung mit dem bitfarm-imap4-Service

Erstellen Sie im ersten Schritt ein Verzeichnis *bfaimap4* im bitfarm-Archiv-Verzeichnis und fügen Sie alle Dateien aus dem Verzeichnis *...\bitfarm-Archiv\install\Bitfarm-Tools\bfa_imap4* hinzu.

Zur Installation des Tools als Dienst, führen Sie eine Administratorkonsole im neu erstellten Verzeichnis aus. In dieser Konsole führen Sie den Befehl `bfa_imap4_service.exe install` aus. In der Konsole wird direkt der bitfarm-Dienstebenutzer ermittelt. Tragen Sie noch das Passwort dieses Nutzers ein und der Dienst wird unter diesem Benutzer installiert.

Hinweis: Sollte noch kein bitfarm-Dienst installiert worden sein, muss der Benutzer manuell eingetragen werden, da der Benutzer anhand der schon installierten bitfarm-Dienste ermittelt wird.

Nach der Installation des Dienstes kann die Konfiguration erfolgen. Nutzen Sie dazu die Datei `bfa_imap4_example.ini` und nennen Sie diese in `bfa_imap4.ini` um, denn nur unter diesem Namen wird Sie im Dienstekontext aufgerufen.

Die Konfigurationsdatei ist in zwei Sektionen eingeteilt. Der Bereich `[main]` definiert dabei die technischen Details:

```
[main]
logfile = c:\bitfarm-archiv\bfa_imap4\bfa_imap4.log
loglevel = debug
logbackup = 7
service_interval = 900
```

Unter `logfile` wird der Pfad und Name des Logs definiert, in welches die Informationen zum Programmablauf geschrieben werden, mit `loglevel` (`debug`, `info`, `error`) wird die Loggingstufe definiert und `logbackup` definiert die Aufbewahrungsfrist für die einzelnen Loggingdateien. Der Schalter `service_interval` definiert die Abfragezeit in Sekunden für den installierten imap4-Dienst.

Beachten Sie: Fehler im Zusammenhang mit dem Dienst, werden im Eventlog protokolliert, Fehler bei der Ausführung in der `Log`-Datei. Für die Fehleranalyse empfehlen wir die `Log`-Datei, welche detaillierte Meldungen beinhaltet.

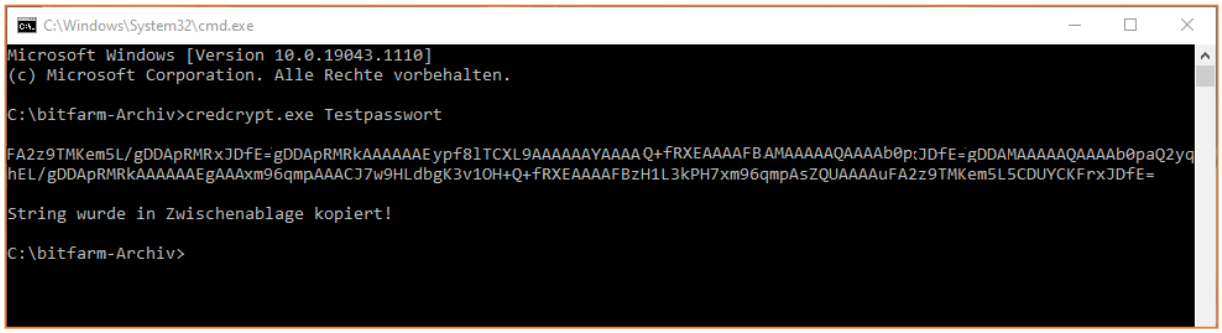
Der zweite Abschnitt definiert die technischen Details des Postfachs, den Ablageort und den Umgang mit den schon archivierten Mails bzw. Anhängen. Es können mehrere Sektionen erstellt werden, in denen jeweils ein Postfach überwacht wird. Nachfolgend eine beispielhafte Konfiguration und Erläuterung:

```
[rechnungen@meinefirma.de]
host = Mailservername.hoster.de
port = 993
encryption_method = ssltls
fetch_method = all
delete_after_fetch = False
username = Beispielnutzer
use_dpapi = True
password = 47110815abc6rdcbj655678ijgfdw3…
maildirs = INBOX
savefolder = C:\bitfarm-archiv\import
maildir_as_subfolder = False
python_plugin = extract_pdf_att
```

In eckigen Klammern definieren Sie zuerst einen Sektionsnamen. Dieser muss dabei eindeutig sein. Es können auch mehrere Folgesektionen für verschiedene Postfächer genutzt werden. Unter `host` tragen Sie den imap-Server ein, auf dem das Postfach überwacht werden soll. Der Befehl `port` definiert den genutzten Port, Standard ist hier `993`, bzw. `143` für imap4-Konten, je nach Verschlüsselungsmethode. Diese kann auch im nachfolgenden Schalter `encryption_method` definiert werden. Dabei stehen `ssltls`, `starttls` und `none` zur Verfügung. Diese Informationen können vom entsprechenden Mailprovider zur Verfügung gestellt werden.

Hierfür kann auch eine Abfrage als definierter Benutzer notwendig sein. Diesen definieren Sie mit `username` und mit `password` das jeweilige Passwort. Dabei kann mit dem Schalter `use_dpapi` entschieden werden, ob das Passwort in der Konfigurationsdatei verschlüsselt abgelegt werden soll.

Hinweis: Zum Verschlüsseln des Passwortes muss das Tool `credcrypt.exe` aus dem *bitfarm-Archiv*-Verzeichnis genutzt werden. Rufen Sie dieses in einer Konsole mit dem zu verschlüsselten Begriff als nachfolgendem Parameter auf, und die Ausgabe ist der verschlüsselte Begriff, wie in der nachfolgenden Abbildung:




Der Schalter `fetch_method` definiert, ob alle Mails (`all`), oder nur ungelesene Mails (`unseen`) archiviert werden sollen. Mit dem Wert `unseen` werden dabei nur ungelesene Mails gespeichert, schon gespeicherte Mails werden nicht wieder abgegriffen.

Mit `delete_after_fetch` kann noch eingestellt werden, ob die Mails im Postfach nach dem archivieren in bitfarm, gelöscht werden sollen (`true`), oder weiterhin im Postfach verweilen sollen (`false`). Zudem kann mit dem Schalter `maildirs` noch entschieden werden, welche Unterverzeichnisse des Postfachs überwacht werden sollen. Dabei können kommagetrennt auch mehrere Verzeichnisse abgefragt werden. Die Schreibweise muss dabei der auf dem Mailserver entsprechen. Die genauen Schreibweisen der Unterverzeichnisse können dabei nach der Ausführung aus dem Log entnommen werden (bei `loglevel=debug`):

```
2021-01-01 08:00:00,000 - bfa_imap4 - DEBUG - list of available maildirs: ['INBOX', '"INBOX.Rechnung Mobil"']
```

Hinweis: Sollte das Unterverzeichnis Sonder- oder Leerzeichen enthalten, muss der Name Anführungszeichen stehen.

Beachten Sie: Sollten Sie als Abholmethode unter `fetch_method=all` definiert haben, und das Postfach nicht löschen lassen, wird er bei jedem Vorgang ALLE Mails neu archivieren.

Um zu definieren, wo die Mails abgespeichert werden, kann der Schalter `savefolder` genutzt werden. Hierbei muss ein vom bitfarm-Spooldienst überwachtes und vorhandenes Verzeichnis ausgewählt werden. Sollen die Mails direkt in den `maildirs` definierten Unterordnern gespeichert werden, muss der Schalter `maildir_as_subfolder=true` gesetzt werden. Dabei werden gegebenenfalls die Unterordner im definierten `savefolder` automatisch erstellt und die Mails direkt in den entsprechenden Ordner gelegt.

Beispiel:

```
savefolder=c:\bitfarm-archiv\import\
maildir=inbox, „inbox.rechnungen privat“
maildir_as_subfolder=true
```

Hierbei würden im *import*-Ordner zwei weitere Verzeichnisse generiert werden, sofern noch nicht vorhanden, mit dem Namen *inbox* und *inbox.rechnungen privat*. Die Mails würden dem jeweiligen Unterverzeichnis entsprechend abgelegt werden.

Hinweis: Die Unterverzeichnisse sollten als Template für den Import in das bitfarm-Archiv DMS definiert worden sein, da die Emails sonst im *unverteilt*-Verzeichnis landen könnten.

Mit dem Schalter `python_plugin` kann ein Pluginaufruf erfolgen, um die Mail weiterzubearbeiten, bevor Sie archiviert wird, beispielsweise eine Trennung von Mail und Anhang. Hierfür muss der Name des Plugins im dafür vorgesehenen gleichnamigen Unterverzeichnis angesprochen werden. In jeder Sektion kann dabei maximal ein Plugin aufgerufen werden.

Hinweis: Das Tool kann auch in einer Batchdatei oder in der Konsole aufgerufen werden. Führen Sie einfach die `bfa_imap4.exe` in der Konsole aus. Dabei werden in der Konsole Informationen zum Vorgang eingetragen, entsprechend Ihrem definierten `loglevel`.

### Beispielhafter Pluginaufruf durch den imap4-Dienst

Als `python-plugin` für den imap4-Dienst wird beispielhaft nachfolgend das Tool `extract_att` angesprochen. Dieses ermöglicht Ihnen Anhänge der Mail zu extrahieren und die Mail zu

verwerfen. Dieses befindet sich unter dem Pfad *bfa_imap4\python_plugins\extract_att*. Darin befindet sich eine Konfigurationsdatei, welche wir für die Einrichtung des Plugins nutzen können. Nennen Sie diese zuerst in `extract_att.ini` um, damit das Plugin diese später einliest.

Hinweis: Die Datei muss immer den Namen `extract_att.ini` tragen. Machen Sie zudem eine Sicherung der Beispielkonfiguration.

Hinweis: Sie können das Plugin auch mehrfach aufrufen, wenn Sie Ihm einen anderen Namen geben und so unterschiedliche Fälle abbilden. Dafür muss der Ordner entsprechend kopiert und umbenannt werden. Die Konfigurationsdatei muss immer denselben Namen im Unterverzeichnis tragen.

In der Konfigurationsdatei können sich unterschiedliche, eindeutig benannte Sektionen befinden. Diese werden durch die eckigen Klammern symbolisiert. Beispielhaft einmal nachfolgend eine Sektion:

```
[Lieferscheine_Müller_GmbH]
att_regex =  .*lieferschein.pdf
sender_regex = .*@mueller-gmbh.com>
subject_regex = .*Lieferschein
importfolder = Lieferscheine
continue_on_match=False
```

Diese Sektion befasst sich mit den Lieferscheinen der Firma Müller GmbH. Dabei wird mit dem Befehl `att_regex` nach dem Namen des Anhangs via regulärem Ausdruck gesucht. Hier muss der Anhang also den Titel `Lieferschein.pdf` tragen, bzw. genauer gesagt mit `lieferschein.pdf` enden. Da der Absender sich unterscheiden kann, wird in `sender_regex`, welches den Absender prüft, hier nur nach dem Firmennamen gesucht, der spezielle Absender kann dabei variabel sein. Sollte der Befehl leer bleiben, spielt der Absender keine Rolle und jede reinkommende Mail wird betrachtet. Der Befehl `subject_regex` definiert den Betreff der Mail und wurde hier auf `Lieferschein` gesetzt. Sofern das Wort Lieferschein also im Betreff steht, greift auch diese Regel. Da es sich dann eindeutig um einen Lieferschein handeln muss, wurde hier der Importordner mithilfe des Befehls `importfolder` noch genauer definiert, unabhängig vom ausgeführten imap4-Dienst-Ordner. Sollte kein spezieller Ordner gewählt werden müssen, kann dieser auch mit `NONE` gefüllt werden und der Standard-Importordner des imap4-Dienstes wird genutzt.

Zuletzt kann mit dem Befehl `continue_on_match` noch entschieden werden, ob nach der Sektion das Plugin weiter ausgeführt werden soll, oder nur nach der Sektion enden soll, wenn ein Treffer gefunden wurde.

So können auch mehrere Sektionen für unterschiedliche Dokumente definiert werden.

**Wichtig:** Es müssen alle regulären Ausdrücke greifen, ansonsten wird der Anhang nicht über das Plugin archiviert. Die drei Befehle `att_regex`, `sender_regex` und `subject_regex` können zudem nur gegen den Email-Header matchen. Beachten Sie die entsprechende Schreibweise! Die Ausdrücke werden dabei Case-Sensitive betrachtet.

Nachfolgend ein weiteres Beispiel:

```
[JPG Rechnung]
att_regex = Rechnung\d{6}.jpg
sender_regex = .*@bitfarm-archiv.de>
subject_regex =
importfolder = None
continue_on_match = True

[PDF]
att_regex = .*\.pdf
sender_regex = .*<rechnungen@bitfarm-archiv.de>
subject_regex =
importfolder = None
continue_on_match = True
```

Sofern eine Rechnung mit einem Titel „`Rechnung123456.jpg`“, wobei die Zahlen für zufällige Zahlen stehen, von der Adresse `…@bitfarm-archiv.de` gesendet wird, wird der Anhang archiviert. Durch `continue_on_match` wird die nachfolgende Regel auch noch angewandt und jedes Dokument, mit einer PDF-Endung von der Adresse `rechnungen@bitfarm-archiv.de` entsprechend genauso archiviert. Für beide Dateitypen wurde kein spezielles Importverzeichnis gewählt. Auch nach der PDF-Regel werden weitere Sektionen betrachtet, da hier `continue_on_match` auf `True` gesetzt ist.

Hinweis: Der Absender einer Mail ist nicht nur der Name der Email-Adresse. Meist sieht der Absender eher folgendermaßen aus `=?iso-8553-1?Q?Peter_M=FCller?= p.mueller@bitfarm-archiv.de`. Achten Sie bei der Erstellung des regulären Ausdruckes in `sender_regex` darauf.

Hinweis: Sie können gegebenenfalls auch leere Befehle nutzen, wenn beispielsweise der Betreff oder Absender irrelevant sind und nur der Name bzw. Dateityp des Anhangs entscheidend ist.

