bitfarm® ARCHIV
Dokumentenmanagement

Interfaces / Data Transfer / Integration possibilities

Enterprise-Version 3.6.1

Table of Contents

I. Document import

1. Drop in watched folders

In the configuration-file `scripts.ini`, in the bitfarm-Archiv program directory on the server, you can enter the path to a folder under

`ScannerImportPath=`
and/or
`ExtendedImport=`
that is watched by the bitfarm-Archiv *Spool-client*. Files that are moved here are put into the archive-queue, indexed by the *archiving client,* sorted according to the settings, supplemented with keywords, and then archived. You can add further steps (like *workflows*). If there are no sorting-rules, nor any plugins at work, documents are archived in 'undistributed'.

You can create sub-folders inside the watched folders, in which files can be archived directly. Those need to have the same name as the templates of the archives. If documents are placed in these sub-folders, the software will search for a template of the same name *%bitfarm-archiv%**templates*-folder. If there is one, it will move the document to the archive that is set in the template.

The target-archive (or user) can also be determined via file names.

The most common usable files types are: `TIF, .PDF, .JPG, .GIF, .BMP, .DOC, .XLS, .PPT, .MSG, .EML, .DWG, .PLT, .DXF, .RTF, .HTM, .HTML, .ASC, .TXT`

2. Data transfer via .job-file

A similar process to the data transfer with XML-files is via .job files, which bitfarm-Archiv also uses for internal information transfer. To do this, every document that you want to archive needs to have .job file of the same name. You can import the document by moving the job file and then the document to the main transfer folder. This folder is called Clientspooler or UNC-Clientspooler in the profile-file.

The .job file is created by a template-file, which is generated automatically by the DMS when creating the archive. It functions as a framework and already shows which fields in an archive are valid. The template then needs to be copied, filled and renamed to .job by the imported application. Here is an example for one such template-file, which you can find in the program directory *%bitfarm-archiv%* on the server in the sub-folder templates (Archivname.tpl):

```
[Version]
Version=36
```

```
[Archiv]
Profil=template-gmbh
Name=creditors
Tabelle=28052015172108
Arcid=10
[Document]
DOCID=
gdoc_id=
Titel=                          (title)
Orginal=
Kopie=
Quelle=
Benutzer=                       (archiving user)
userid=
Volltext=
Status=
StatStr=
OCR_QF=
OCR_Typ=OMP
Schlagworte=                    (tag list)
Datum=                          (date of document)
Filter=
Pages=
[Hash]
SHA1=
[Referenz]                      (references)
Anzahl=0
[SVN]
SVN_Link=
SVN_Version=
SVN_Revision=
SVN_Datum=
SVN_Info=
[Tasks]
Termin=                         (resubmission date, task date)
Alarm=                          (deadline date)
Bearbeiter=                     (user, person in charge)
Notiz=                          (note, hint on the task to do)
TimerOptionen=0
[Zusatzfelder]                  (additional fields)
Felder=17
ZusTitel1=Ordnernumber
ZusFeld1=add_5
ZusWert1=                       (i.e. value for „Ordernumber“)
ZusTitel2=Invoicenumber
ZusFeld2=add_3
```

```
ZusWert2=
ZusTitel3=Invoicedate
ZusFeld3=add_2
ZusWert3=
ZusTitel4=Invoiceamount
ZusFeld4=add_14
ZusWert4=
ZusTitel5=Auditor
…
```

The following fields can be filled by the importing system:

- date
- user
- editor
- title
- references
- additional fields
- deadline
- reminder
- note

It can also deliver a .txt file with the same name, which is used for the full-text search. For this you need to enter TXT in the OCR-type to disable the OCR.

You can also add a .slw file. Its content is saved to the global keywords.

3. Printing into the archive

You can set up virtual printers for the archiving function on the client (refer to the related chapter in the system manual). When setting up the printer you can select the target archive and a real printer (which has to be installed on the server) for additional paper output.

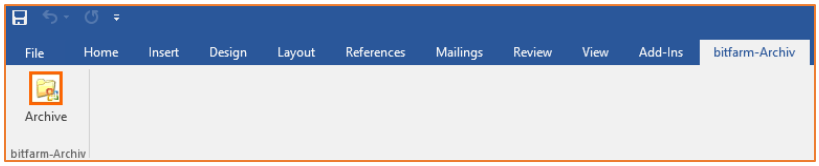
Printing-files can be read and indexed accurately and then be used in the .wfd file for keywording. On disadvantage this process has to the other two procedures described above is that it can only send information to the DMS, which are present on the document itself, or can be deducted from its structure (see plugins).

4. Importing ASCII-data (COLD)

For systems (mostly UNIX-host based ERP-applications), which can neither export PDF/TIF/DOC nor print on Windows printers, you can import printing-data via COLD. To do this you need to have the data for the printer written into a file and copied into one of the previously mentioned watched folders. You can leave out the file extension or enter ASC. At the moment, an Epson

printer-emulation is being programmed into it. For more commands, consult the bitfarm-Archiv software-support.

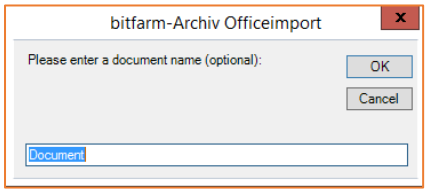
5. MS-Office add-ins



For the Microsoft-Office applications (up until version 2019) Word, Excel, and Outlook you can make use of add-ins. With these you can quickly import files from these applications to the DMS. These add-ins are automatically installed together with the client, or can be installed manually later. For the manual installation, open the file `bfaOfficeSetup.msi` on your DMS-server.

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

Select the archiving add-in in Word or Excel.



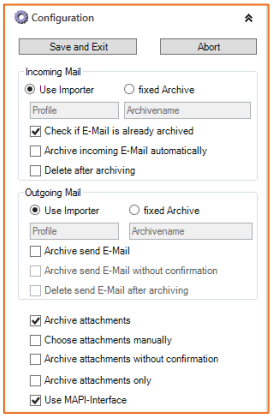
If the document is still untitled, you can name it before importing it to the DMS, or you can change it afterwards.

Attention: In the bitfarm-Archiv DMS you can search documents via name.

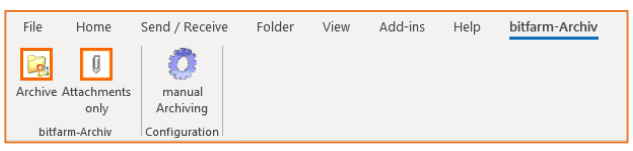
With Outlook you also have the option to only archive the attached files, instead of the entire mail. Click on *manually archive* to set it up accordingly.

Automatically archiving emails is not recommended in user-mailboxes, because you cannot usually predict which type of document it is.

Hint: you can archive mailboxes like: invoice@yourcompany.com by using the bfaPop3Connector as laid out in the following chapter.



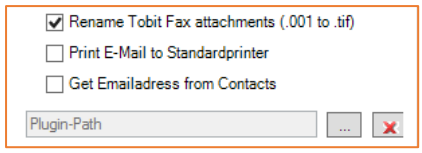
When you activate the option 'manually select attachments', another window will open, in which you can see and select which attachments are sent.



Example: In the configuration of the add-in you can also choose to import into a predetermined archive. Use the option 'fixed archive' and then select the archive.

6. Automatic mail-import (POP3)

You can automatically enter and archive important mail accounts using the bitfarm e-mail client together with the bfaPop3-Connector.



Attention: We do not recommend archiving all of your company's mails this way. You should use it for specific mailboxes such as invoice@mycompany.com or application@mycompany.com, there it is clear what kind of document you are dealing with and in which archive they should be saved.