# 8. Archivierung von SAP-Belegen über die bitfarm Content-Server Schnittstelle (bfaSAP)

## a) Beschreibung

Die bitfarm-Archiv SAP-Schnittstelle ist ein sogenannter Content-Server (CS) gemäß der Schnittstellenbeschreibung 'SAP Content Server http 4.5'. Sie wurde vollständig nach der Schnittstellenbeschreibung implementiert. Die Schnittstellenbeschreibung ist über diesen Link erreichbar:

http://help.sap.com/saphelp_nw70ehp2/helpdata/de/9b/e8c186eaf811d195580000e82deb58/frameset.htm

Im Wesentlichen handelt es sich bei dieser Schnittstelle um einen Web-Service (Web-Server-Dienst), der über das HTTP-Protokoll Dokumente von SAP empfangen, oder diese auf Anforderung an SAP senden kann. Hierzu sendet SAP sogenannte HTTP-Requests an den Content-Server, um Dokumente zu übermitteln oder anzufordern.

Bei der Verwendung dieser Schnittstelle kommen nur allgemeine, internationale Standards zum Einsatz, wie das HTTP-Protokoll 1.1 (RFC 2068), HTML (HyperText-Markup-Language), URL (Uniform Resource Locators, RFC 2396), UTC (Universal Time Coordinated), sowie digitale Verschlüsselungsverfahren nach dem Public/Private-Key Prinzip mit PKCS#7 Signaturen (RFC 2315)

In SAP werden sogenannte Repositories angelegt, denen verschiedene Belegarten zugeordnet werden können. Den Repositories wird eine Netzwerkadresse und Port zugeordnet. Über diese Adresse ist für SAP damit ein Content-Server definiert, in unserem Fall der bitfarm Archiv Content-Server (bfaSAP). Im bitfarm Content-Server wird wiederum eine Zuordnung von Repositories zu Archiven im DMS definiert.

Die Kommunikation über die Schnittstelle kann optional digital signiert werden. Dazu signiert SAP alle Request an den Content-Server. Der Content-Server kann mittels eines öffentlich Schlüssels, der von SAP übermittelt wird, überprüfen, ob die Requests unverfälscht sind.

## b) Archivierung und Bereitstellung von Dokumenten

Wird in SAP ein Dokument erzeugt, sendet SAP einen HTTP-Request an die CS-Schnittstelle in Form einer URL. In dieser URL wird sowohl der Befehl, als auch die notwendigen Parameter und optional eine Signatur übergeben. Soll zum Beispiel ein Dokument erzeugt werden, so ist dies ein Create-Request mit Angaben des Repository, der SAP eigenen Dokumenten-ID, die Zugriffsberechtigung etc. Im sogenannten Body des http-Request befindet sich dann das eigentliche Dokument, in der Regel als PDF-Datei. Diese Datei wird dann vom Content-Server

in der Archivablage im revisionssicheren Bereich gespeichert und ein Datenbankeintrag (Dokument) in der bitfarm-Datenbank erzeugt. Hierbei wird auch die SAP eigene Dokumenten-ID in einer gesonderten Tabelle (sys_sap) hinterlegt. Abschließend übermittelt der Content-Server einen HTTP Return- oder Errorcode zurück an SAP.

Wird das Dokument nun seitens SAP angefordert, übermittelt SAP einen Get-Request mit der ID des geforderten Dokuments. Anhand der SAP-ID wird das entsprechende Dokument in der bitfarm Datenbank ermittelt und im Body der Server-Antwort an SAP übermittelt.

Darüber hinaus verarbeitet der CS die weiteren Befehle zum Aktualisieren, Ergänzen, Suchen und Löschen von Dokumenten, sowie noch administrative Befehle zum Abfragen von Informationen zu Dokument oder Content-Server, oder zum Übermitteln der Zertifikate für die jeweiligen Repositories.

### c) Revisionssicherheit

Der bitfarm Content-Server läuft auf dem bitfarm Archiv Server als ein Windows-Dienst. Dieser Dienst wird mit den Rechten des bitfarm-Archiv Dienstuser gestartet, ein generisches Benutzerkonto, welches nur für die bitfarm-Dienste eingerichtet wird. Dieser Dienstuser hat als Einziger schreibenden Zugriff auf den revisionssicheren Bereich der Archivablage.

Der revisionssichere Bereich ist eine, von bitfarm-Archiv verwaltete, Verzeichnisstruktur.

Wenn jetzt über einen Create-Request ein Dokument von SAP an den Content-Server übermittelt wird, speichert dieser das Dokument (den Body des http-Request), sofort und ohne Umwege, direkt in den revisionssicheren Bereich. Einmal dort gespeichert, kann dieses Dokument nicht mehr geändert werden, da außer dem Dienstuser niemand Schreib- oder Ändernrechte auf diesen Bereich besitzt.

Dieses Dokument / die Datei bleibt physikalisch an diesem Ort liegen, selbst wenn innerhalb des DMS das Dokument einem anderen Archiv zugeordnet wird, da dies nur über eine entsprechende Zuordnung innerhalb der bitfarm Datenbank geschieht.

Sollte seitens SAP eine Delete-Request erfolgen, so wird das Dokument in der Datenbank tatsächlich nur als 'Gelöscht' markiert. Auch hier bleibt die ursprüngliche Datei unverändert.

Über die Historytabelle im bitfarm Archiv DMS kann über die gesamte Laufzeit nachvollzogen werden, was mit dem Dokument nach der Archivierung noch weiterhin geschehen ist.

### d) Technische Voraussetzungen

Der bitfarm-Archiv Content-Server läuft auf Windows-Servern, getestet mit der Version Server2003, Server2008 R2 und Server2016. Theoretisch wird jede Plattform unterstützt, die einen Python-Interpreter ausführen kann.

Python ist die zugrunde liegende Programmiersprache des Content-Servers, die verwendete Version ist Python 2.7.2, bzw. 2.7.3.

Für die Webserverfunktionalität wird das Python-Framework CherryPy 3.2 eingesetzt, welches ebenfalls installiert werden muss.

Die Prüfung der Signaturen anhand der SAP-Zertifikate im signierten Modus (falls konfiguriert) wird mit OpenSSL in der Version 1.0.1L durchgeführt und muss dann ebenfalls vorhanden sein.

### e) Konfiguration

Der bitfarm-Archiv Content-Server benötigt zwei Konfigurationsdateien, die `bfaSAP.conf` für die Angabe der Netzwerkadresse und Port, die der Webserver benötigt und die `bfaSAP.ini`, in der allgemeine Einstellungen gemacht werden und die Repositories den Ziel-Archiven im DMS zugeordnet werden. Jedes SAP-Repository wird mit einer Kennung aus zwei Buchstaben angegeben, entsprechend gibt es für diese Kennungen Sektionen in der `ini`-Datei, wo das zugeordnete Archiv hinterlegt ist. Der Aufbau der beiden Dateien ist eigentlich selbsterklärend.

Hinweis: Die Angaben `arcid`, `tblname`, `arcname` können der Tabelle `sys_arc` in der bitfarm Datenbank entnommen werden, oder aus erstellten Archiv-Templates, zu finden in *%bitfarm-archiv%\templates*

Als Beispiel:

```
[global]
server.socket_host = "127.0.0.1"
server.socket_port = 8080
server.thread_pool = 10
engine.autoreload_on = False
```

*Beispiel: bfaSAP.conf*

```
[SAP]
; mehrere Repositorys mit ; getrennt angeben
repository=T1;T2
server=dms-server
signed=false
[bitfarm]
profil=demo
changeable=.doc;.xls
openssl=c:\OpenSSL\bin\openssl.exe
[T1]
arcid=68
tblname=09032010135935
arcname=Rechnungen_Gutschriften_Vertrieb
description=Rechnungen_Gutschriften_Vertrieb
[T2]
arcid=111
tblname=28042010161913
arcname=Rechnungen_Gutschriften_ET
description=Rechnungen_Gutschriften_ET
```

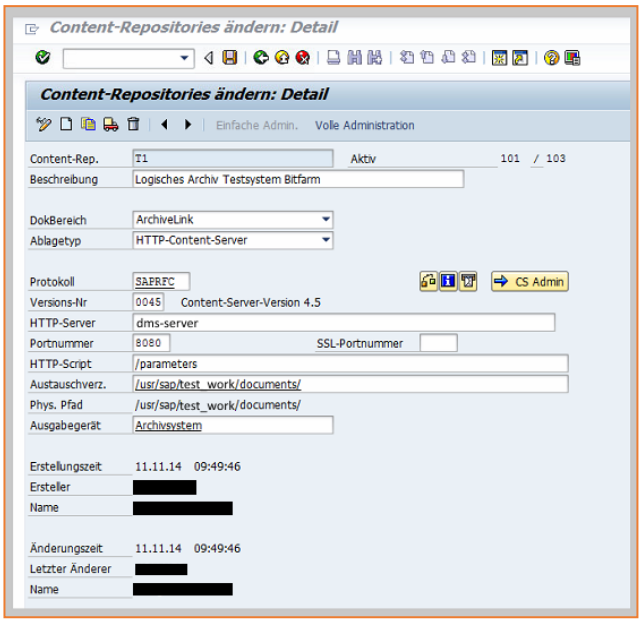
(…)

Beispiel: bfaSAP.ini

Auf der SAP-Seite muss dann der bitfarm Content-Server in der Konfiguration der SAP-Repositories hinterlegt werden. Erstellen Sie hierzu neue Repositories oder Ändern Sie die Repositories, deren Dokumentenarten zukünftig in bitfarm-Archiv archiviert werden sollen.

(SAP Transaktion OAC0)

Relevant sind dabei im Grunde nur die Repository Bezeichnung, der http-Server (Name des bitfarm-Servers), die Portnummer und das http-Script (/parameters).




## f) Weiterführende Informationen

### Schnittstellenbeschreibung SAP http 4.5 Content-Server :

http://help.sap.com/saphelp_nw70ehp2/helpdata/de/9b/e8c186eaf811d195580000e82deb58/frameset.htm

### Python:

http://www.python.org/

### CherryPy:

http://www.cherrypy.org/

### OpenSSL:

http://www.openssl.org/

Die Verfahrensdokumentation zu bitfarm-Archiv DMS ist standardgemäß im Dokumentationen-Ordner im bitfarm-Programmverzeichnis verfügbar.

# 9. Datenübergabe im XML-Format, bitfarm XML-Importer

## a) Allgemein

Der Bitfarm-XML-Importer dient dem Import von Dokumenten, deren Metadaten in einer XML-Datei hinterlegt sind. Allgemein wird ein Import über folgenden Aufruf des Programms gestartet:

`bfa_xmlimport.exe <Pfad auf Konfigurationsdatei> <xml-Datei oder Pfad auf das Verzeichnis, welches xml-Datei(en) enthält>`

Per Konfiguration kann dabei gesteuert werden, in welches Archiv ein Dokument importiert wird und wie welche Metadaten an bitfarm übergeben werden. Die Konfigurationsdatei (ini-Datei) enthält immer einen Abschnitt `[main]`, in dem folgende grundlegende Variablen definiert werden.

- `profile=`Der Name des bitfarm-Profils
- `bfauser=`bitfarm-Benutzer, mit dem sich am bfaServer 36 angemeldet wird (verschlüsselt)
- `bfapass=`Passwort des bitfarm-Benutzers (verschlüsselt)
- `rootnode=`Wurzel-Tag
  Alle in der Konfiguration angegebenen XML-Knoten (Nodes) verwenden den hier definierten Knoten als Ursprung. Der Knoten muss dabei inklusive den umschließenden Tags angegeben werden, also z.B. <rootnode>.
- `document=`file oder <Bezeichnung des Nodes, in dem das Dokument referenziert ist>
  Hierüber wird gesteuert, wie das Dokument gefunden wird:
  - Das Dokument hat den gleichen Dateinamen, wie die importierte xml-Datei und liegt im gleichen Verzeichnis: `document=`file
  - Der Pfad auf das Dokument steht in einem Knoten der XML-Datei: `document=`<Name des Nodes>

    Hinweis: Der Pfad kann ein absoluter Pfad sein (mit Verzeichnis), ist nur der Dateiname angegeben, so wird das Dokument im gleichen Pfad wie die XML-Datei erwartet.
  - `bitfarmpath=`Pfad auf das bitfarm-Archiv-Programmverzeichnis (Server)
  - `templatepath=`Pfad auf den Templates-Ordner
  - `uebergabe=`Pfad auf *bitfarm-Archiv\uebergabe*
  - `logfile=`Pfad auf Logdatei
  - `loglevel=`info oder debug
  - `deletexml=`False oder True

Nach erfolgreichem Import wird die XML-Datei gelöscht (True) oder nicht gelöscht (False).

- `deletedoc=`False oder True

  Nach erfolgreichem Import wird die Dokumenten-Datei gelöscht (True) oder nicht gelöscht (False)

### b) Setzen von bfauser/bfapass

Der zu verwendende bitfarm-Benutzer und sein Kennwort werden in der Konfiguration verschlüsselt hinterlegt. Um das zu realisieren, starten Sie auf dem Rechner, auf dem der Import zukünftig ausgeführt werden soll, mit dem Windows-Benutzer, unter dem der Import gestartet werden soll, eine Kommandokonsole. Starten Sie in der Konsole das Programm `bfa_xmlimport.exe` und übergeben folgende Parameter: `-cred <Pfad auf die Konfigurationsdatei>`.

Nun werden Benutzer und Passwort abgefragt, welche Sie bitte beide in das Konsolenfenster eingeben und jeweils mit „Enter“ bestätigen. Im Anschluss werden sowohl der Benutzer als auch das Passwort automatisch in der Konfiguration in `[main]` gespeichert.

**Achtung: Wenn sich der Rechner auf dem der Import erfolgt ändert, oder der Windows-Benutzer für den Import geändert wird, muss der Vorgang wiederholt werden, um die Zugangsdaten zu aktualisieren.**

### c) Sortierung

In einem Abschnitt „Archive“ in der Konfiguration werden Bedingungen definiert, die angeben, in welches Archiv (Name des Templates) ein Dokument importiert wird. Trifft keine der angegebenen Bedingungen zu, wird die xml-Datei ignoriert und das Dokument nicht importiert. In jeder Zeile unterhalb der Sektion „Archive“ ist jeweils der Basisname eines Templates und die jeweilige Bedingung anzugeben. Soll ein Dokument zum Beispiel in das Archiv mit dem Templatenamen „Eingangsrechnungen“ importiert werden, wenn in der XML-Datei der Knoten `<Kopf><DokTyp>` den Wert „Rechnung“ hat, trägt man folgende Zeile in der Sektion ein:

`Eingangsrechnungen=(<Kopf><DokTyp>=Rechnung)`

Dabei kann in der Bedingung mit logischen Aussagen gearbeitet werden und es können die Schlüsselworte „`and`“, „`or`“, „`and not`“, „`or not`“, sowie Klammern verwendet werden, um z.B. Konstrukte wie ((`<Kopf><DokTyp>`=Rechnung) and (`<Kopf><Betreff>`= Sanierung)) zu ermöglichen. Jeder einzelne Vergleich sollte, ebenso wie der gesamte Ausdruck, dabei in Klammern gesetzt werden. Für den Vergleich von Werten können die Operatoren „`=`“, „`<`“, „`>`“, „`<=`“, „`>=`“, „`contains`“ und „`contains not`“, verwendet werden.

### d) Metadaten

Für jedes angegebene Archiv (genauer: Template-Bezeichnung) kann im folgenden ermittelt werden, wie welche Zusatzfeld- und Statusfeld-Werte aus den XML-Daten ermittelt werden. Für Zusatzfelder wird dazu eine neue Sektion „zus“ <Name des Templates> angelegt, für Statusfelder eine Sektion „stat“ <Name des Templates>.

### e) Statusfelder

Für das o.g. Beispiel nehmen wir an, dass das Archiv „Eingangsrechnungen“ (Eingangsrechnungen.tpl) über ein Statusfeld „gebucht“ verfügt. „gebucht“ soll gesetzt sein, wenn in der XML-Datei der Knoten <Kopf><verbucht> den Wert „Ja“ trägt. Die entsprechende Sektion sähe dann so aus.

```
[stat_Eingangsrechnungen]
gebucht=(<Kopf><verbucht>=Ja)
```

Für die Bedingung kann mit den gleichen Schnlüsselworten und Operatoren gearbeitet werden, wie in der Sektion „Archive“. Wird die angegebene Bedingung zu True ausgewertet, wird das Statusfeld aktiv gesetzt, sonst nicht.

### f) Zusatzfelder

Die Konfiguration für Zusatzfelder erfolgt wie die Statusfeld-Konfiguration auch pro Archiv in einer eigenen Sektion, deren Optionen allgemein so aufgebaut sind:

```
<Name des bitfarm-Zusatzfeldes>=<Wert>
```

<Wert> kann dabei in einfachsten Fall ein bestimmter Node der XML-Datei sein. Für das o.g. Beispiel nehmen wir an, dass das Archiv „Eingangsrechnungen“ über ein Zusatzfeld „Rechnungsbetrag“ verfügt. Das Feld soll den Wert aus dem XML-Knoten <Kopf><RgBetrag> erhalten. Die entsprechende Sektion sähe dann so aus:

```
[zus_Eingangsrechnungen]
Rechnungsbetrag=<Kopf><RgBetrag>
```

Der Wert kann auch mit beliebigem Freitext erweitert und/oder aus den Werten mehrerer Knoten zusammengesetzt werden. Nehmen wir fogende XML-Struktur an:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <testroot>
        <Unterknoten>
            <test1>
                t1
            </test1>
            <test2>
                t2
            </test2>
        </Unterknoten>
    </testroot>
```

Als Wert für das Zusatzfeld "Test" soll für diese xml nun "t1 - t2" gesetzt werden. Die Konfiguration dazu lautet dann folgendermaßen:

```
Test=<Unterknoten><test1> - <Unterknoten><test2>
```

Nodes werden in der Konfiguration immer in "<" und ">" eingefasst. Stehen in der Konfiguration dabei zwei Nodes direkt hintereinander, wird der zweite Node als Kind-Element des erstgenannten interpretiert. Sind die Nodes voneinander getrennt, werden sie als eigenständige Nodes auf gleicher Ebene behandelt. Im Beispiel-XML ist "test1" also ein Kind-Element von "Unterknoten". Wäre in der Konfiguration Folgendes hinterlegt:

*Test=<Unterknoten> <test1>*

sähe die zugehörige xml so aus:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<testroot>
    <Unterknoten>
        UK1
    </Unterknoten>
    <test1>
        t1
    </test1>
</testroot>
```

Das Zusatzfeld würde in diesem Fall den Wert "UK1 - t1" erhalten.

Bei der Konfiguration der Zusatzfeldwerte kann, neben dem einfachen Mapping eines Element-Wertes, auch mit einer Anweisung "List" oder einer Anweisung "If" gearbeitet werden, um das Auslesen mehrerer gleichnamiger Nodes (List) bzw. das bedingte Ermitteln eines Node-Wertes (If) zu erreichen.

### g) List-Anweisung für Zusatzfelder

In einer XML-Datei kann es mehrere gleichnamige Knoten geben, die unterschiedliche Daten enthalten. Damit festgelegt werden kann, aus welchem Knoten ein Zusatzfeldwert geholt werden soll, muss eine Bedingung definiert werden, die den Knoten eindeutig identifiziert.
Zur Veranschaulichung nehmen wir folgende beispielhafte XML-Struktur an:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Indexierung>
    <Kopf>
        <Adresse>
            <Satzart>Erfasser</Satzart>
            <AdressTyp>Mitarbeiter</AdressTyp>
            <Name1>Homer Simpson</Name1>
```

```
            <Strasse>Evergreen Terrace 742</Strasse>
            <Ort>Springfield</Ort>
        </Adresse>
        <Adresse>
            <Satzart>Bezogener</Satzart>
            <AdressTyp>Verkäufer</AdressTyp>
            <Name1>Morticia A. Addams</Name1>
            <Name2/>
            <Strasse>1313 Mockingbird Lane</Strasse>
            <Ort>Mockingbird Heights</Ort>
        </Adresse>
        <Adresse>
            <Satzart>Empfänger</Satzart>
            <AdressTyp>Kunde</AdressTyp>
            <Name1>Wile E. Coyote</Name1>
            <Strasse>Desert Road 12</Strasse>
            <Ort>Desert</Ort>
        </Adresse>
    </Kopf>
</Indexierung>
```

Wir sehen hier 3 Nodes mit der Bezeichnung "Adresse", deren Daten in jeweils ein Zusatzfeld gemappt werden sollen. Nehmen wir an, das Zielarchiv enthält u.a. 3 Zusatzfelder "Mitarbeiter", "Verkäufer" und "Kunde", die jeweils mit dem zugehörigen Wert von "Name - Ort" aus der XML-Datei befüllt werden sollen. Erwartete Werte:

Mitarbeiter: "Homer Simpson - Springfield"
Verkäufer: "Morticia A. Addams - Mockingbird Heights"
Kunde: "Wile E. Coyote - Desert"

Die allgemeine Syntax einer If-Anweisung lautet:

```
If((condition), <startnode>, <iternode>) <valuenode>
```

Die Konfiguration für das Beispiel lautet dann:

```
Mitarbeiter=If(<AdressTyp>=Mitarbeiter, <Kopf>, <Adresse>) <Name1> - <Ort>
Verkäufer=If(<AdressTyp>=Verkäufer, <Kopf>, <Adresse>) <Name1> - <Ort>
Kunde=If(<AdressTyp>=Kunde, <Kopf>, <Adresse>) <Name1> - <Ort>
```

Bei der Angabe der "condition" kann wiederum mit den bekannten Schlüsselworten und Operatoren gearbeitet werden.

Würde der Knoten "Adresse" im Beispiel-XML nur ein einziges Mal auftauchen, und wir wollen einen Wert einfach nur unter Bedingungen auslesen, kann bei der If-Anweisung auch auf die

Angabe von `<startnode>` und `<iternode>` verzichtet werden. Es wird lediglich die Bedingung benötigt. Wird diese zu `true` ausgewertet, wird der angegebene Value ausgelesen, sonst nicht. Bei Verwendung dieser Syntax wird dann allerdings auch nur der erste gefundene Knoten analysiert, weitere gleichnamige Knoten werden ignoriert.

### h) Plugin-Konfiguration

In der Konfigurationsdatei kann ein Abschnitt "Plugins" angelegt werden. Pro Archiv (nochmal der Hinweis: eigentlich handelt es sich um Template-Bezeichnungen) kann hier der Pfad auf ein Plugin (ausführbare Datei) angegeben werden, welches gestartet wird, bevor Dokument + Job-Datei an den Bitfarm-Server übergeben wird. Dem Plugin wird dabei die erzeugte Job-Datei übergeben.

Ein sinnvoller Einsatz für ein Plugin wäre z.B., wenn ein MYSQL-Timestamp aus einer XML-Datei gelesen wird, welches in ein Bitfarm-Datumsfeld übertragen werden soll. Für den erfolgreichen Import in Bitfarm muss das Datum im Job in der Form TT.MM.JJJJ angegeben werden. Ein Plugin könnte dafür Sorge tragen, dass der MYSQL-Timestamp vor der Übergabe an bitfarm in ein passendes Datum gewandelt wird.

# II. Übernahme von Dokumenteninformationen aus dem DMS in andere Systeme

## 1. Job-Datei

In der `scripts.ini` kann mit
`autoexportpath=`
ein Pfad für den Metadatenexport definiert werden. Bei jedem neuen Dokument, welches in das DMS kommt, werden die Informationen, welche in die Datenbank geschrieben werden, in Form einer .job-Datei in den Exportordner gelegt. Diese Informationen können dann in anderen Systemen verwendet werden und ermöglichen z.B. die Konfiguration von ERP-/Buchhaltungssystemen als führende Anwendungen. Das Aufräumen der ausgelesenden Dateien muss ebenfalls dann vom externen System übernommen werden. Hier ein Beispiel für eine solche exportierte .job-Datei:

```
[Version]
Version=36
[Archiv]
Profil=muster-gmbh
Name=Kreditoren
Tabelle=28052015172108
Arcid=10
[Document]
DOCID=10.531
gdoc_id=531
```

```
Titel=Rechnung Ingram Micro-09586-11
Orginal=%bfastore0%\daten-rs\2019\09\20\1992011957NBLK236825147.tif
Kopie=
Quelle=
Benutzer=m.mustermann
userid=
Volltext=%bfastore0%\daten-rs\2019\09\20\1992011957NBLK236825147.txt
Status=0
StatStr=Dokument indiziert und archiviert.
OCR_QF=-1
OCR_Typ=OMP
Schlagworte=
Datum=2019-09-20 11:09:34
Filter=
Pages=1
[Hash]
SHA1=
[Referenz]
Anzahl=0
[SVN]
SVN_Link=
SVN_Version=
SVN_Revision=
SVN_Datum=
SVN_Info=
[Tasks]
Termin=
Alarm=
Bearbeiter=
Notiz=
TimerOptionen=0
[Zusatzfelder]
Felder=17
ZusTitel1=Auftragsnummer
ZusFeld1=add_5
ZusWert1=09586-11
ZusTitel2=Rechnungsnummer
ZusFeld2=add_3
ZusWert2=44-3281708
ZusTitel3=Rechnungsdatum
ZusFeld3=add_2
ZusWert3=15.10.2019
ZusTitel4=Rechnungsbetrag
ZusFeld4=add_14
ZusWert4=622,37
ZusTitel5=Rechnungsprüfer
ZusFeld5=add_14
ZusWert5=m.mustermann
…
```

Interessant für externe Systeme ist ggf. die gdocid, da hierüber direkt das Dokument aus externen Systemen über den DMS-Client aufrufbar ist. Über die Zusatzfelder, die auch z.B. eine Barcodenummer mitführen können, kann die Verbindung des Dokuments in externen Datenbanken hergestellt werden.

## 2. Excel-Export über konfigurierbares Viewer-Plugin

Metadaten, welche sich in Zusatzfeldern befinden, können grundsätzlich per Rechtsklick, z.B. nach Excel oder als csv-Datei, exportiert werden. Allerdings werden dabei immer alle Zusatzfeldwerte und auch dokumentspezifische Metdaten exportiert, welche Sie womöglich nicht für Ihren Export benötigen. Mit dem Excel-Export Viewer-Plugin, können Sie genau definieren, welche Zusatzfeld-Werte exportiert werden sollen.

### a) Vorbereitung

Verschieben Sie den Inhalt des Ordners Excel-Export, welchen Sie in *%bitfarm-archiv%\install\Bitfarm-Tools\Viewer-Plugins\Excel-Export* finden, in das Verzeichnis *%bitfarm-archiv%\Viewer-files\plugins*

Achtung: Verwenden Sie zum Bearbeiten der Konfigurationsdateien unbedingt einen Editor, der UTF-8 Kodierung unterstützt (kein Windows Editor)

### b) Konfiguration

```
'######## hier anpassen ################
exportpath="askuserfile"
hyperlinks=True
expandmulti=False
```

`Excel-Export.vbs`

Achtung: Verwenden Sie für die `Excel-Export.vbs` die Kodierung ANSI

`exportpath="askuserfile"`

Sie können Zwischen drei verschiedenen Optionen wählen. Mit „`askuser`“ wird der Benutzer nach einem Pfad gefragt in welchen exportiert werden soll. „`askuserfile`“ fragt nach dem Pfad + der Dateiendung. Hier ist es wichtig, dass auch die Endung jedes Mal mitgegeben wird. Geben Sie einen UNC-Pfad an und es wird direkt in dem angegebenen Pfad, ohne Rückfrage, gespeichert.

`hyperlinks=True`

Hyperlinks werden benutzt, um via Klick auf den Link, direkt zu dem Dokument im bitfarm-DMS zu gelangen. Ist dieser auf `True` gestellt werden die Hyperlinks mit exportiert, bei False stehen nur die Gdoc-IDs ohne Link in der Datei.

`expandmulti=True`

Wenn Mehrfachauswahlfelder in einzelne Spalten exportiert werden sollen, können sie hier True angeben, ist dies nicht erwünscht, `False`

Im darunterliegenden Bereich müssen zusätzlich die Pfade Ihrer Umgebung entsprechend angepasst werden.

```
'#####################################
'#################### hier anpassen ####################
sourcedir = "\\vmrb2\bitfarm-archiv\Viewer-files\plugins\excel_export_gui"
installdir = appdata & "\bitfarm-archiv\plugins\excel_export_gui"
localexe = installdir & "\excel_export_gui.exe"
remoteexe = sourcedir & "\excel_export_gui.exe"
updatecheck = True
startlocal = True
'#########################################################
```

`Plugins.ini`

In der `plugins.ini` muss der `counter` angepasst werden. Für jedes Plugin wird dieser Wert um eins erhöht.