POP3-service as a client

Attention: If the POP3 service is running in a client-context, it will always clear the mailbox once the mails have been exported to bitfarm.

First, create a folder *bfaEMail* in the *bitfarm-Archiv*-folder and add all files from the folder *...\bitfarm-Archiv\install\Bitfarm-Tools\bfaEMail*.

Open `bfapop3.ini` and adjust all entries:

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

With the command `POP3 Server` you can specify the name or IP-adress of the POP3 server. Under `Username` you enter the name of the account. With the command `transfer` you can select in which archive the documents should be imported. Remember that the *import* is watched by the spool-client, because e-mails will not be imported into the DMS otherwise. With `UseTLS` you can decide if the login needs to happen via TLS. Possible values are `true` or `false`. The command `TLSUser` determines the (Windows-) user of the POP3 account. With a custom port (default is 110) you can add the command `Port=`. What follows defines the used port.

Use the admin console to run the command `bfaEMailService.exe -install` and the POP3 service is installed. In the client-context you will also need to assign a registration to it via bitfarm-User.

When starting the client, it will start to check the configured post box every five minutes and sends all found mails to the path defined under `transfer`. Additionally you can add the starting-parameter `-t:<sec>` and change the pick-up interval.

Attention: In the client-context the POP3 generates service-logs which are saved in …*bitfarm-Archiv\bfaEMail\logs-folder*.

POP3-service as console command

The POP3-service can also be started as a console command. You can view all possible parameters with the command `bfapop3.exe -h`:

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

The meaning of each parameter is:

`POP3-Server`	Name or IP of the POP3 server
`-auto`	Parameter for the automated pick-up. Uses the connection set in bfapop3.ini.
`User`	User of the POP3 account
`Pass`	Password of the POP 3 account
`transfer-path`	Folder where emails are saved.

Optional:

`-TLS`	Activating the login via TLS
`-TLSUser`	(Windows-)Username of the POP3 account
`-p`	Port (default is 110)
`-nodelete`	Emails in the mailbox will not be deleted
`-log`	Writes a log in a separate sub-folder

Example:

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

You can retrieve from multiple POP3 accounts at the same time using a `BAT`-file.

7. Archiving SAP-documents via the bitfarm Content-Server interface (bfaSAP)

a) Description

The bitfarm-Archiv SAP-interface is a so-called Content-Server (CS) according to the interface-description ‘SAP Content Server http 4.5‘. It was fully implemented according to the interface-description, which you can find here:

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

The interface is mostly a web-service (web-service client) that can send to, or receive documents from SAP via the HTTP-protocol. For this SAP sends HTTP-requests to the Content-Server to send or request documents.

Only common international standards may be used when using this interface, such as: HTTP-protocol 1.1 (RFC 2068), HTML (HyperText-Markup-Language), URL (Uniform Resource Locators, RFC 2396), UTC (Universal Time Coordinated), and digital encryption methods according to the Public/Private-Key principle with PKCS#7 signatures (RFC 2315)

In SAP so-called Repositories are created to which different document types can be assigned. They will also be assigned to a network address and a port. Via this address a Content-Server is defined for SAP, in this case the bitfarm Archiv Content-Server (bfaSAP). In the bitfarm Content-Server the repositories are assigned to archives in the DMS.

Optionally, the communication via the interfaces can be digitally signed. SAP signs all requests to the Content-Server. The Content-Server can use a public key, provided by SAP, to check if the requests are legit.

b) Archiving and Provision of documents

If a document is generated in SAP, it will send a HTTP-Request to the CS-interface a URL. Included in this URL are the command as well as the necessary parameters and optionally a signature. If a document should be created, it is a create-request with information on the repository, the document-ID of the SAP, access authorization etc. The actual document can be found in the so-called body of the http request, usually as a PDF-file. The Content-Server will then save the file in the tamper-proof section of the archive and create a database-entry (document) in the bitfarm-database. At the same time the document-ID of the SAP is entered into a separate table (sys_sap). Finally, the Content-Server sends a HTTP-return or an error code back to SAP.

If SAP requests the document, it sends a get-request with the ID of the requested document. Using the SAP-ID the document is located in the bitfarm database and sent in the body of the server-answer to SAP.

Additionally, the CS can update, complete, search, and delete documents. It can also give administrative commands to request information on documents or Content-Servers, as well as sending the certificates for repositories.

c) Tamper-proofing

The bitfarm Content-Server runs on the bitfarm Archiv server as a Windows-client. This client works as a generic user account, which has the same access as a bitfarm-Archiv client-user and is only created for bitfarm clients. This client-user is the only one who can edit inside the tamper-proof section of the archive.

The tamper-proof section is a folder structure that is managed by bitfarm-Archiv.

If SAP sends a document via create-request to the Content-Server, it will save the document (the body of the http-request) directly to the tamper-proof section. Once it is saved in this section, it is impossible to edit it for anyone except the client-user.

This document/file will remain at this location, even if it is relocated to another folder within the DMS, since that only happens within the bitfarm database.

If SAP sends a delete-request, the document will only be marked as 'deleted' in the database. The file itself still remains.

Using the history-spreadsheet in the bitfarm-Archiv DMS, you have a complete overview of what happened to the document from the moment it has been archived.

d) Technical requirements

The bitfarm-Archiv Content-Server runs on Windows-Servers, and was tested with the versions Server2003, Server2008 R2 and Server2016. In principle, it will run on any platform that can execute a Python interpreter.

The Content-Server is based on version 2.7.2 / 2.7.3 of the Python programming language.

The web-server uses the Python-framework CherryPy 3.2, which also needs to be installed.

To check signatures with the SAP-certificates, you also need OpenSSL version 1.0.1L.

e) Configuration

The bitfarm-Archiv Content-Server requires two configuration-files: `bfaSAP.conf` which the web-server needs for the network-address, as well as the port and `bfaSAP.ini`, for general settings and with which the repositories are assigned to the target-archives in the DMS. Every SAP-repository is identified by two letters. The related section can also be found in the `ini`-file, where the assigned archive can be found.

Attention: The statements `arcid`, `tblname`, `arcname` can be found in the table `sys_arc` in the bitfarm database, or as generated archive-templates in *%bitfarm-archiv%\templates*

Example

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

Beispiel: bfaSAP.conf

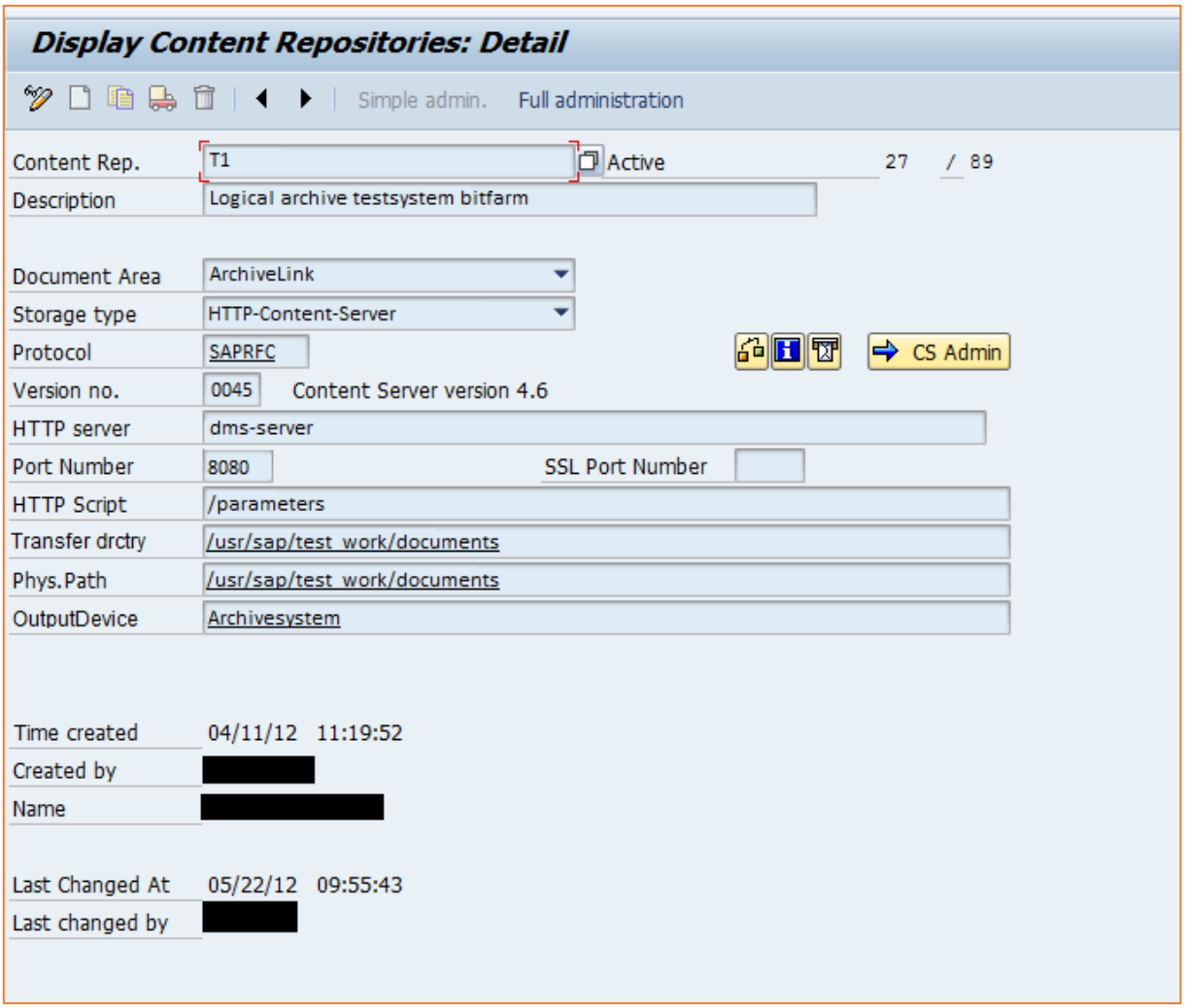
```
[SAP]
; mehrere Repositorys mit ; getrennt angeben
repository=T1;T2
server=dms-server
signed=false
[bitfarm]
profil=demo
changeable=.doc;.xls
openssl=c:\OpenSSL\bin\openssl.exe
[T1]
arcid=68
tblname=09032010135935
arcname=Rechnungen_Gutschriften_Vertrieb
description=Rechnungen_Gutschriften_Vertrieb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

Beispiel: bfaSAP.ini

On the SAP page, you now have to add the bitfarm Content-Server to the configuration of the SAP-repositories. To do this you need to either create a new repository, or change the ones of which the document types should be saved in bitfarm-Archiv.

(SAP transaction OAC0)

Relevant are only the names of the http-servers (name of the bitfarm-Servers), the port number



and the http-script (/parameters).

f) Further Information

Interface description 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/

The procedural documentation of the bitfarm-Archiv DMS can be found in the documentations-folder in the bitfarm program directory.

8. Transferring data in the XML-format, bitfarm XML-Importer

a) General

With the Bitfarm-XML-Importer you can import documents, which have metadata in an XML-file. The import can be started via the following request:

```
bfa_xmlimport.exe <path to the configuration file> <xml-file or path to the folder that contains the xml-file(s)>
```

Furthermore, you can configure, in which archive a document should be imported and which metadata should be sent to bitfarm. The configuration file (ini-file) always contains a `[main]` section, in which you can determine the following variables.

- `profile=`Name of the bitfarm-profile
- `bfauser=`bitfarm-user, which is used to sign on to the bfaServer 36 (encrypted)
- `bfapass=`Password of the bitfarm-user (encrypted)
- `rootnode=`Root-tag
  All XML-nodes, that are selected in the configuration use this node as the source. You will also need to include the enclosed tag (for example <rootnode>).
- `document=`file or <name of the node, in which the document is referenced>
  With this you can control how the document is found:
  - The document has the same file-name as the xml-file and is located in the same folder: `document=`file
  - The path to the document is located within a node of the XML-file: `document=`<name of the node>

    Hint: The path can be an absolute one (including folder). If only the name of the document is entered, the document is expected to be in the same path as the XML-file.
  - `bitfarmpath=`Path to the bitfarm-Archiv-program folder (Server)
  - `templatepath=`Path to the templates-folder
  - `uebergabe=`Path to *bitfarm-Archiv\transfer*
  - `logfile=`Path to the log file
  - `loglevel=`info or debug
  - `deletexml=`False or True

    XML file is deleted after transfer (true) or not (false)

- deletedoc=False or True

  Document-file is deleted after transfer (true) or not (false)

b) Setting bfauser/bfapass

The bitfarm-user and password are encrypted and put into the configuration. Run a command console on the computer, on which the import is done, using the Windows-user who should run the import. Use the console to run the program `bfa_xmlimport.exe` and enter the following parameters `-cred <Path to the configuration file>`.

Now, enter the username and password and confirm with 'enter'. The log-in data is saved automatically in the configuration in `[main]`.

Warning: If the computer, or the user on which the import occurs, changes, you will need to repeat this process to update the log-in data.

c) Sorting

In the 'archives' section of the configuration you can set conditions that determine in which archive (name of the template) a document is imported. If none of the conditions are met, the xml-file will be ignored and the document will not be imported. You need to enter the base name and condition of a template in each line underneath the 'archives' section. For example, if a document should be imported into the archive with the name 'received invoices' when the XML-file contains 'invoice' in the nodes `<head><doctype>`,you need to enter the following line in this section:

`received invoices=(<head><doctype>=invoice)`

You can also use logical statements as well as „and“, „or“, „and not“, „or not“, and parenthesis for constructions such ((<head><doctype>=invoice) and (<head><topic>= renovation)).You need to put every comparisons and the entire expression into parenthesis. To compare values you can use the operators „=“, „<“, „>“, „<=“, „>=“, „contains“and „contains not“.

d) Metadata

For every archive (more accurately: template-name) you can determine which additional- and status field values should be extracted from the XML-files. For additional fields you create a new section „add“<name of the templateme>, for status fields a section „stat“<name of the template>.

e) Status fields

Let us assume now for the above mentioned example that the archive 'received invoices' (`received_invoices.tpl`) 'booked' should be set, when the nodes `<head><booked>` of the XML-file have the value of 'yes'. It would appear like this.