```
[Conf]
counter=1
[Names]
file0=Excel Export.vbs
[Params]
file0=jobfile
```

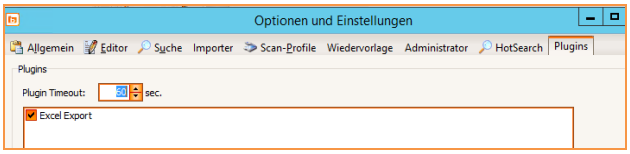
Unter `[Names]` wird das auszuführende Skript angegeben, in unserem Beispiel `Excel Export.vbs`

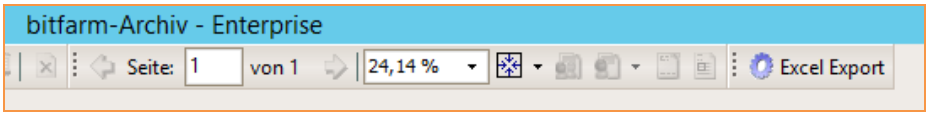
Hinweis: Geben Sie den kompletten Namen der Datei an, die ausgeführt werden soll.

`file0` gibt dabei das erste Plugin an. Für jedes weitere wird ein neuer Eintrag erstellt, so heißt das zweite Plugin file1. In der Sektion `[Params]` werden dem Plugin Parameter übergeben. In diesem Fall wird dem Plugin die Job-Datei zum jeweiligen Dokument, auf welches das Plugin ausgeführt wird, übergeben. Welche Übergabeparameter es gibt, können Sie dem Systemhandbuch entnehmen oder beim bitfarm-Archiv Softwaresupport erfragen.

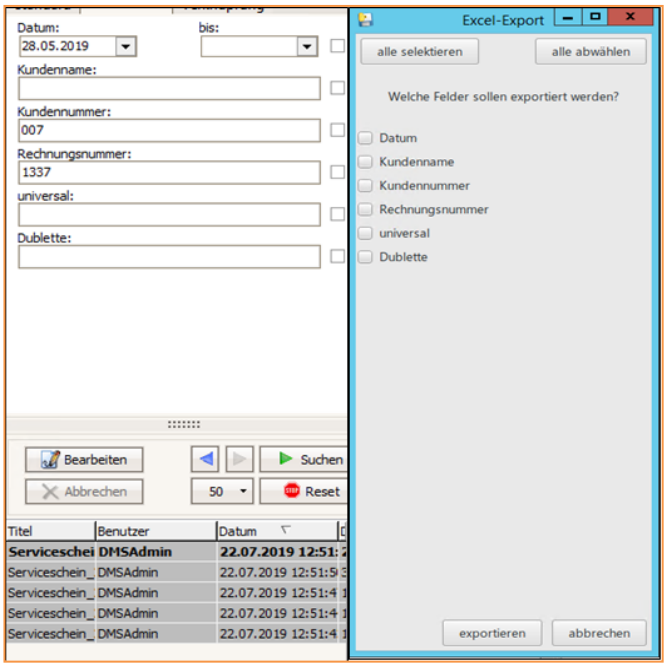
### c) Ausführung des Plugins

Haben Sie ein oder mehrere Dokumente in der Trefferliste, können Sie das Plugin entweder per Rechtsklick aus dem Kontextmenü asuwählen oder über eine entsprechnende Schaltfläche im Viewer. Wenn Sie ein Viewer-Plugin als „Knopf“ im Viewer angezeigt bekommen wollen, so müssen Sie dies in den Viewer-Optionen konfigurieren.




Der erste Start des Plugins dauert etwas länger, das Plugin wird beim ersten Start auf dem Client installiert. Bei erfolgtem Start sehen Sie ein Auswahlfenster, in welchem Sie Zusatzfelder an und abwählen können.




| | A | B | C |
|---|---|---|---|
| 1 | gdocid | Kundennummer | Rechnungsnummer |
| 2 | gdocid:34 | 911 | 2600 |
| 3 | gdocid:38 | 007 | 1337 |
| 4 | gdocid:37 | 0815 | 109876543210 |
| 5 | gdocid:36 | 66 | 420 |
| 6 | gdocid:35 | 69 | 96 |

Sind die Zusatzfelder ausgewählt, klicken Sie auf „exportieren“. Je nach Konfiguration sehen Sie ein neues Fenster, in welchem Sie auswählen können, wohin die Datei gespeichert werden soll. Im Anschluss können Sie die Excel-Datei öffnen, welche die von Ihnen ausgewählten und exportierten Metadaten enthält.

## 3. Erweiteter CSV-Export über konfigurierbares Viewer-Plugin

Immer wieder gibt es die Notwendigkeit, archivierte Daten und/oder deren Metadaten, einem anderen Programm zur Verfügung zu stellen, z.B. einer Fibu-Software. Die gewünschten Metadaten zum Dokument können mit dem Tool bfa_csv_export als .csv-Datei, wahlweise mit Bilddatei und Setzen eines Status, exportiert werden.

### a) Vorbereitung

Hinweis: Für die Einrichtung des Tools muss eine kompatible Version des Bitfarm-Archiv DMS installiert sein. Sollte auf Ihrem System eine veraltete oder inkompatible Version installiert sein, aktualisieren Sie diese bitte mithilfe Ihres Kundenzugangs auf der bitfarm-Archiv Webseite und den jeweiligen Paketen der Enterprise-Version.

Das Tool befindet sich im Verzeichnis *%bitfarm-archiv%\install\Bitfarm-Tools\Viewer-Plugins\bfa_csv_export.*

Achtung: Verwenden Sie zum Bearbeiten der Konfigurationsdateien des bfa_csv_export unbedingt einen Editor, der UTF-8 Kodierung unterstützt (kein Windows-Editor).

Verschieben Sie den Ordner und die Datei bfa_csv_export in das Verzeichnis *%bitfarm-archiv%\viewer-files\plugins.*

### b) Konfiguration

In der Datei `bfa_csv_export.vbs` muss der Bereich, der mit „hier anpassen“ markiert ist, bearbeitet werden. Achten Sie darauf, dass Sie für diese Datei die **Kodierung ANSI** verwenden. Sollten Sie keine Veränderungen an den Pfaden vorgenommen haben, kann die Einstellung beibehalten werden.

Wenn Sie die Ordner nicht wie vorgegeben in das *Viewer-files\plugins*-Verzeichnis eingefügt haben, müssen Sie hinter `configfile=` und `sourcedir=` den Dateipfad auf die

```
'#################### hier anpassen ####################
configfile = clientprogram&"Viewer-files\plugins\bfa_csv_export\bfa_csv_export.ini"
sourcedir = clientprogram&"Viewer-files\plugins\bfa_csv_export"
installdir = appdata & "\bitfarm-archiv\plugins\bfa_csv_export"
updatecheck = True
startlocal = True
'#######################################################
```

`bfa_csv_export.ini` und das Verzeichnis indem die `bfa_export.ini` liegt, angeben.

Hinter `updatecheck=` und `startlocal=` werden die Eigenschaften für den Client angegeben. Wenn das Tool lokal auf dem Client starten soll, muss unter `startlocal=True` eingetragen werden. Mit `updatecheck=` wird eingestellt, ob das Tool mit der Serverversion abgeglichen und bei Updates der Serverversion entsprechend aktualisiert wird.

### c) bfa_csv_export.ini

**Achtung:** Bearbeiten Sie die INI nur mit UTF-8 Kodierung.

Öffnen Sie die `bfa_csv_export.ini` in dem Verzeichnis %bitfarm-archiv%\Viewer-files\plugins\bfa_csv_export\ mit einem geeigneten Editor.

### d) [main]

```
[main]
exportdir=C:\datev-export
export_csv=True
export_tif=True
export_pdf=True
tif_filter=\\vmrb2\bitfarm-archiv\TIFF-300-bw.txt
logfile=\\vmrb2\bitfarm-archiv\bin\logs\bfa_csv_export_%user%.log
datetimeformat=%Y-%m-%d-%H%M%S.%f
dateformat=%Y-%m-%d
exportfile=%archiv%_%date%
showmsg=True
multidoc_csv=False
```

Die Sektion `[main]` bildet den Hauptabschnitt der ini-Datei. Mit der Variable `exportdir` geben Sie dabei das Verzeichnis an, in die das Tool die Dokumente speichern soll. Geben Sie hier einen Pfad an, der vom Client aus erreichbar ist.

Mit den drei nachfolgenden Schaltern `export_csv`, `export_tif` und `export_pdf` entscheiden Sie, welche Dateitypen Sie speichern wollen.

Mit `tif_filter` kann eine Richtlinie für `TIF`-Dokumente angegeben werden. Sie finden mehrere Beispieldateien (TIFF-100.txt, etc.) in Ihrem *bitfarm-Archiv*-Verzeichnis. Sie können auch eigene Filterrichtlinien schreiben. Eine Beschreibung und die dazugehörigen Befehle finden Sie in einer der vorhandenen Filterdateien.

Protokolle, die beim Ausführen des Tools geschrieben werden, werden in `logfile` angegeben. Geben Sie hier einen Pfad an, in welchem die Protokolle gespeichert werden sollen. Unsere Empfehlung ist diese auf dem Server in dem Verzeichnis `Logs` zu speichern. Dabei können Sie etwa den Benutzer des Tools als Parameter im Dateinamen mit angeben, wie in der Beispielabbildung. Datumsformate können Sie mit `datetimeformat` und `dateformat` ändern.

Hinweis: Das Datumsformat folgt einem streng gebundenem Aufbau. Dieser kann nachgelesen werden, indem Sie sich in die Python Funktion strftime einlesen. Ändern Sie nichts, wenn Sie mit dieser Funktion nicht vertraut sind.

Die Namensgebung der zu exportierenden Dateien erfolgt mit `exportfile`. Dabei können Sie verschiedenste Parameter für den Dateinamen angeben, wie etwa den Archivnamen, das Datum, aber auch Zusatzfeldinfos. Übergeben Sie dabei den entsprechenden Wert mit %-Zeichen, etwa `%archiv%.` Weitere Möglichkeiten sind u.a. `%gdocid%, %date%, %datetime%` oder `%Zusatzfeldname%`. Soll dem Benutzer nach dem Export eine kurze Nachricht in Form einer MessageBox angezeigt werden, so stellen Sie den Schalter `showmsg` auf `True`. In manchen Fällen ist das Verabreiten von einer CSV-Datei für verschiedene Buchungssätze sinnvoller, statt einzelne CSV-Dateien für jeden Vorgang einzulesen. Damit das Tool eine CSV-Datei für alle im Viewer ausgewählten (markierten) Dokumente erstellt, muss der Schalter `multidoc_csv` auf `True` gestellt werden.

Hinweis: Bei Multidoc Exporten muss `export_TIF` und `export_PDF` auf `False` gestellt sein.

### e) [csv]

Achtung: In der Sektion `[csv]` werden die verschiedenen csv-Parameter definiert.

**Hier sollten im Regelfall keine Änderungen notwendig sein.**

In der Sektion `[csv]` werden Trennzeichen, Escape Zeichen, Zeilenenden und Quote´s

```
[csv]
delimiter=";"
doublequote=False
escapechar="\"
lineterminator="\r\n"
quotechar="""
quoting=QUOTE_ALL
skipinitialspace=False
```

angepasst. `delimiter` beschreibt das Trennzeichen. Achten Sie darauf, dass es immer in Anführungszeichen steht. Mit `doubleqoute` wird geregelt, wie ein Objekt in dem

festgelegten Zeichen (quotechar) gekennzeichnet wird. Wird doublequote auf True gesetzt, wird das Objekt, in die von quotechar gesetzten Zeichen, mit doppelten Quoting-Zeichen gesetzt. Auf False wird es in einfache quoting-Zeichen gesetzt. Ist doublequote auf False gesetzt muss ein escapechar vorhanden sein. Mit der Variablen escapechar= wird angegeben, welches Zeichen dazu genutzt werden soll, um ein anderes Zeichen als besonderes Zeichen zu interpretieren.

Die Eingabe für das Ende einer Zeile wird in lineterminator festgelegt. Als Standard ist dort \r\n eingestellt. Mit dem Befehl quotechar können Sie das Markierungszeichen definieren. Geben Sie dies immer in Anführungszeichen (“‘”, “#”, “””) an. Mit quoting geben Sie an, was alles in die angegebenen Quoting-Zeichen gesetzt werden soll. Hierbei gibt es die Auswahlmöglichkeiten QUOTE_ALL, welches alle Objekte in Zeichen setzt, QUOTE_MINIMAL welche Objekte mit special chars in Quoting-Zeichen setzt, QUOTE_NONNUMERIC um alle Nummerntypen in einen float umzuwandeln und QUOTE_NONE welches, falls kein escapechar angegeben ist zu Error Meldungen führt, keine Quoting-Zeichen setzt.

### f) [header]

Die Sektion [header] beinhaltet die Überschriften die in der CSV Datei gesetzt werden.
Mit den einzelnen Buchstaben werden die Spalten angegeben.

Hinweis: Wenn Sie die CSV-Datei für das Fibu-System der Fa. DATEV eG. erstellen und erzeugen wollen, so benötigen Sie einen für die jeweilige DATEV-Version festgeschriebenen Header der bei jedem Export so zu sein hat. Fragen Sie hierzu Ihren DATEV-Experten.
Als Beispiel eine mögliche Konfiguration

```
[header]
A=EXTF
B=700
C=21
D=Buchungsstapel
E=11
H=BitFarm-Archiv
K=Beraternummer
L=Mandantennummer
M=%year%0101
N=5
O=%date%
P=%date%
S=1
T=0
U=0
V=EUR
```

In der Sektion [row2] können weitere Überschriften erstellt werden. Diese stellen letztendlich die Bezeichnungen der Datenbankfelder dar, die Sie womöglich so ähnlich auch als Zusatzfelder im bitfarm-Archiv DMS angelegt haben. Damit die Zuordnung beim Import der CSV-Datei in Ihr Fibu-System problemlos funktioniert, achten Sie unbedingt darauf hier die Bezeichnungen Ihres Fibu-Systems zu nehmen. Nachfolgend ein Ausschnitt typischer DATEV Bezeichnungen für die Sektion [row2].

```
[row2]
cols=Umsatz (ohne Soll/Haben-Kz)|Soll/Haben-Kennzeichen|WKZ Umsatz|Kurs|Basis-Umsatz|WKZ Basis-Umsatz|Konto|Gegenkonto (ohne BU-Schlüssel)|BU-Schlüssel|Belegdatum|Belegfeld 1|Belegfeld 2|Skonto|Buchungstext|Postensperre|Di
```

Dabei werden die Spalten nicht mehr in „A, B oder C“ angegeben, sondern durch einen senkrechten Strich (Pipe). Benutzen Sie [row2] nicht, um die Zusatzfeldwerte zu exportieren, dies

geschieht in der Darunter folgenden Sektion `[cols].` In dieser werden wieder Buchstaben zu den Spalten angegeben.

### g) [cols]

Die Variablen werden dabei in Prozentzeichen gesetzt, um bspw. die GDocID anzeigen zu lassen, wird `%gdocid%` in das Feld geschrieben. Vor und hinter die Variablen kann auch statischer Text geschrieben werden. Für Zusatzfelder die exportiert werden sollen gilt, jedes Zusatzfeld muss in Prozentzeichen gesetzt werden.