```
[stat_received_invoices]
booked=(<head><booked>=yes)
```

You can use the same operators and keywords as in the section 'archives', in the conditions. Only if it is *true,* the field is activated.

f) Additional fields

Additional fields are configured, like status fields, per archive, with each having its own section with the following structure:

```
<name of the bitfarm additional field>=<value>
```

<value> can be a certain node of an XML-file. Let us assume that in the previous example the 'received invoices' archive has an additional field 'invoice amount'. The field should receive the value from the XML-node `<head><inamount>`. It would appear like this:

```
[add_received_invoices]
invoice amount=<head><inamount>
```

You can expand the value with free text and/or with the values from multiple nodes. Let us assume the following XML-structure:

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

Set the value of the additional field 'test' of this xml to "t1 - t2". It should have the following configuration:

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

Nodes are always surrounded by '<' and '>' in the configuration. If two nodes appear consecutively, the second one is interpreted as a kind-element of the first one. If they are separated from each other, they are treated as individual nodes on the same level. In the example-XML 'test1' is a kind-element of 'subnodes'. If the configuration contained the following:

Test=<subnode> <test1>

the xml would appear like this:

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

In this case, the additional field would receive the value 'UK1 - t1''.

When setting up additional fields you can, in addition to simple mapping, also work with 'list' and 'if' commands to read multiple nodes of the same name (list) or to determine a node value (if).

g) List-commands for additional fields

In an XML-file there can be multiple nodes by the same name, which contain different data. To determine from which node an additional fields should be drawn from, you need to define a condition, which identifies the node. Let us assume the following XML-structure for visualization:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<indexing>
    <head>
        <address>
            <sentencetype>Erfasser</sentencetype>
            <addresstype>colleague</addresstype>
            <Name1>Homer Simpson</Name1>
            <street>Evergreen Terrace 742</street>
            <location>Springfield</location>
        </address>
        <address>
            <sentencetype>Bezogener</sentencetype>
            <addresstype>seller</addresstype>
            <Name1>Morticia A. Addams</Name1>
            <Name2/>
            <street>1313 Mockingbird Lane</street>
            <location>Mockingbird Heights</location>
        </address>
        <address>
            <sentencetype>receiver</sentencetype>
            <addresstype>customer</addresstype>
            <Name1>Wile E. Coyote</Name1>
            <street>Desert Road 12</street>
            <location>Desert</location>
        </address>
    </head>
</indexing>
```

Here we can see 3 nodes with the name 'address' the data of which should be mapped onto the additional field. Let us assume that the target archive has 3 additional fields: 'employee', 'seller' and 'customer', which need to be filled with the related value from 'name – location' from the XML-file. Expected values:

Employee: "Homer Simpson - Springfield"
Seller: "Morticia A. Addams - Mockingbird Heights"
Customer: "Wile E. Coyote - Desert"

The general syntax of an if-command is:

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

The configuration for this example would be:

```
employee=If(<AdressTyp>=employee, <head>, <address>) <name1> - <location>
   seller=If(<addresstype>=seller, <head>, <address>) <name1> - <location>
   customer=If(<addresstype>=customer, <head>, <address>) <name1> -
<location>
```

When entering the 'condition' you can work with the aforementioned keywords and operators.

If the node 'address' only exists once in the example XML, and you only want to read a value when they meet the conditions, you can omit the use of `<startnode>` and `<iternode>` because you will only need the conditions. If it is `true`, the value is read. When using this syntax only the first found node is analyzed, whereas other nodes of the same name are ignored.

h) Plugin-configuration

You can create a 'plugins'-section in the configuration-file. Per archive (remember: those are actually the names of templates) you can enter the path to a plugin (executable file) before the document and its job-file are sent to the bitfarm-server. The plugin will receive the generated job-file.

An example for such a use of plugins: if a MYSQL-timestamp should be read from a XML-file, which should be sent to a bitfarm date-field. To successfully import it to bitfarm the date in the job needs to be in the DD.MM.YYYY format. A plugin can take care of transforming the MSQL-timestamp into a fitting format.

II. Carrying over document information from the DMS to other systems

1. Job-file

In the `scripts.ini` you can use
`autoexportpath=`
to define a path for the export of metadata. For each new document that is imported into the DMS the information that is written into the database is sent to the export-folder as a job-file. You can use this information in other systems, for example to enable configuration of the ERP-/accounting-systems as leading applications. The organization of the read documents also need to be done through an external system. Example for an exported .job-file:

```
[Version]
Version=36
[archive]
profile=template-gmbh
Name=creditors
table=28052015172108
Arcid=10
[Document]
DOCID=10.531
gdoc_id=531
title=invoice Ingram Micro-09586-11
Orginal=%bfastore0%\daten-rs\2019\09\20\1992011957NBLK236825147.tif
copy=
source=
user=m.mustermann
userid=
fulltext=%bfastore0%\daten-rs\2019\09\20\1992011957NBLK236825147.txt
Status=0
StatStr=document indicated and archived.
OCR_QF=-1
OCR_Typ=OMP
keywords=
date=2019-09-20 11:09:34
filter=
Pages=1
[Hash]
SHA1=
[reference]
number=0
[SVN]
SVN_Link=
SVN_Version=
```

```
SVN_Revision=
SVN_date=
SVN_Info=
[Tasks]
Termin=
Alarm=
editor=
note=
TimerOptionen=0
[Zusatzfelder]
Felder=17
ZusTitel1=Ordernumber
ZusFeld1=add_5
ZusWert1=09586-11
ZusTitel2=Invoicenumber
ZusFeld2=add_3
ZusWert2=44-3281708
ZusTitel3=Invoicedate
ZusFeld3=add_2
ZusWert3=15.10.2019
ZusTitel4=Invoiceamount
ZusFeld4=add_14
ZusWert4=622,37
ZusTitel5=Auditor
ZusFeld5=add_14
ZusWert5=m.mustermann
…
```

The gdocid can be useful for external systems. With it you can access the document from external systems via the DMS-client. Using the additional fields, which can also have a barcode, you can establish a connection to the document in external databases.

2. Excel-Export via configurable viewer plugins

Metadata in additional fields can be exported to Excel, or as a csv-file by right-clicking them. However, this will also export the document-specific metadata as well as all the additional fields, which you might not need. Using the Excel-Export-Viewer-Plugin, you can define exactly which additional field-values should be exported.

a) Preparation

Move the content of the folder Excel-export, which you can find in *%bitfarm-archiv%\install\Bitfarm-Tools\Viewer-Plugins\Excel-Export*, to the folder *%bitfarm-archiv%\Viewer-files\plugins*

Attention: To edit the configuration-file, always use an editor that supports UTF-8 (not Windows editor)

b) Configuration

`Excel-Export.vbs`

Attention: For `Excel-Export.vbs` use ANSI

`exportpath="askuserfile"`

You can choose between three different options. With „`askuser`"the user is asked to enter a path, to which the export should happen. „`askuserfile`" asks for the path + the file ending. Make sure to always include the ending. Enter a UNC-path and it will is immediately to the entered path.

`hyperlinks=True`

Hyperlinks are used to access document in bitfarm-DMS via a click. If it is set to `True` the hyperlinks will also be exported, if it is set to false, only the Gdoc-IDs without link are written

```
'######## adjust here ###############
exportpath="askuserfile"
hyperlinks=True
expandmulti=False
```

into the file.

`expandmulti=True`

If multiple-selection fields should be exported into individual columns, enter `False`.

```
'#######################################
'#################### adjust here ######################
sourcedir = "C:\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
'#########################################################
```

In the section underneath you also need to adjust the paths to the environment.

`Plugins.ini`

In the `plugins.ini` you need to adjust the `counter`. The value goes up by one for each plugin.

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

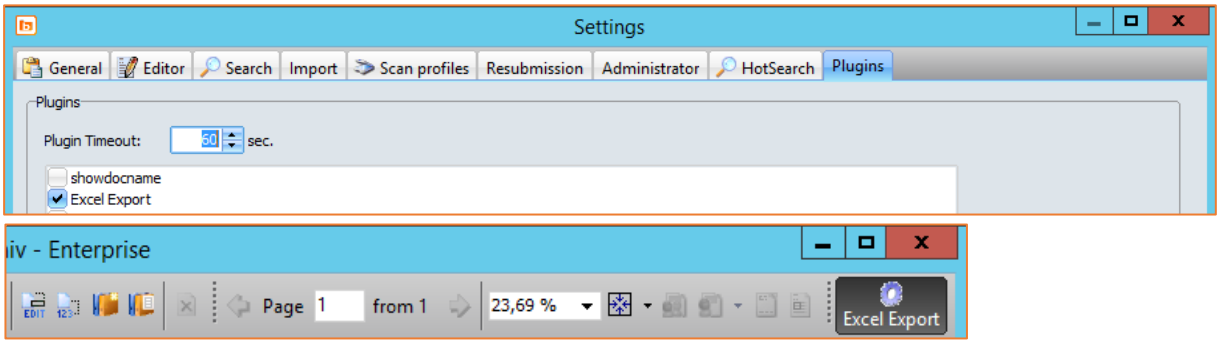
Under `[Names]` you select the script that should be executed. In our example this would be: `Excel Export.vbs`

Hint: Enter the entire name of the file that should be executed.

`file0` names the first plugin. Each new will create another entry, which means that the second one is file1. In the section `[Params]` are sent to the plugin parameter. In this case the job-file of a document, on which the plugin runs, is sent to the plugin. To find out, what sending-parameters exist, consult the system manual or ask the bitfarm-Archiv software-support.

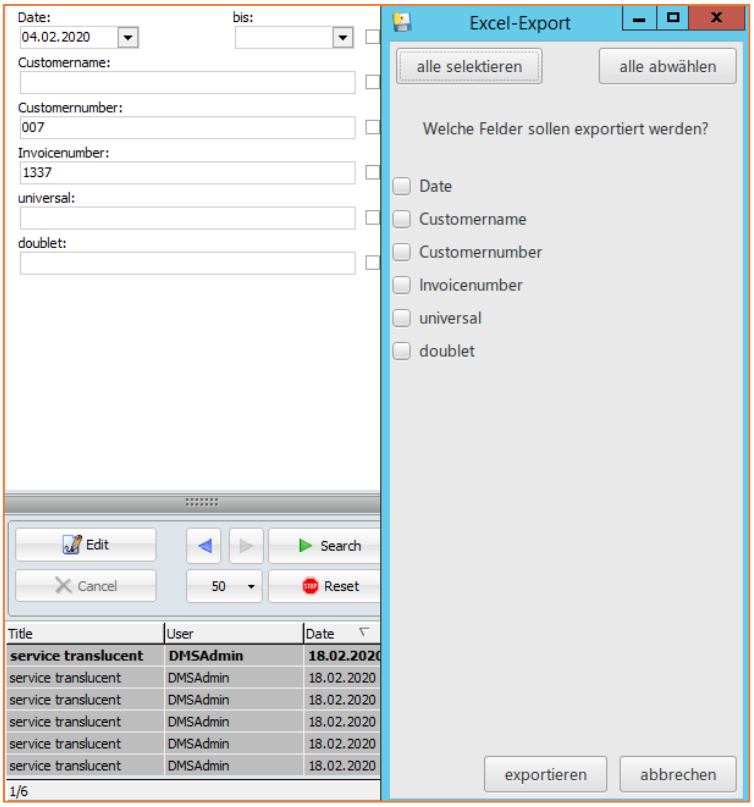
c) Executing the plugin

If you have more than one document in the list, you can select the plugin via right-click in the context menu, or via the button in the viewer. If you want a viewer plugin to appear as a button



in the viewer, you need to set it up in the viewer options.

The first start of the plugin takes a bit more time because it needs to be installed on the client. When it starts, you can see a window, where you can select and de-select additional fields.



If additional fields are selected, click on 'export'. Depending on the configuration, you will now see a new window, in which you can determine where the file should be saved to. Now, you can open the Excel-file, which contains the metadata that you have selected and exported.

	A	B	C
1	gdocid	Customer number	Invoice number
2	gdocid 32	2131634	20938
3	gdocid 38	7483926	12935
4	gdocid 34	3274894	83728
5	gdocid 31	5829371	837293
6	gdocid 28	2182738	27382

3. Advanced CSV-export via configurable viewer-plugins

There are many reasons to make archived data and/or metadata available for other programs such as a Fibu-Software. You can export selected metadata (if needed, with an image-file) via the tool bfa_csv_export as a .csv-file.

a) Preparations

Hint: To set up the tool you need to have a compatible version of bitfarm-Archiv DMS installed. If you have an old or incompatible version, update it through the customer section on the bitfarm-Archiv website and the related packages of the enterprise-version.

The tool can be found in the folder *…\bitfarm-archiv\install\Bitfarm-Tools\Viewer-Plugins\bfa_csv_export.*

Attention: To edit the configuration-file of the bfa_csv_export you need to use an editor that supports UTF-8 (no Windows-editor).

Send the folder and the file bfa_csv_export to the folder *…\bitfarm-archiv\viewer-files\plugins.*

b) Configuration

In the file `bfa_csv_export.vbs` you need to edit the section that is marked by 'edit here'. Make sure to use **ANSI** when editing this section. If you have made no changes to the paths, you will not need to change anything here.

If you did not send the folder in the *Viewer-files\plugins*-folder you need to enter the path to `bfa_csv_export.ini` and the folder in which the `bfa_export.ini` can be found after `configfile=` and `sourcedir=`.

After `updatecheck=` and `startlocal=` you enter the settings for the client. If the tool

```
'#################### adjust here ####################
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
'#####################################################
```

should start on the local client, you need to enter `startlocal=True`. With `updatecheck=` you decide if the tool is adjusted to the server-version and if it should be updated together with the server version.

c) bfa_csv_export.ini

Attention: Only use UTF-8 to edit the INI.

Open the `bfa_csv_export.ini` in the folder …\bitfarm-archiv\Viewer-files\plugins\bfa_csv_export\ with a suitable editor.

d) [main]

`[main]`-makes up the main section of the ini-file. With the variable `exportdir` you select the folder, in which the tool should send the documents. Enter a path that can be reached from the client.

With the switches `export_csv`, `export_tif` and `export_pdf` you can determine which

```
[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
```

file-types you want to save.

With `tif_filter` you can set a guideline for `TIF`-documents. You can find multiple example-files (TIFF-100.txt, etc.) in your *bitfarm-Archiv*-folder. You can also write your own filter-guide-lines. You can find a description and the related commands in one of the filter-files.

Protocols, which are written while running the tool, are sent to the `logfile`. Here you can enter a path in which the protocols should be saved. We recommend the folder `Logs` on the

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

server. You can add enter the user of the tool as a parameter in the file name (as shown in the example). Date-formats can be changed using `datetimeformat` and `dateformat`.

Hint: The date-format is built following strict rules. You can learn about it by researching the Python function strftime. If you have no experience with it, we recommend not changing anything.

Naming the for export can be done via `exportfile`. You can select many different parameters as the file name, such as the archive-name, the date, or information from additional fields. You can transfer the value with %-symbols: `%archiv%.` Other options include: `%gdocid%, %date%, %datetime%` oder `%Zusatzfeldname%`. If the user should receive a message-box after exporting, set the switch `showmsg` to `True`. In some cases it might make more sense to process a single CSV-file for different booking entries, instead of mulitple CSV-files for each one. To make the viewer create a CSV-file for all the selected documents, you need to set the switch `multidoc_csv` to `True`.

Hint: With Multidoc -exports you need to set `export_TIF` and `export_PDF` to `False`.

e) [csv]

Attention: In the section `[csv]` the different csv-parameters are defined.

Usually there is no need for change here.

In the section `[csv]` you can adjust the separators, escape-symbols, line-ends, and quotes. `delimiter` describes the separators. Make sure to always put them in quotation marks. With `doubleqoute` you can determine how an object is marked in the set symbol (`quotechar`). If `doublequote` is set to `True`, the object is put in between the symbols set in `quotechar` surrounded by two sets of quotation-marks. If it is set to `False`, only one set of quotation marks. If `doublequote` is set to `False` there needs to be an escapechar. With the variables `escapechar=` you determine which symbol is used to mark other symbols as special.