```
[cols]
; alle Variablen werden in % eingefasst (%archiv%, %gdocid% oder %Name des Zusatzfelds%)
; per Semikolon getrennter zweiter oder folgender Wert -> neue Zeile in csv wenn eins der Felder an der Position einen Wert trägt
A=%gdocid%
B=%archiv%
C=%Belegdatum%
D=%Lieferant%
E=%Rechnungsnummer%
```

### h) [grid_data]

Falls Sie auch Metadaten aus einem für das Archiv konfigurierten Grid exportieren möchten, in welchem Sie bspw. Ihre Splittbuchungen Zeile für Zeile erfasst haben, so legen Sie das Mapping der entsprechenden Werte in der Sektion `[grid_data]` fest.

Hinweis: Die Erstellung eines Grids ist im Systemhandbuch beschrieben.

```
[grid_data]
; each -> jede Buchungszeile wird um Zusatzfeldwerte erweitert
; first, last -> Zusatzfeldwerte vor/nach den Buchungsdaten

grid_field=Kontierung
doc_data=last
F=WKZ
G=Menge
H=Betrag
I=Stückpreis
J=SH-Kennzeichen
K=Steuersatz
L=Buchungstext
M=Konto
N=Gegenkonto
```

Hinter `grid_field=` tragen Sie den Namen Ihres Grids ein. Sollten Sie unsicher sein, so können Sie in der Administration des DMS nachschauen. Hinter `doc_data=` können Sie entscheiden, ob in der CSV-Datei die Grid-Daten vor oder nach den „normalen“ Zusatzfeld-daten stehen sollen.

### i) [conditions]

Die Sektion `[conditions]` wird benötigt um Bedingungen zu erschaffen. Mit diesen werden dann Voraussetzungen gesetzt, unter welchen das Tool Exportieren darf.

```
[conditions]
archive=Eingangsrechnungen
stat_gebucht=False
stat_ungültig=False
stat_geprüft=True
stat_freigegeben=True
zus_Rechnungsbetrag=!=|
zus_Lief-Name=!=|
zus_Rechnungsnummer=!=|
zus_Rechnungsdatum=!=|
```

Diese Variablen können wie folgt eingesetzt werden:

archive= gibt an in welchen Archiven das Tool ausgeführt werden darf. Es können mehrere Archive semikolongetrennt angegeben werden. Für alle conditions gilt, ist nichts gesetzt darf überall ein Export gestartet werden. Um den Export an Zusatzfelder zu binden benötigt man lediglich den oder die Namen des oder der Zusatzfelder, die man wie folgt abfragen kann zus_Rechnungsbetrag=. Das erste Gleichheitszeichen bezieht sich auf die Variable. Der Operator, hinter dem ersten Gleichheitszeichen, != prüft das Zusatzfeld auf nicht leer. Der senkrechte Strich (Pipe) dient als Trennzeichen zwischen Operator und Wert (in diesem Falle ein leerer Wert). Sobald eines der im o.g. Zusatzfelder leer ist, wird der Export unterbunden.

Statusfelder können ebenso abgefragt werden. Dazu verwenden Sie stat_geprüft wobei geprüft jedem Statusfeldnamen, der überprüft werden soll, zugewiesen werden kann. Weitere Operatoren für die Zusatzfeldüberprüfung sind:

```
[update]
movefirst=False
moveto=78
zus_Datum=%date%
cast_Gesamtbetrag=string_to_decimal
stat_Geprüft=True
```

= welches auf Gleichheit prüft, als Beispiel: zus_Kundennr==|57893

< kleiner, welches alle Werte kleiner des angegebenen Wertes überprüft

> größer, welches nur Werte die größer als der angegebene Wert sind zulässt

~ enthält, eine Zeichenkette die in diesem Zusatzfeld vorkommen muss, als Beispiel: zus_Kundennr=~|89. Es werden alle Dokumente exportiert, in welchen eine 89 vorkommt, dazu zählt auch die Kundennr 57893.

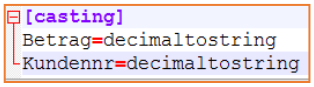
.> beginnt mit, jedes Zusatzfeld welches mit z.B. 57 beginnt

.< endet mit, jedes Zusatzfeld welches mit z.B. 93 endet.

### j) [update]

Mit der Sektion [update] können Dokumentenmetadaten, nach dem erfolgten Export, aktualisiert werden. Sollen Dokumente vor dem Abändern der Zusatz- und Statusfelder verschoben werden, setzen Sie movefirst= auf True, ist dies nicht gewünscht verwenden Sie False. Um das Archiv, in welches mit moveto= verschoben werden soll, zu wählen, benötigen Sie die Archiv-ID. Diese kann im Administrationsbereich der Applikation eingesehen werden. Die ausgewählte Archiv-ID tragen Sie hinter der Variable moveto= ein. Mit den nachfolgenden Variablen können Sie arbeiten, um Status- und Zusatzfelder abzuändern. Nach dem Export ist es von Vorteil ein Statusfeld, welches auf exportierte Dokumente verweist, zu setzen, etwa den Status exportiert. Nutzen Sie hierfür den Befehl stat_Exportiert=. Es können alle vorhandenen Statusfelder abgeändert werden, dazu ersetzten Sie Exportiert mit dem Statusfeld Ihrer Wahl. Dies geschieht indem Sie diesem True übergeben. Sie können auch Zusatzfelder nach dem Export abändern, mit zus_Datum= können sie z.B. das Zusatzfeld Datum abändern, mit %date% wird diesem das Datum des Exporttages übergeben.

Falls Sie Werte abändern möchten, die in ein anderes Format gebracht werden müssen, benutzten Sie entsprechende *caster*. Achten Sie darauf, dass Sie dort die richtige Funktion übergeben. Für die unterschiedlichen Behandlungen können sie beliebige eigene caster definieren, indem Sie die Datei `casting.py` entsprechend anpassen. Hier sind auch schon einige Funktionen vorgegeben, an denen Sie sich orientieren können.




### k) [casting]

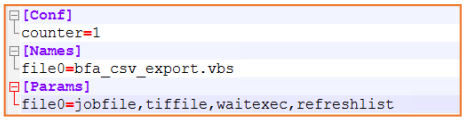
Die Sektion `[casting]` ändert das Zeichenformat. Für jedes Zusatzfeld welches angegeben wird kann eine Abänderung erfolgen. So können für den Export beispielsweise Betragswerte, die in bitfarm als dezimaler Wert gespeichert werden, in einen String abgewandelt werden.

Das Ganze geschieht mit dem exemplarischen caster `decimaltostring`. Nummerische Werte sollten für den Export unbedingt in einen String abgeändert werden, da sonst führende Nullen gestrichen werden können. Dies kann zu ungewolltem Verhalten und sogar Fehlern führen.

Der Aufbau der Einträge in der Sektion casting ist dabei wie folgt einzuhalten: `Zusatzfeldname=Variable` des Casters.

Tipp: Stellen Sie der Benutzergruppe die diesen Export erstellen soll, ein Globales Lesezeichen zur Verfügung, welches alle Dokumente anzeigt, die exportiert werden können. Auf diese Trefferliste oder auch nur auf einem Teil davon können die Benutzer dann das Viewer-Plugin für den konfigurierten CSV-Export ausführen.

### l) Plugins.ini




Viewer-Plugins werden über die Datei `%bitfarm-Archiv%\Viewer files\plugins\plugins.ini` angebunden. In der `plugins.ini` muss ggf. der `counter` angepasst werden. Für jedes auszuführende Tool wird dieser um eins erhöht.

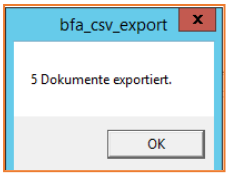
Unter `[Names]` wird das auszuführende Script (mit Endung) angegeben, in unserem Beispiel `bfa_csv_export.vbs`. Geben Sie hier den kompletten Namen der Datei ein, die ausgeführt werden soll. `file0` gibt dabei das erste Tool an, für jedes Weitere wird ein neuer Eintrag erstellt, so heißt das zweite Tool `file1`. In der Sektion `[Params]` werden dem Plugin Parameter übergeben. `jobfile` übergibt dem Plugin dabei die Metadaten des Dokumentes. Mit `tiffile` wird angegeben, das dem Plugin das TIF mit übergeben werden soll. `waitexec` pausiert den Viewer sobald der Export gestartet wurde. Damit wird verhindert, das es zu korrupten Daten kommt. `refreshlist` lädt das Archiv neu, sobald das Tool ausgeführt wurde. Sollten Status und Zusatzfelder abgeändert werden, sind diese nach dem Export direkt zu sehen. Der Parameter `tiffile` ist hierbei als zwingend erforderlich zu betrachten. Die weiteren Übergabeparameter und deren Anwendung erfahren Sie in diesem Kapitel. Viewer-Plugins

### m) Viewer-Konfiguration

Starten sie den Viewer und Öffnen Sie die Optionen (Datei -> Optionen), in dem Fenster Optionen wählen Sie den Tab Plugins aus. Hier fügen Sie das Tool, durch setzen eines Hakens vor dem Toolnamen, in die Schnellauswahl mit ein.


Alternativ können Sie das Tool mit einem Rechtsklick auf das ausgewählte Dokument unter Plugins ausführen.

Wählen sie ein oder mehrere Dokumente in einem Archiv aus und Starten sie das Tool. Nach erfolgreichem Export sollten Sie, entsprechende Einstellung in der `bfa_csv_export.ini` vorausgesetzt, folgende Abbildung sehen.




# III. Übernahme von Metadaten aus Fremddatenbanken

## 4. bfa_sqlmapper

### a) Anwendungsbereich

Üblicherweise werden die Metadaten eines Dokuments von Benutzern eingegeben oder mit Hilfe von Verschlagwortungsregeln (wfd-Datei) aus dem Volltext extrahiert. Oftmals sind zu Dokumenten aber auch die Metadaten von Interesse, die dem Dokument nicht direkt zu entnehmen sind, sondern in externen Datenbanken vorliegen. Der SQL-Mapper kann dazu verwendet werden externen Datenbanken per ODBC abzufragen und die erlangten Werte zum Dokument zu speichern.

Für eine möglichst flexible Anbindung kann der SQL-Mapper auf 3 verschiedenen Weisen angebunden werden.

- Angebunden über eine neue Anweisung „sqlmapper“ in wfd-naming sections
- Vom Endanwender als Plugin aus dem Viewer heraus gestartet
- Als standalone Programm, z.B. manuell oder wiederholt per geplantem Task

### b) Anwendungsbeispiele

- Zu einem Dokument wird per wfd-Regel eine Kundennummer ausgelesen. Der Name des Kunden ist nicht im Dokument vorhanden, er soll aus einer vorhandenen Datenbank ermittelt werden, in der zu jeder Kundennummer der Name gepeichert ist.
- In bitfarm-Archiv werden Angebote archiviert, die zugehörige Angebotsnummer wird aus dem Volltext ausgelesen und in einem Zusatzfeld gespeichert. Neben den Angeboten werden auch Aufträge im DMS archiviert, die mit einer Auftragsnummer versehen sind, welche ebenso in einem Zusatzfeld gespeichert ist. Zusätzlich zu der Auftragsnummer soll nun ein Feld befüllt werden, in dem die zum Auftrag gehörende

Angebotsnummer erfasst werden soll. Diese Information ist in einer externen Datenbank gespeichert.

### c) Einstellungen/Konfiguration

Die Konfiguration des sql-mappers erfolgt per ini-Datei (standardmäßig in *%bitfarm-archiv%\bfa_sqlmapper.ini*). Beim Bearbeiten der Konfigurationsdatei muss darauf geachtet werden, dass ein geeigneter Editor verwendet wird, um das Encoding der Datei (UTF-8) nicht zu verändern. So ist z.B. Notepad dafür bekannt, das Enconding beim Speichern zu verändern. Wir empfehlen den Einsatz eines modernen Text-Editors wie z.B. Notepad++. In der Datei werden neben allgemeinen Einstellungen, wie z.B. die Logging- und auch die Verbindungsdaten zur externen Datenquelle, als auch die entsprechenden SQL-Queries und Mappings zu bitfarm-Archiv hinterlegt. Der SQL-Mapper benötigt außerdem Zugangsdaten zur Anmeldung am bitfarm-Server.

Die beiden folgenden Konfigurationsabschnitte werden grundsätzlich benötigt.