What appears at the end of a line is determined in `lineterminator`. By default it is set to \r\n. With the quotechar command you can define the marking-symbol. Always enter it in quotation marks (`"'", "#", """`). With quoting you determine what should appear in the quoting-symbols. You have different options: `QUOTE_ALL`, which puts all objects between the

symbols, QUOTE_MINIMAL which sets objects with special cars in quoting-symbols, QUOTE_NONNUMERIC which converts all number-types to a float, QUOTE_NONE which leads to an error-message if there is no escapechar.

f) [header]

```
[header]
A=GDocID
B=Amount
C=Date
D=Customername
E=Customernumber
F=
G=
```

The section [header] includes the headers set in the CSV-file.
The letters denote the columns. If you enter GdocID under the 'A'-column, the gdocid of the document will appear in the first line of the 'A'-column. The same applies to all the letters. Every letter stands for the column headed by it like in Excel.

```
[row2]
cols=Here|row|two|for|comments|||||||||
```

	A	B	C	D	E
1	GDocID	Amount	Customername	Date	Customernumber
2					

In the section [row2] you can create more headlines.
Here, no letters are used to identify the columns, but instead a vertical line (pipe). Do not use [row2] to export values of additional fields, instead this is done in the following section

g) [cols]

```
[cols]
A=DocNr%gdocid%
B=%Amount%
C=%Customername%
D=%Date%
E=%Customernumber%
F=
G=
```

In this one, letters are again used to identify the columns. To show the GDocID, enter %gdocid% into the fields. You can also add text before and after the variable, like in A=. Here the text DocNr

	A	B	C	D	E
1	GDocID	Amount	Customername	Date	Customernumber
2	Here	row	two	for	comments
3	DocNr272	600	Elisabeth Nomer		89756
4	DocNr273	1337	James Bond		96587
5	DocNr274	430	Bryan Laser		56489
6	DocNr270	600	Adam Peters		36591
7	DocNr271	67	Paul Parker		36987

was put in front of the variable %gdocid%t. If additional fields need to be exported, every field needs to be put in between %-symbols.

h) [conditions]

```
[conditions]
archive=Datev
zus_Customernumber=!=|
sts_Checked = True
```

The [conditions]-section is needed to create conditions. With this you can define, under which circumstances a tool is allowed to export.

You can use the variables like this:

archive= determines in which archives the tool can be used. You can enter multiple archives divided by semicolons. The rule for all conditions is that if nothing is set, everything can be exported. To tie the export to additional fields, you need the zus_Customernumber= . Custombernumber can be any additional fields which should be checked. The first equals-sign which appears in zus_Customernumber relates to the variable. The operator after the first equals-sign, != checks if the additional field is not empty. The vertical line (pipe) functions as a separator between operator and value (in this case a null value). Once the additional field Customernumber is empty, the export is stopped.

You can also send requests to status-fields. For this use: sts_Checked. Checked can be any status field name that needs to be checked. More operators for checking additional fields are:

= checks for equivalence. For example: zus_Customernumber==|57893
< less than, which checks all values smaller than the one entered.
> more than, which only allows values larger than the one entered
~ includes a string of symbols that needs to appear in the additional field. For example: zus_Customernumber=~|89. All documents which include a 89 are exported (including the customer number 57893).
.> starts with every additional field, that starts with, for example 57
.< ends with every additional field, that ends with, for example, 93.

i) [update]

With the [update]-section you can update metadata of documents after export. If documents need to be sent before updating the additional- and status fields, set movefirst= to

```
[update]
movefirst=False
moveto=78
zus_Date=%Date%
cast_TotalAmount=string_to_decimal
stat_Checked=True
```

True, if not, set it to False. To select the archive that needs to be moved with moveto=, you need to have the archive-ID, which can be found in the admin-section of the viewer. Select it and move it after the variable moveto=. The following variables can be used to edit status- and additional fields. After the export it might be useful to set a status field which refers to the exported document (for example: by copying the status). For this use the command stat_Exportiert=. All available status fields can be edited, just replace Exportiert with any status field you need. This works by setting it to true True. You can edit additional fields even after export. With zus_Date= you can change the date additional field. With %date% you can set the date of the export.

```
[Conf]
counter=1
[Names]
file0=bfa_csv_export.vbs
[Params]
file0=jobfile,tiffile,waitexec,refreshlist
```

If you need to edit values, that need to be converted to a different format, use the corresponding *caster*. Make sure to enter the correct function. You can also create your own casters, if needed, by editing the casting.py-file. Here, you can find other functions that may help you.

j) [casting]

```
[casting]
Amount=decimaltostring
Customernumber=decimaltostring
```

The [casting]-section changes the character format. You can edit all of the set additional fields. For exports, you can convert the absolute value from a decimal to a string.

All of this happens with the template caster decimaltostring. Numeric values should always be converted in to a strong when exporting, because leading zeroes might be deleted, which can lead to unwanted behavior, or even, corruption.

The following structure needs to be maintained: Zusatzfeldname=Variable of the caster.

k) Plugins.ini

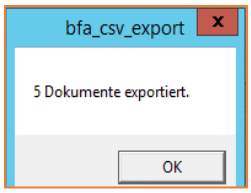
In the plugins.ini the counter need to be counter. It is raised by one for each added tool.

Under [Names] you can enter the script (with ending) that needs to be run (in our example bfa_csv_export.vbs). Enter the entire name of the file that should be executed. file0 stands for the first tool. For every additional tool a new entry is create, so the second one is called file1. In the section [Params] the tools receive the parameters. jobfile transfers the metadata of the document to the tool. With tiffile you can determine that the tool should also receive the TIF. waitexec pauses the viewer as soon as the export is started. This prevents corrupted data. refreshlist refreshes the archive once the tool has been started. If status- and additional fields are changed, the changes are visible right after export. The parameter tiffile is mandatory in this case. You can get more transfer-parameters from the bitfarm software-support.

l) Viewer-configuration

Start the viewer and open the settings (file → settings), then select the tab plugins from the settings-window. Here you can add tools to the quick selection by checking the box next to the name of the tool. You can also start tools by right-clicking the document and then selecting plugins.

Select one or more documents from an archive and the start the tool. After the export is done you should see the following:



III. Transferring metadata from external databases

1. bfa_sqlmapper

a) Application

Usually the metadata of a document is entered by the user, or extracted from the full-text via keyword-rules (wfd-file). Sometimes you need metadata that cannot be found within the document itself but in external databases. You can use the SQL-mapper to request values from a document via ODBC from external databases.

You can connect the SQL-mapper in three different ways:

- Connection via a new command 'sqlmapper in the wfd-naming sections'
- Starting it as a plugin from the viewer
- As a stand-alone program, manually or repeated via a planned task

b) Examples

- A customer-number is extracted from a document using a wfd-rule. The name of the customer cannot be found in the document and should be extracted from a database, which contains the names of the customer, connected to their number.
- In bitfarm-Archiv offers are archived. The related offer-number is extracted from the full-text and saved in an additional field. The DMS also contains contracts together with their number, which is saved in an additional field. If in addition to the contract-number you want to fill a field with the offer-number related to the contract, you can fiond the information in an external database.

c) Settings/configuration

The configuration of the swl-mapper is done via ini-file (by default in *%bitfarm-archiv%\bfa_sqlmapper.ini*). When editing the configuration-file make sure to use a suitable editor, which does not change the encoding of the file (UTF-8). Notepad is known to change the encoding when saving. We recommend the use of a modern text-editor, for example Notepad++. This file is not only used for general settings (such as logging- and connection files to external data-sources), but also for the SQL-queries and mappings of bitfarm-Archiv. The SQL-mapper also needs access data to log on to the bitfarm-server.

You need the following two configuration sections.

```
[main]
bfauser=AQAAANCMnd8BFdERjHoAwE/Cl+…
bfapass=AQAAANCMnd8BFdERjHoAwE/Cl+…
profile=template362

[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
```

In the `[main]`-section you can find the access-data for the bitfarm-server client next to the username. This data is encrypted. To encrypt of data you need the command-line tool `credcrypt.exe`, which can be found in the folder *%bitfarm-archiv%* on the bitfarm-server. Log on to the server with the user that is running the bitfarm-clients and open a regular console in the folder *%bitfarm-archiv%.* Enter `credcrypt.exe` ,followed by the user name with which bfa_sqlmapper should log in (for example DMSAdmin). The encrypted username is shown on the console and copied. Enter it in the configuration after `bfauser=`. Use the same process to create the encrypted password of the user and enter it after `bfapass=`.

Attention: For de- and encryption the Windows Data Protection API is used (further information at https://msdn.microsoft.com/en-us/library/ms995355.aspx). If you change the bitfarm client-user, or if the server is moved to different hardware, you need to encrypt bfauser and bfapass, even if the username and password are still the same!

In the `[logging]`-section you adjust the following settings:

- `viewerplugin/standard/standalone`: For each context you can enter an individual path for the log-file. Make sure that the path to the log-file, in the context-viewer-plugin, is a path from the perspective of the client!

- `level`: with this you can determine how extensive the log should be. Possible values are: 'DEBUG' (extensive) and 'INFO' (less extensive)
- `clientlog`: Controls determines if information on the communication between the sql-mapper and the bfaserver should be recorded.
- `backupcount`: Every day a log-file fill be generated. Via backupcount you can determine how many log-files should be used as backup.

In the following sections you set the configurations specific to the user. Example- configuration (minimal):

```
[get-customer_name]
constring=dsn=customerdb-odbc
sql=select customer_name from customerdb where customer_number=%customer_number%
customer_name=%0%
Overwrite customer_name=True
emptyonmissing=True
emptyonnoresult=True
```

The name of the section can be anything, but it needs to be unambiguous within the configuration. First, you need to enter ODBC-source in to the operating system. On Windows, you can start the configuration-interface with the command *%windir%\syswow64\odbcad32.exe* (always start the 32-bit version even on 64-bit systems). Select the tab 'System-DSN' in the configuration-dialog and click on 'add'. In the following dialog you need to select the driver, or install it if it is not installed yet (in case of any confusion ask the manufacturer of the external database). In the following dialog you can find the names of the data sources and further details on the ODBC-access. Click on 'test' to make sure that the configuration is correct and that a connection is possible. Now, you need to enter the reference to the new ODBC-connection as a connection-string in the option 'constring'. How to create one depends on the type/driver of the data-source(you can request this information from the provider). In many configurations you only need to enter the dsn like this: constring=dsn=<Data Source Name>. You can find further information on this at: https://www.connectionstrings.com.
After the ODBC-connection has been set up you can use the sql mapper to check the connectivity of the external database. To do this open a console on the bitfarm-server in the folder %bitfarm-archiv% and enter the following command:

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

With the parameter `-s` you can choose to run the test on all sections found in the configuration. You can also, instead of using 'all', enter the names of the sections that you want to check, individually. If it is unsuccessful, the program will display the reason for why no connection could be established. Next, you need a SQL-query that delivers the desired information. The value for the search comes from an additional field. Put the name of the additional field between %-symbols in the query. In the example above you can see that the search was carried out with the value in the field 'customer-number'.
Now, we will look at how to map the results of the query into the related additional fields. Select the name of the additional field as the option and for the value select the column-index

surrounded by the %-symbols from the results of the query. In the example above there is only one column as a result (with the customer_name) therefore, you would use '%0%' (Index starts at 0!). In addition to the index you can also use fixed strings as values. This way you can supplement the value from the database with additional fixed information. If you entered ---%0%--- as the customer-name and the query showed 'bitfarm-Archiv GmbH', the value of the additional field would be '---bitfarm-Archiv GmbH---'.

The two switches `emptyonmissing` and `emptyonresult` determine if the provided target-fields should be emptied, if the the searched value is missing (`emptyonmissing`), or if the results of the query are empty (`emptyonresult`). Both switches are set to `False` by default.

If the target field already has a value at start, it will not be overwritten. If you want that to happen, you can activate the option `overwrite_<Feldname>=True` (in the example above that would be `overwrite_customername=True`).
This concludes the minimal configuration for the SQL-mapper. In the section 'advanced configuration' you will find additional switches, with which you can work for more specific applications.

d) Connecting the sql-mapper via wfd-rules

In order to compare the metadata after archiving a document, you need to connect the sql-mapper in the wfd-file of a `naming section`. Make sure that the value used to perform a search in an external database (in our example the customer-number) is already available during the archiving-process. That means you should enter the customer-number into the import-window, or determine it via naming-rule from the scan.

In the `naming section` you can define conditions for executing this section (for example: `archivtabelle=ReceivedInvoices` etc.). Then, you just need to enter the section from the sql-mapper configuration, which should be executed in this context:

```
naming section crm-customer
archivtabelle=customer
sqlmapper=get-customer_name
end section
```

You can make multiple sql-mapper assignments within a single naming section to use multiple sections of the sql-mapper configuration.

Hint: When processing wfd-rules, sql-mapper-commands are executed only after all the other sorting- and naming-commands are done. This way the lookup-value can be read via wfd.
When the scriptdebug in the `archiving.vbs` is activated, the access, as well as the results are protocolled in the event-log. You can also find the log-notifications in the configured log-file of the sql-mapper.

e) Connecting the sql-mappers as a viewer-plugin

In order to run the sql-mapper on the client, you need install the ODBC-connection on the client (as shown before). Next, you need to make sure that to adjust the file `plugins.ini` in the folder *%bitfarm-archiv%\viewer-files\plugins* (see: system-manual, section 'plugin options') enter the script `bfa_sqlmapper.vbs` into the `plugins.ini` and use 'jobfile', 'waitexec', and 'refreshdocument' as parameters. As with all the other plugins, you need to restart the viewer after connecting the sql-mapper. Start the plugin to check if everything works as it should. If the target-field does not have the expected value after start-up, check the sql-mapper-log for possible reasons of the error.

f) Advanced viewer-plugin configuration

By default, all sections of the `bfa_sqlmapper.ini` are run when executing as a viewer-plugin. If that should not happen, and, for example, only one specific section should be analyzed, you can make the necessary adjustments, in the `bfa_sqlmapper.vbs` (replace the line: section="all" with for example: section="get-customer_name").

Alternatively, you can formulate conditions in the `bfa_sqlmapper.ini` for each section, that should be analyzed by the swl-mapper. The section will then only be run, if all the conditions are met. For example, if the section [get-customer_number] should only be run, if the document comes from the 'customer_invoices' archive, you can add a new section in the `bfa_sqlmapper.ini` called `[vp_cond_get-customer_name]`. (vp_cond_<name of the section>). Within this section you can now establish conditions. In our example it would look like this:

```
[vp_cond_get-customer_name]
archive=customer_invoices
```

You can also use additional- or status fields as conditions for analyzing the section. Additional-field options always start with `add_` followed by the name of the additional field. Status fields start with `sts_` followed by the name of the status field. For additional fields you first need to add an operator after the equals sign (possible options: '=' (equals), '<' (less than), '>' (greater than), '!=' (not equal to), '~' (includes), '.>' (starts with), '.<' (ends with). Behind the operator you need to add a pipe (|) to divide the operator from the reference-value. In the notation for status-field conditions there are no operators. Instead, you can work with `sts_<name of the status field>=True` or `sts_<name of the status field>=False`.