```
[main]
bfauser=AQAAANCMnd8BFdERjHoAwE/Cl+…
bfapass=AQAAANCMnd8BFdERjHoAwE/Cl+…
profile=muster362

[logging]
viewerplugin=%appdata%\bitfarm-archiv\plugins\bfa_sqlmapper.log
standard=%bitfarm-archiv%\bin\logs\bfa_sqlmapper.log
standalone=%bitfarm-archiv%\bin\logs\bfa_sqlmapper.log
level=DEBUG
clientlog=False
backupcount=7
```

Im Abschnitt `[main]` werden neben dem Profilnamen die Zugangsdaten zur Anmeldung am bitfarm-Serverdienst gespeichert. Die Anmeldedaten werden dabei verschlüsselt hinterlegt. Zum Verschlüsseln der Daten verwendet man das Kommandozeilen-Tool `credcrypt.exe`, welches sich im Verzeichnis *%bitfarm-archiv%* auf dem bitfarm Server befindet. Melden Sie sich auf dem Server mit dem Benutzer an, unter dem die bitfarm-Dienste laufen und öffnen eine normale Konsole im Ordner *%bitfarm-archiv%.* Geben Sie `credcrypt.exe` ein, gefolgt vom Benutzernamen, mit dem sich der bfa_sqlmapper anmelden soll (z.B. DMSAdmin). Der verschlüsselte Benutzername wird auf der Konsole ausgegeben und zusätzlich auch in der Zwischenablage gespeichert, übertragen Sie ihn in die Konfiguration hinter `bfauser=`. Auf die gleiche Weise erzeugen Sie nun das verschlüsselte Passwort des Benutzers und tragen es hinter `bfapass=` ein.

**Achtung:** Zum Ver- bzw. Entschlüsseln wird die Windows Data Protection API verwendet (mehr Infos unter https://msdn.microsoft.com/en-us/library/ms995355.aspx). Wenn der bitfarm-Dienstbenutzer geändert wird oder der Server auf neue Hardware umgezogen wird,

müssen bfauser und bfapass neu verschlüsselt werden, auch wenn sich weder Benutzer noch Passwort geändert haben!

Im Abschnitt `[logging]` können folgende Optionen gesetzt werden:

- `viewerplugin/standard/standalone`: Für jeden Kontext kann ein individueller Pfad für die Log-Datei angegeben werden. Achten Sie darauf, dass es sich bei dem Pfad auf die Log-Datei im Kontext viewerplugin, um einen Pfad aus Sicht des Clients handelt!
- `level`: hierüber kann gesteuert werden, wie ausführlich geloggt werden soll. Mögliche Werte sind hier „DEBUG“ (ausführlich) und „INFO“ (weniger ausführlich)
- `clientlog`: Steuert, ob auch Informationen zur Kommunikation zwischen sql-mapper und bfaServer protokolliert werden sollen.
- `backupcount`: Pro Tag wird eine Logdatei erstellt. Über backupcount kann gesteuert werden, wie viele Logdateien als Backup vorgehalten werden sollen.

In weiteren Sektionen werden die anwenderspezifischen Konfigurationen vorgenommen.
Beispiel-Konfiguration (minimal):

```
[hole-Kundenname]
constring=dsn=kundendb-odbc
sql=select Kundenname from kundendb where kundennummer=%Kundennummer%
Kundenname=%0%
overwrite_Kundenname=True
emptyonmissing=True
emptyonnoresult=True
```

Der Name der Sektion kann frei gewählt werden, muss aber innerhalb der Konfiguration eindeutig sein. Zunächst muss die ODBC-Datenquelle im Betriebssystem hinterlegt werden. Die Konfigurationsoberfläche dazu lässt sich unter Windows über den Befehl *%windir%\syswow64\odbcad32.exe* starten (starten Sie auf jeden Fall die 32bit-Variante, auch auf 64bit Betriebssystemen).
Wechseln Sie in dem Konfigurationsdialog auf den Reiter „System-DSN“ und klicken auf „Hinzufügen“. Im folgenden Dialog muss zunächst der Treiber ausgewählt werden (ggf. muss dieser nachinstalliert werden, im Zweifel fragen Sie beim Hersteller der externen Datenbank nach), im darauffolgenden Dialog werden neben dem Data Source Namen weitere Angaben für den ODBC-Zugriff hinterlegt. Vergewissern Sie sich mit einem Klick auf „Test“, dass die Konfiguration korrekt ist und eine Verbindung möglich ist.
Der Verweis auf die neu eingerichtete ODBC-Verbindung muss nun in der Option „constring“ als connection-string hinterlegt werden. Wie dieser aussehen muss, hängt vom Typ/Treiber der Datenquelle ab, ggfs. ist diese Information beim Anbieter zu erfragen. In vielen Konstellationen reicht die Angabe des dsn in der Form: constring=dsn=<Data Source Name>. Weitere Informationen liefert z.B. die Seite https://www.connectionstrings.com.
Der sql-mapper kann nach Einrichtung der ODBC-Verbindung und Anlegen der Grundkonfiguration die Erreichbarkeit der externen Datenbank testen. Öffnen Sie dazu auf dem bitfarm-Server eine Konsole im Ordner %bitfarm-archiv% und geben folgenden Befehl ein:

```
bfa_sqlmapper -c odbctest -s all
```

Mit dem Parameter `-s` all wird gesteuert, dass der Verbindungstest für alle in der Konfiguration gefundenen Sektionen ausgeführt wird, es ist aber auch möglich statt „all“ den Namen einer einzelnen zu testenden Sektion anzugeben. Sollte der Verbindungstest nicht erfolgreich sein, wird der dabei aufgetretene Fehler angezeigt. Als nächstes wird eine SQL-Query benötigt, die die gewünschten Informationen liefert. Der Wert, mit dem gesucht werden soll, stammt hierbei aus einem Zusatzfeld. Der Name des Zusatzfeldes wird in der Query in „%“ - Zeichen eingefasst, im obenstehenden Beispiel wird also mit dem Wert der im Zusatzfeld „Kundennummer“ steht, gesucht.
Im Weiteren wird das Mapping des Resultats der Query in die entsprechenden Zusatzfelder konfiguriert. Als Option wählt man dazu den Namen des Zusatzfeldes und als Wert verwendet man den in „%“ - Zeichen eingefassten Spalten-Index aus dem Ergebnis der Query. Im o.g. Beispiel wird das Resultat nur eine einzelne Spalte enthalten (die mit dem Kundennamen), daher wird hier „%0%“ (Index beginnt bei 0!) verwendet. Neben dem Index können als Wert auch feste Zeichenketten angegeben werden. Damit ist es möglich den Wert aus der Datenbank noch um weitere fixe Angaben zu ergänzen. Würde im o.g. Beispiel für den Kundennamen ---%0%--- angegeben, und die Query liefert als Kundennamen z.B. „bitfarm-Archiv GmbH“, würde der gespeicherte Zusatzfeldwert „---bitfarm-Archiv GmbH---„ lauten.

Die beiden Schalter `emptyonmissing` und `emptyonnoresult` steuern, ob die angegebenen Zielfelder im Falle eines leeren Suchwertes (`emptyonmissing`) bzw. im Falle eines leeren Results der Query (`emptyonnoresult`), geleert werden sollen. Standardmäßig sind die beiden Schalter auf `False` gesetzt.

Wenn das Zielfeld beim Start des Vorgangs bereits einen Wert enthält, wird dieser standardmäßig nicht überschrieben. Sollte dies doch gewünscht sein, kann eine Option `overwrite_<Feldname>=True` entsprechend aktiviert werden (im o.g. Beispiel also `overwrite_Kundenname=True`).
Damit ist eine minimal-Konfiguration für den SQL-Mapper vollständig. Im Abschnitt „erweiterte Konfiguration“ werden zusätzliche Konfigurationsschalter beschrieben, mit denen weitere spezielle Anwendungsfälle abgedeckt werden können.

### d) Anbindung des sql-mappers per wfd-Regel

Damit beim Archivieren eines Dokuments der Abgleich der Metadaten erfolgen kann, muss der sql-mapper noch in der wfd-Datei in einer `naming section` angebunden werden. Beachten Sie in diesem Fall, dass der Wert, mit dem in der externen Datenbank gesucht werden soll (in unserem Beispiel also die Kundennummer), dann natürlich zur Laufzeit der Archivierung bereits vorhanden sein muss, d.h., dass die Kundennummer bereits beim Import eingegeben oder per naming Regel von der Archivierung, ermittelt werden kann.

In der `naming section` können Sie Bedingungen für das Ausführen der Sektion angeben (z.B. `archivtabelle=Eingangsrechnungen` etc.), danach wird lediglich die Sektion aus der sql-mapper Konfiguration angegeben, die in diesem Kontext ausgeführt werden soll:

```
naming section crm-Kunden
archivtabelle=Kunden
sqlmapper=hole-Kundennamen
end section
```

Sollen mehrere Sektionen der sql-mapper Konfiguration verwendet werden, kann dies durch mehrfache sql-mapper Anweisungen innerhalb der naming section erfolgen.

Hinweis: Bei der Verarbeitung von wfd-Regeln werden sql-mapper Anweisungen erst nach Ausführen aller anderen sorting- und naming- Regeln angewandt, damit es möglich ist, den lookup-Wert ebenfalls per wfd auszulesen.
Bei aktiviertem scriptdebug in der `archivierung.vbs` wird der Aufruf des sql-mappers sowie das Ergebnis der Ausführung im Eventlog protokolliert. Grundsätzlich finden Sie die Logmeldungen natürlich auch im konfigurierten Logfile des sql-mappers.

### e) Anbindung des sql-mappers als Viewer-Plugin

Damit der sql-mapper auf dem Client ausgeführt werden kann, muss zunächst die ODBC-Verbindung wie oben beschrieben, auf dem betroffenen Client eingerichtet werden. Stellen Sie dann sicher, dass Sie auf dem Server im Ordner *%bitfarm-archiv%\viewer-files\plugins* die Datei `plugins.ini` anpassen (siehe Systemhandbuch Abschnitt „Plugin-Möglichkeiten“) Tragen Sie das Skript `bfa_sqlmapper.vbs` in die `plugins.ini` ein und verwenden als Parameter „jobfile“, „waitexec“, „refreshdocument“. Wie bei jedem anderen Viewer-Plugin, muss der Viewer nach der Anbindung des sql-mappers neu gestartet werden. Testen Sie, ob der sql-mapper korrekt eingerichtet ist, indem Sie das Plugin einmal ausführen. Sollte das Zielfeld nach Ausführung nicht den erwarteten Wert tragen, sollten Sie im sql-mapper-Log nach Hinweisen für die Fehlersuche schauen.

### f) Erweiterte Viewer-Plugin Konfiguration

Im Standardfall werden bei der Ausführung als Viewer-Plugin alle Sektionen aus der `bfa_sqlmapper.ini` ausgeführt. Sollte dies nicht gewünscht sein, sondern z.B. grundsätzlich nur eine bestimmte Sektion ausgewertet werden, kann dies durch Anpassung der `bfa_sqlmapper.vbs` erfolgen (ersetzen Sie dazu die Zeile section=“all“ z.B. durch section=“hole-Kundennamen“).

Alternativ dazu können in der `bfa_sqlmapper.ini` auch pro Sektion Bedingungen formuliert werden, die vom sql-mapper ausgewertet werden. Die Sektion wird nur dann ausgeführt, wenn alle angegebenen Bedingungen zutreffen. Soll z.B. unsere Sektion [hole-Kundennamen] nur dann ausgewertet werden, wenn es sich bei dem Dokument um eines aus dem Archiv „Kundenrechnungen“ handelt, kann in der `bfa_sqlmapper.ini` eine neue Sektion namens `[vp_cond_hole-Kundenname]` hinzugefügt werden. (vp_cond_<Name der Sektion>). Innerhalb dieser können nun die Bedingungen formuliert werden, in unserem Beispiel sähe dann folgendermaßen aus:

```
[vp_cond_hole-Kundennamen]
archive=Kundenrechnungen
```

Achtung: [vp_cond_<Name der Sektion>] wird nur ausgewertet, wenn der sql-mapper als Viewer-Plugin ausgeführt wird.

Neben dem Archiv können auch Zusatz- bzw. Statusfeldwerte als Bedingung für die Auswertung der Sektion angegeben werden. Zusatzfeld-Optionen beginnen dabei immer mit `add_` gefolgt vom Namen des Zusatzfeldes, Statusfelder beginnen mit `sts_` gefolgt vom Statusfeldnamen. Nach dem Gleichheitszeichen wird für Zusatzfelder nun zunächst ein Operator hinterlegt. (mögliche Ausprägungen: „=“ (gleich), „<“ (kleiner), „>“ (größer), „!=“ (ungleich), „~“ (enthält), „.>“ (beginnt mit), „.<“ (endet mit). Nach dem Operator wird immer ein Pipe gesetzt (|), der den Operator und den folgenden Vergleichswert voneinander trennt. In der Notation für Statusfeld-Bedingungen gibt es keinen Operator, hier kann mit `sts_<NamedesStatusfeldes>=True` oder `sts_<NamedesStatusfeldes>=False`, gearbeitet werden.

Soll in unserem Beispiel der sql-mapper als Viewer-Plugin nur dann die Sektion [hole-Kundennamen] ausführen, wenn

1. Das Dokument im Archiv Kundennrechnungen
2. die Kundennummer immer mit einer „1“ beginnt
3. der Status „gebucht! des Dokuments gesetzt ist.

sieht die vollständige Bedingungs-Sektion folgendermaßen aus.

```
[vp_cond_hole-Kundennamen]
archive=Kundenrechnungen
add_Kundennummer=.>|1
sts_gebucht=True
```

### g) sql-mapper als standalone-Tool

Unter Umständen macht es Sinn, den sql-mapper in bestimmten Zeitabständen zu starten und dabei den Metadatenabgleich für mehrere Dokumente im Hintergund laufen zu lassen. Bei der Ausführung als Viewer-Plugin, wird der sql-mapper auf die vom Benutzer ausgewählten Dokumente angewandt, bei der Anbindung über die wfd-Datei auf das aktuell zu archivierende Dokument. Um im standalone-Kontext festlegen zu können, für welche Dokumente der sql-mapper den Abgleich durchführen soll, legen Sie mit dem bitfarm-Benutzer, der verschlüsselt in die Konfiguration eingetragen wurde, ein Lesezeichen an, welches genau die gewünschten Dokumente liefert. In unserem Beispiel könnte ein Lesezeichen angelegt werden, welches alle Dokumente findet, bei denen der Kundenname noch leer ist. Der Name des Lesezeichens wird dann in der Konfigurations-Sektion als bookmark angegeben. Heißt das Lesezeichen „ohne-Kundenname“ tragen Sie in die Sektion `[hole-Kundenname]` also bookmark=hole-Kundenname ein:

```
[hole-Kundenname]
constring=dsn=kundendb-odbc
bookmark=ohne-Kundenname
sql=select Kundenname from kundendb where kundennummer=%Kundennummer%
Kundenname=%0%
overwrite_Kundenname=True
```

```
emptyonmissing=True
emptyonnoresult=True
```

Zur Ausführung des sql-mappers öffnen Sie eine Konsole auf dem bitfarm-Server im Ordner *%bitfarm-archiv%*. Setzen Sie danach folgenden Befehl ab:

```
bfa_sqlmapper.exe -c standalone -s hole-Kundenname
```

Mit dem Parameter `-c standalone` wird angegeben, dass der sql-mapper im Kontext standalone ausgeführt werden soll, mit dem Parameter `-s hole-Kundenname` wird die Sektion [hole-Kundenname] ausgewählt. Mögliche Probleme bei der Ausführung werden sowohl in der Konsole angezeigt, als auch in das Logfile geschrieben, welches im Abschnitt logging unter standalone konfiguriert wurde.

### h) Erweiterte Konfiguration

#### 1. indexlookup

In unserem Beispiel suchen wir mit der zum Dokument erfassten Kundennummer nach dem Namen des Kunden. Ggfs. können zu dem Dokument auch mehrere Kundennummern erfasst worden sein, die dann z.B. in einem Universalfeld semikolongetrennt hinterlegt sind. In diesem Fall muss in der Konfiguration des sql-mappers hinterlegt werden, mit welcher Kundennummer der lookup durchgeführt wird. Hierbei besteht die Möglichkeit den Index des zu verwendenden Wertes anzugeben (beginnend bei 0!). Erweitern Sie die entsprechende Sektion um den Eintrag „lookupindex=0“, um immer die erste Kundennummer aus dem Zusatzfeld zu verwenden. Mit „lookupindex=1“ würde immer die zweite angegebene Nummer verwendet etc. Sollen für den lookup alle Kundennummern verwendet werden, tragen Sie in die Sektion „lookupindex=all“ ein. In diesem Fall würde das Ergebnis der sql-Query mehrere Ergebniszeilen liefern. Daher müssen die zu befüllenden Zusatzfelder entweder vom Typ Universal, Auswahl oder Mehrfachauswahl sein, da nur diese Felder mehr als einen einzelnen Wert tragen können.

#### 2. inputcasting/outputcasting

Wenn bei der Ausführung des sql-mappers sql-Fehler angezeigt/protokolliert werden, kann dies an nicht kompatiblen Datentypen liegen. Wenn z.B. der lookup-Wert aus einem Universalfeld ermittelt wird, so ist dieser vom Typ unicode-String. Nun kann es aber sein, dass der lookup-Wert tatsächlich nur aus Ziffern besteht und in der externen Datenbank als Integer-Wert gespeichert ist. Wird das sql-Statement ausgeführt, resultiert das in einem Fehler, da der Datentyp inkompatibel ist. Der Zusatzfeldwert muss vor dem Ausführen der Query in einen Integer gewandelt werden, damit die Query ausgeführt werden kann. Hierzu kann in der Konfiguration ein inputcaster eingetragen werden. Alle casting-Funktionen werden in der Datei `bfa_sqlmapper_casting.py` deklariert. Im Auslieferungszustand befinden sich in diesem Skript bereits einige Funktionen, das Skript kann aber beliebig um individuelle casting-Funktionen erweitert werden. Jede casting-Funktion wird mit 2 Argumenten aufgerufen, erstens dem Wert aus dem bitfarm-Zusatzfeld und als zweites der Kontext, in dem

der sql-mapper ausgeführt wird. Die Funktion muss so geschrieben werden, dass sie den Zusatzfeldwert in den passenden Datentyp für die Zieldatenbank konvertiert und zurückgibt.

Für das angegebene Beispiel (Lookup-Wert muss von string nach integer gewandelt werden), kann die casting-Funktion „toint“ verwendet werden, welche in der Konfigurations-Sektion als „inputcasting=toint“ hinterlegt werden kann.

Werden mehrere lookup-Werte in der Query verwendet, so werden die casting-Funktionen semikolon-getrennt angegeben und zwar passend zur Reihenfolge der lookup-Werte aus dem sql-statement. Für Werte, die keiner Wandlung bedürfen kann als casting-Funktion „pipe“ verwendet werden, um den Wert unverändert in der Query zu verwenden.

Wenn die Query ohne Fehler ausgeführt werden kann und ein Ergebnis liefert, kann es sein, dass die Ergebnis-Werte der Query für die Übertragung in ein Bitfarm-Zusatzfeld nun ebenso eine Typ-Wandlung benötigen. Es ist denkbar, dass die Query z.B. einen Integer liefert und das Ergebnis in ein Universal-Feld geschrieben werden soll. Da der sql-mapper den Datentyp des Wertes aus der externen Datenbank kennt und auch den Zieldatentyp ermitteln kann, ist hier idR. keine Angabe einer casting-Funktion notwendig. Sollte die automatische Wandlung jedoch scheitern, besteht auch hier die Möglichkeit in der bfa_sqlmapper_casting.py eine Funktion zu schreiben, die die Wandlung korrekt vollzieht. Die Funktion wird dann in der Sektion als outputcasting hinterlegt. Wenn bei der Query mehr als eine einzelne Spalte abgefragt wird, ist es notwendig für jede Spalte eine casting-Funktion anzugeben (in der Reihenfolge der selektierten Spalten aus der Query). Sollen manche Spalten mit einer eigenen Funktion gewandelt werden, andere Spalten aber automatisch gewandelt werden, so kann für letztere als casting-Funktion „auto“ verwendet werden.

# 5. add_sync_values

## a. Anwendungsbereich

Sie wollen den Benutzern Werte in einem Auswahl-Zusatzfeld zur Verfügung stellen, diese Werte aber nicht manuell eingeben sondern die möglichen Auswahlwerte mit bspw. einer entsprechenden Tabelle Ihrer ERP-Datenbank abgleichen.

## b. Anwendungsbeispiele

Häufig sind es Lieferantennamen, Kostenstellen, die Namen ihrer Rechnungsprüfer, Bauleiter… etc. die den Benutzern zur Auswahl gestellt werden sollen und bereits in einer Ihrer Datenbanken zur Verfügung stehen.

## c. Einstellungen/Konfiguration

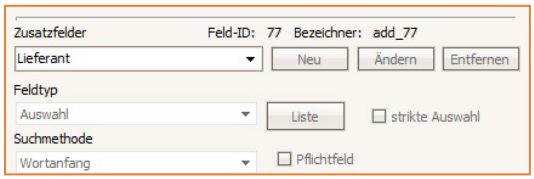
Die Konfiguration erfolgt wie beim bfa_sqlmapper per ini-Datei (standardmäßig in *%bitfarm-archiv%\ Bitfarm-Tools\add_sync_values\ add_sync_values.ini*). Beim Bearbeiten der Konfigurationsdatei muss darauf geachtet werden, dass ein geeigneter Editor verwendet wird, um das Encoding der Datei (UTF-8) nicht zu verändern. So ist z.B. Notepad dafür

bekannt, das Enconding beim Speichern zu verändern. Wir empfehlen den Einsatz eines modernen Text-Editors wie z.B. Notepad++.

Wie bei der Konfiguration des *bfa_sqlmappers*, werden auch hier die Anmeldedaten mit dem Kommandozeilen-Tool credcrypt.exe, welches sich auf dem bitfarm Server im Verzeichnis %bitfarm-archiv% befindet, verschlüsselt. Melden Sie sich auf dem Server mit dem Benutzer an, unter dem die bitfarm-Dienste laufen und öffnen eine normale Konsole im Ordner *%bitfarm-archiv%.* Geben Sie `credcrypt.exe` ein, gefolgt vom Benutzernamen, mit dem sich das Tool `add_sync_values.exe` anmelden soll (z.B. *DMSAdmin*). Der verschlüsselte Benutzername wird auf der Konsole ausgegeben und zusätzlich auch in der Zwischenablage gespeichert, übertragen Sie ihn in die Konfiguration hinter `user=`. Auf die gleiche Weise erzeugen Sie nun das verschlüsselte Passwort des Benutzers und tragen es hinter `pass=` ein. Tragen Sie hinter `confile=` den Pfad auf die Con-Datei ein.

```
[main]
user=AQAAANCMnd8BFdERjHoAwE/Cl+…
pass=AQAAANCMnd8BFdERjHoAwE/Cl+…
confile=c:\bitfarm-archiv\Muster-GmbH.con

[add_77]
src=odbc
constring=DSN=lfodbc
sql=SELECT DISTINCT lfname from lieferanten
action=add_values
;action=add_values → alte Werte bleiben erhalten, neue werden hinzugefügt
;action=replace_values → alte Werte werden entfernt, alle ermittelten hinzugefügt
;action=remove_values → entfernt die ermittelten Werte aus der Auswahl
```




Das im bitfarm-Archiv DMS zu befüllende Zusaztfeld wird in eckigen Klammern, in o.g. Notation angegeben. Welche ID das Zusatzfeld hat, welches Sie befüllen lassen möchten, können Sie u.a. im Administrator herausfinden, wenn Sie sich im Viewer als *dmsadmin* anmelden.

Geben Sie hinter `constring=DSN=` den Namen Ihrer eingerichteten 32-bit-ODBC Verbindung ein. Ggf. müssen Sie hier für Ihre spezifische Datenbank einen connection-string hinterlegen. Wie dieser aussehen muss, hängt vom Typ/Treiber der Datenquelle ab, ggfs. ist diese Information beim Anbieter zu erfragen. In vielen Konstellationen reicht die Angabe des dsn in der Form: constring=dsn=<Data Source Name>. Weitere Informationen liefert z.B. die Seite https://www.connectionstrings.com

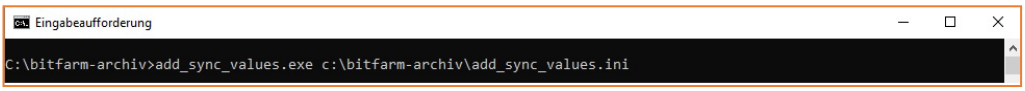
Tragen Sie bei `sql=` den SELECT ein, der auf der abzufragenden DB ausgeführt werden muss, um die Werte zu erhalten, die in das Auswahl-Zusatzfeld übertragen werden sollen.

Mit `action=` legen Sie wie o.g. fest, wie die Werte hinzugefügt, bzw. entfernt werden.

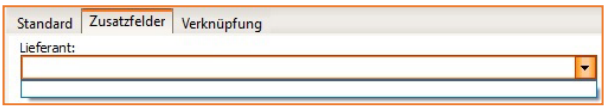
Hinweis: Für den Fall, dass Sie keine ODBC-Verbindung nutzen können, haben Sie auch die Möglichkeit eine .jason-Datei mit den zu importierenden Werten erzeugen zu lassen, den Output entweder auf stdout auszulesen, oder in einem .jason-File zu speichern und diesen dann auszulesen. In der *add_snc_values.ini* ist für diesen Fall eine entsprechende Beispielsektion angelegt.

### d. Ausführung

Binden Sie die Datei `add.sync.values.exe` über die Windows-Aufgabenplanung an und tragen Sie bei Argumente den Pfad auf die ini-Datei ein, um einen regelmäßigen Abgleich zwischen Ihrer Datenbanktabelle und dem Zusatzfeld in bitfarm durchzuführen. Wollen Sie diesen Abgleich einmalig ausführen können Sie dies auch über die Windows-Eingabeaufforderung machen.




Nachdem der Abgleich auf einem dieser Wege durchgeführt wurde, können Sie Ihr Ergebnis im entsprechenden Zusatzfeld im bitfarm-Archiv Viewer betrachten.




Das Auswahl-Zusatzfeld Lieferant (add_77 aus dem Beispiel) vor dem Abgleich.


Das Auswahl-Zusatzfeld Lieferant (add_77 aus dem Beispiel) nach dem Abgleich

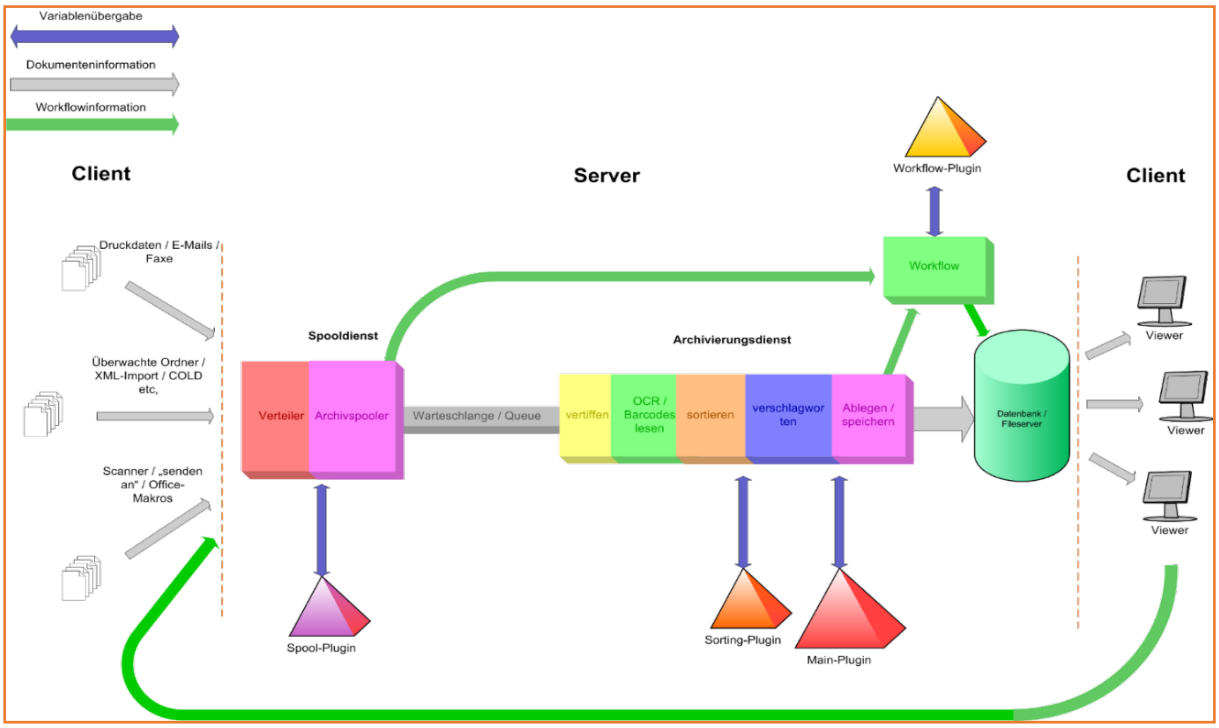
# IV. Individuelle Steuerungen mit Hilfe von Plugins

## 6. Serverplugins

An unterschiedlichen Stationen des Durchlaufes eines Dokumentes durch den Archivierungs- bzw. Workflowprozesses stehen Möglichkeiten zur Verfügung, mit individuell angepassten Programmen (Plugins) Einfluss auf die Prozesse nehmen, Variablen und Feldinhalte zu verändern und/oder bestimmte Aktionen auszulösen. Plugins können beliebige ausführbare Programme sein. Der Informationsaustausch mit dem bitfarm-Archiv Server erfolgt über eine Datei, welche als Parameter beim Start übergeben wird. Diese Datei ist eine Text-Datei, welche in Form von `Variablenname=Wert` alle zur Verfügung stehenden Variablen übergibt. Die Textdatei wird nach Beenden des Plugins wieder in den Server zurückgelesen.

Für die einzelnen Plugins liegen Beispiele unter ..*bitfarm-archiv**plugins*\ in Visual Basic Script bereit.

Die folgende Darstellung zeigt die möglichen Plugins und deren Position im zeitlichen Verlauf den Dokumentenverarbeitung.




## 7. Viewer-Plugins

Viewer-Plugins werden in ...*bitfarm-archiv**Viewer-Files**Plugins* abgelegt und dort in die `Plugins.ini` eigetragen. In der Sektion `[Conf]` wird die Anzahl der Plugins eingetragen, unter `[Name]` die Dateinamen und unter `[Params]` die Optionen. Im Viewer werden die Plugins mit dem Basisnamen (Dateiname ohne Endung) angezeigt.

Wird ein Plugin im Viewer gestartet, so wird die Plugindatei ausgeführt und bekommt den Pfad auf die temporäre Textdatei übergeben. In dieser ist der Pfad auf die zum Dokument gehörende Jobdatei und ggf. Voransichts-Tifdatei eingetragen. Diese Dateien werden für das Plugin im Verzeichnis *%temp%\Viewer* bzw. *%temp%\Viewer\tif* zur Verfügung gestellt.

Viewer-Plugins werden, entweder manuell per Rechtsklick oder über den entsprechenden Button in der Viewerobefläche, gestartet. Möchte man Viewer-Plugins über „Knöpfe\Buttons“ im Viewer starten, so muss man diese erst einblenden. Dies kann man nach Wunsch in den Optionen des Viewers konfigurieren.

Die möglichen Optionen für ein Plugin:

| | |
|---|---|
| `jobfile` | Eine Jobdatei wird exportiert |
| `tiffile` | Das Voransichtstiff wird exportiert |
| `waitexec` | Der Viewer wartet bis das Plugin beendet wurde |
| `refreshlist` | Die Trefferliste wird aktualisiert (Suche erneut ausgeführt) (auch: refreshliste) |
| `refreshresub` | Die Wiedervorlage wird aktualisiert |
| `resync` | Die Jobdatei wird importiert, das Dokument wird mit den eingelesenen Werten aktualisiert (Speichern- oder Verschiebe-Plugins) |

Beispiel: `Plugins.ini`

```
[Conf]
counter=2
[Names]
file0=showdocname.vbs
file1=MoveDoc.exe
[Params]
file0=jobfile, waitexec, refreshdocument
file1=jobfile, tiffile
```

Für ein Speichern oder Verschiebe-Plugin muss unter `[Conf]` noch der Basisname des Plugins angegeben und außerdem im bitfarm-Archiv Administrator, für die gewünschten Archive noch der Haken „Plugin aktiv“ gesetzt werden.

Die Plugins werden dann in diesen Archiven bei jedem Speichern oder Verschiebevorgang ausgeführt. Das Speichern-Plugin wird am Ende des gesamten Speichern-Prozesses ausgeführt, es stehen also schon neu eingegebene Zusatzfeldwerte oder Statusfelder zur Verfügung. Das Verschiebe-Plugin wird unmittelbar vor dem Verschieben ausgeführt, ein returncode = 255 bricht den Verschiebevorgang ab. Aus Performancegründen steht beim Speichern oder Verschiebeplugin nur die Jobdatei zur Verfügung.

```
[conf]
counter=2
moveplugin=MoveDoc
saveplugin=showdocname
```

Für Plugins, die mit der Option `waitexec` oder `resync` ausgeführt werden, kann man in der `con`-Datei unter `[Optionen]` einen `plugintimeout=10` (in Sekunden) angeben.

## 8. Clientsteuerung durch führende Applikationen

Für die Anzeige und den Umgang mit Dokumenten ist der bitfam-Archiv-Viewer das zentrale Clientprogramm des DMS. Die Funktionalität geht weit über eine einfache Darstellung hinaus und ermöglicht den Umgang mit elektronischen Notizen, Datenbankinformationen, Workflows und Verteilung, Recherchen und vieles mehr. Der Viewer ermöglicht also nicht nur den Zugriff auf das eigentliche Dokument, sondern stellt auch alle Zusatzinformationen und Zusatzfunktionen bereit. Bei der Anbindung des DMS an andere Systeme ist es daher sinnvoll, zur Dokumentendarstellung nicht einen einfachen TIF-Viewer anzusteuern, sondern den

bitfarm-Archiv Client direkt zu verwenden. Die hier zur Verfügung stehenden Schnittstellen werden im folgenden beschrieben.

Tipp: In vielen Fällen ist zur Dokumentendarstellung ein zweiter Monitor sinnvoll. Dieser Betriebsmodus wird daher auch vom bitfarm-Archiv Viewer unterstützt.

## 9. Hotsearch-Funktionen

Das Hotsearch-Tool besteht aus einer Datei `Hotsearch.exe` welche auf dem Client-Systemen über den Autostart gestartet werden sollte. Über die Tastenkombination *Strg-C* und anschließend *Strg-B*, kann Text aus beliebigen Windows-Applikationen als Suchtext in den Viewer geschoben werden, der daraufhin gleich eine Volltextsuche in dem zuletzt angewählten Archiv oder Lagerort ausgeführt. Wichtig ist lediglich, dass die Windows-Applikation das Übertagen von Information per *Strg-C* zulässt. So kann beispielsweise der Anwender in seiner ERP-Anwendung den Inhalt des Feldes „Rechnungsnummer“ per Doppelklick markieren und hat dann nach *Strg-C Strg-B* die Rechnung gleich auf dem Bildschirm.

## 10. Direktes Anspringen eines Dokumentes über die GDocID

Dokumente in der DMS-Datenbank haben die sog. *GDocID* als globalen Primäschlüssel. Die *GDocID* wird auch beim Export der Metadaten mitgeliefert, somit gibt es hier eine Möglichkeit für externe Systeme, eine gezielte Referenz auf ein Dokumentzu speichern. Der Aufruf erfolgt dann einfach als einziger Parameter über die Kommandozeile.

```
\\DMSServer\bitfarm-archiv$\viewerv3.exe gdocid:531
```

Ist der Viewer bereits geöffnet, so wird hier die Darstellung entsprechend aktualisiert. Ist noch kein Viewer geöffnet, erfolgt es dann mit Aufforderung zur Anmeldung am System und anschließender Darstellung des Dokuments.

Hinweis: Um ein Dokument über den externen Aufruf der Gdocid anzeigen zu können, benötigt der Benutzer in dem entsprechenden Archiv das Suchen-Recht.

## 11. Programmatische Übergabe von Suchbegriffen

Mit anderen Parametern kann eine Schlagwortsuche bzw. Volltextsuche in einem bestimmten Archiv angestoßen werden:

```
Viewerv3.exe -L:[Lager] -lid:[ID des gewünschten Lagers] -A:[Archiv]
-aid:[ID des gewünschten Archivs] -S:[Schlagworte] -V:[Volltext]
-R:[Verknüpfung] -GO -W: -C: -G:[gdocid] docid:[arc_id.doc_id]
gdocid:[gdoc_id] [arc_id].[doc_id]
-ZUS:[Zusatzfeldname]:[Zusatzfeldwert]
```

Beispiel: Suche im Viewer in einem bestimmten Lager und Archiv nach Schlagwort 4711:

```
viewerv3.exe -L:Einkauf -A:Lieferscheine -S:4711 -GO
```

Beispiel: Suche im Viewer nach Zusatzfeld-Titel mit den Wert „Dokument“ mit Wiedervorlage offen

```
viewerv3.exe -W: -ZUS:Titel:Dokument
```

Tipp: Der parametriesierte Aufruf kann auch über die hotsearch.exe durchgeführt werden. Da diese kleiner ist, ist der Aufruf schneller.

Die Parameter im Einzelnen

`-L:[Lager]` wechselt in das angegebene Lager
`-lid:[Unique ID des Lagers]`
`-A:[Archiv]` wechselt in das angegebene Archiv. Das Archiv muss sich unterhalb des aktuellen Lagerortes befinden, wenn ein Aufruf nur mit -A: erfolgt. Soll in ein Archiv unterhalb eines anderen Lagerortes gewechselt werden, so ist dieses vorher mit -L: anzugeben.

`-aid:[Unique ID des Archivs]`
`-S:[Schlagwort]` füllt das Schlagwortfeld für eine Suche
`-V:[Volltext]` füllt die Volltextsuche mit ein oder mehreren Begriffen
`-R:[Verknüpfung]` füllt das Verknüpfungsfeld, die Suche über Verknüpfung kann nicht mit Volltextsuche oder Schlagwortsuche kombiniert werden.
`-GO` führt die Suche über Schlagwort, Volltext oder Verknüpfung durch.
`-W:` öffnet die Wiedervorlagenliste
`-P:` Druckt das aktuell angezeigte Dokument
`-C:` Schließt den Viewer
`-G:[GocID]` Öffnet das angegebene Dokument über die GdocID
oder `gdocid:[GdocID]`
`-ZUS:[Zusatzfeldname]:[Zusatzfeldwert]`
Führt eine archivweite Suche in dem Zusatzfeld durch, es werden also alle Archive durchsucht, in denen das Feld vorkommt. Es wird nach Wortbestandteil (nicht exakt) gesucht und es können keine Bereichssuchen durchgeführt werden. Wird ein Lager oder ein Archiv vorher angegeben (mit -L: oder -A:) dann wird die Suche auf dieses Lager oder Archiv eingeschränkt. Möchte man Zusatzfelder suchen, die einen anderen Datentyp haben als Universalfelder z.B. Datums-, Fließkommazahl- oder Währungsfelder, dann muss der Suchbegriff so formatiert sein, wie der Datentyp in den Datenbanktabellen abgelegt ist, d.h. bei einer Datumssuche muss das Datum im MySQL-Datumsformat angegeben werden: (YYYY-MM-DD), z.B. -ZUS:Datum:2019-10-12
Bei Zahlen mit Nachkommestellen muss das Komma durch einen Punkt ersetzt werden, z.B. -ZUS:Betrag:219.77

### a. Bei archivseitig entzogener Suchberechtigung

Die zuvor beschriebene *programmatische Übergabe von Suchbegriffen* setzt immer voraus, dass die Benutzer, die diesen Aufruf machen, auch eine Such-Berechtigung auf dem Archiv haben, indem das Dokument liegt.

In der Praxis ist es jedoch immer wieder so, dass die Benutzer in einem Archiv nur bestimmte Dokumente sehen dürfen.

Beispiel: Lischen Müller darf im *Archiv* „Kreditoren" nur die Rechnungen sehen, dir Ihr zugewiesen sind, bspw. nur die Rechnungen, bei denen ihr Name in einem *Zusatzfeld* namens „Prüfer" steht. Im DMS bildet man dies ab, indem man ihr auf dem *Archiv* „Kreditoren" die Berechtigung des Suchens entzieht, dafür aber ein so genanntens *globales Lesezeichen* für sie anlegt. Dies ist letztendlich eine gespeicherte SQL-Suchabfrage (beschrieben im Systemhandbuch). Diese Art der Suchabfrage(n) kann vom Administrator des DMS für die entsprechenden Benutzer zur Verfügung gestellt werden.

Über diese *globalen Lesezeichen* kann den jeweiligen Benutzern eine Menge von Dokumenten zur Verfügung gestellt werden, auf welche sie dann eine aufbauende Suche ausführen dürfen.

Hinweis: Man kann den externen Aufruf eines Lesezeichens auch mit dem Aufruf eines Lesezeichen-XMLs oder Strings oder einer *gdocid*-Suche kombinieren. So kann man aufbauend auf das globale Lesezeichen, gezielte Suchen von extern ausführen.

Die Parameter im Einzelnen

`-gs:<Name eines globalen Lesezeichens>` führt ein *globales Lesezeichen* aus, welches beim angemeldeten Benutzer vorhanden sein muss und per Definition die eingeschränkte Ergebnismenge liefert, die dieser Benutzer sehen darf.

Hinweis: Der Benutzer sollte kein weiteres *globales Lesezeichen* mit gleichem Namen haben.

aufbauend darauf kann mit

`-so:<Name einer XML Datei>`

`-so:<Name eines XML Lesezeichenstring>`

`-so:<Name eines lokalen Lesezeichens>`

oder mit dem Parameter

`gdocid:ID`

die Suche kombiniert werden.

Beispiel: `ViewerV3.exe -gs:Meine Rechnungen gdocid:4711`

➔ Zeigt das Dokument mit der *gdocid* 4711 an, auch wenn der Benutzer keine Suchberechtigung auf entsprechendem Archiv hat, sofern das Dokument über das globale Lesezeichen erreichbar ist. So wird zum einen das Berechtigungskonzept im DMS aufrechterhalten und andererseits auch „Suchen“ von außerhalb ermöglicht.

Hinweis: Das für diesen Zweck angelegte globale Lesezeichen darf keine „Leersuche“ in dem jeweiligen Archiv sein.

Hinweis: Bei Verwendung des `-so:` Parameters kann man vollständige Lesezeichen XML-Dateien verwenden. Diese werden aufbauend auf das globale Lesezeichen ausgeführt. So sind auch Suchen nach weiteren Kriterien, wie Zusatz- und/oder Statusfeldern möglich. Im Lesezeicheneditor kann man durch Klick auf das Infosymbol das Lesezeichen XML anzeigen lassen und auch speichern.

Man kann aber auch direkt einen Lesezeichen-String mitgeben. Dieser muss valides XML mit einem `<bookmark>` und einem darunterliegenden `<fav>` Node sein. Da in der kombinierten Suche mit einem globalen Leszeichen nur der SQL Where String des Lesezeichens verwendet wird, würde so ein XML-String in der Minimalfassung aussehen:

Beispiel:
```
<?xmlversion='1.0'?><bookmark><fav><sqlwhere>(sd.gdocid=4711)
</sqlwhere></fav> </bookmark>
```

Im Beispiel wird nach der *gdocid* 4711 gesucht

Hinweis: Innerhalb der XML müssen einfache Anführungszeichen verwendet werden.

Beispiel:
```
-so:<?xmlversion='1.0'?><bookmark><fav><sqlwhere>
(add_2='Suchtext')</sqlwhere></fav></bookmark>
```

## 12. Erweiterte Suche mit einer .FND-Datei

Mit dem bitfarm-Archiv Administrator, kann für jedes Archiv eine „Suchvorlage“ erstellt werden. Diese Datei mit der Endung .fnd ist ein leeres Gerüst ähnlich einer .tpl-Datei, welches mit Suchinformationen gefüllt werden kann und im Viewer dann als Parameter übergeben wird. So lassen sich auch von außen gesteuerte Suchen über bestimmte Zusatzfelder realisieren.

# V. Kontakt bitfarm-Archiv Softwaresupport

Bei Fragen oder Problemen, wenden Sie sich an den bitfarm-Softwaresupport

Telefon: +49 (271) 31396-0

E-Mail: support@bitfarm-archiv.de

Telefonische Erreichbarkeit des Supports:

Mo. – Do. 8:00 Uhr bis 17:00 Uhr

Fr. 8:00 bis 15:00 Uhr

Bitte halten Sie Ihre Vertrags- oder Partner-Nummer bereit.