If, in our example, the section [get-customer_name] should only be run by the sql-mapper as a viewer-plugin if

1. The document is located in the customer-invoices archive
2. the customer-number starts with '1'
3. the status of the document is set to „booked!.

then the complete conditions-section would look like this:

```
[vp_cond_get-customer_name]
archive=customer_invoices
add_customer_number=.>|1
sts_booked=True
```

g) sql-mapper as standalone-Tool

In some cases it might be useful to start the mapper in regular time intervals while running the metadata-comparison for multiple documents in the background. When running the mapper as a viewer-plugin, it will process the documents selected by the user, if it is connected via the wfd-file, it will process the document currently selected for archiving. To establish stand-alone contexts, which determine which documents should be compared by the sql-mapper, create a bookmark using the user that has been encrypted and put into the configuration. In our example we could create a bookmark which finds all documents, where the customer-name is still empty. You need to enter the name of the bookmark as a bookmark into the configuration section. If the bookmark is called 'no_customer_name', enter bookmark=get-customer_name into the section `[get-customer_name]` :

```
[get-customer_name]
constring=dsn=customerdb-odbc
bookmark=without-customer_name
sql=select customer_name from customerdb where customer_number=%customer_num-
ber%
customer_name=%0%
overwrite_customer_name=True
emptyonmissing=True
emptyonnoresult=True
```

To run the sql-mapper open a console on the bitfarm-Server in the folder *%bitfarm-archiv%* and enter the following command:

```
bfa_sqlmapper.exe -c standalone -s get-customer_name
```

With the parameter `-c standalone` you can make the sql-mapper run in the standalone-context. With the parameter `-s get-customer_name` you select the section[get-customer_name]. Any errors that occur during the execution will appear in the console and be written into the logfile, which we configured in the section logging under standalone.

h) Advanced configuration

1. index-lookup

In our example we are searching for the name of the customer using the customer number that we have gathered from the document. There might be more than one number extracted from one document, in that case those might be put into a universal field, separated by semicolons.

In that case you need to enter the customer number, with which the look-up is done, into the configuration of the sql-mapper. Here, you also have the option to enter the index of the used value (starting with 0!). Expand this section with the entry 'lookupindex=0', to always use the first number from the additional field. With 'lookupindex=1' the second entry is used. If all customer numbers should be used for the look-up, enter 'lookupindex=all'. In this case there are multiple lines of results. Therefore, the additional fields will need to be of the universal or selection (or multiple selection) because only those can carry more than one value.

2. inputcaster/outputcaster

Sql-errors that appear when running the mapper might be due to an incompatible data-type. For example, if the look-up-value comes from a universal field, it is a Unicode-string. There is a chance that the look-up-value consists of only numbers and is saved as an integer in the external database. When you run the sql-statement, it will result in an error because the datatype is incompatible. The value of the additional field needs to be converted into an integer, before running the query. To do this you can enter an inputcaster into the configuration. All casting-functions are declared in the file `bfa_sqlmapper_casting.py`. By default, there are already some functions in the script, but you can add more casting-functions if needed. Every casting-function is started by 2 arguments: first, the value from the bitfarm additional-field and secondly by the context on which the sql-mapper is running. The function needs to be written so that it returns the value of the additional field after converting it into a proper datatype.

For our example (look-up-value needs to be converted from a string to an integer) we can use the casting-function ‚toint', which can be found in the configuration-section as ‚inputcaster=toint'.

If you have multiple look-up-values in the query, you need to separate the casting-functions by semicolons analogue to the order of the look-up-values from the sql-statement. For values that do not need to be converted you can use 'pipe' as a casting-function to use the unmodified value in the query.

If you can run the query without any errors and gives results, there might be a possibility that the result-values still need to be converted in order to put them into a bitfarm additional-field. It might be possible that the query delivers an integer, but the result needs to be written into a universal field. Since the mapper knows the datatype of the value from the external database, as well as the target datatype, there is usually no need to enter a casting-function. If the automatic conversion fails, you can still write a function in bfa_sqlmapper_casting.py which converts it properly. This function is saved as an outputcaster in the section. If the query sends a request to more than one column, you need to assign a casting-function to each column (in the same order as the selected columns from the query). If some columns need tob e converted with a special function, while others should be converted automatically, you can use the casting-function ‚auto' for the latter.

2. add_sync_values

a. Application

Sometimes you need values in selection-addfields and you don't want to enter these values manually, but rather compare the possible selection values with, for example, a corresponding table in your ERP database.

b. Examples

Often, the supplier names, your auditors, construction managers etc. should be made available for selection in selection-addfields by syncing with your database.

c. Settings/Configuration

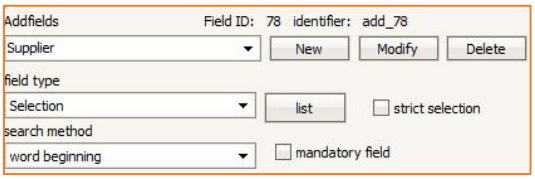
The configuration of add_sync_values is done via ini-file (by default in *%bitfarm-archiv%\ Bitfarm-Tools\add_sync_values\ add_sync_values.ini*). When editing the configuration-file, make sure to use a suitable editor, which does not change the encoding of the file (UTF-8). Notepad is known to change the encoding when saving. We recommend the use of a modern text-editor, for example Notepad++.

The login data is encrypted with the command line tool `credcrypt.exe`, which is located on the *bitfarm* server in the *%bitfarm-archiv%* directory. Log on to the server with the user running the bitfarm services and open a normal console in the %bitfarm-archiv% folder. Enter `credcrypt.exe`, followed by the user name which the tool `add_sync_values.exe` should log in (e.g. dmsadmin). The encrypted username is shown on the console and copied. Enter it in the configuration after `user=`. Use the same process to create the encrypted password of the user and enter it after `pass=`. Enter the path to the con-file after `confile=`.

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

[add_78]
src=odbc
constring=DSN=lfodbc
sql=SELECT DISTINCT splname from supplier
action=add_values
;action=add_values → old values remain, new ones are added
;action=replace_values → old values are removed, all determined values are added
;action=remove_values → removes the determined values from the selection
```

The addfield to be filled in the bitfarm-archiv DMS, is enclosed in square brackets. You can find out the ID of the addfield, for example in the bitfarm-Archiv Administrator.



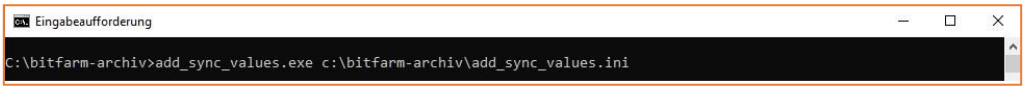
After `constring=DSN=` enter the Name of your 32-bit-ODBC connection. If necessary, you may have to enter a connection string for your specific database. How to create one depends on the type/driver of the data-source (you can request this information from the provider). In many configurations you only need to enter the dsn like this: constring=dsn=<Data Source Name>. You can find further information on this at: https://www.connectionstrings.com

After `sql=` enter the SELECT that must be executed on the Database to be queried to obtain the values that are to be transferred to the selection addfield.

With `action=` you determine how the values are added or removed, as described above.

d. Execution

Attach the file `add.sync.values.exe` via Windows Task Scheduler and enter the path to the ini-file to the arguments for a periodic comparison between your database and the addfield in bitfarm-Archiv DMS. If you want to execute this comparison one, you can also do this at the Windows command prompt.



After the comparison has been executed in one of these ways, you can see the result in the corresponding addfield in the bitfarm-Archiv Viewer.

The selection addfield, supplier, (add_78 from example) before comparison.

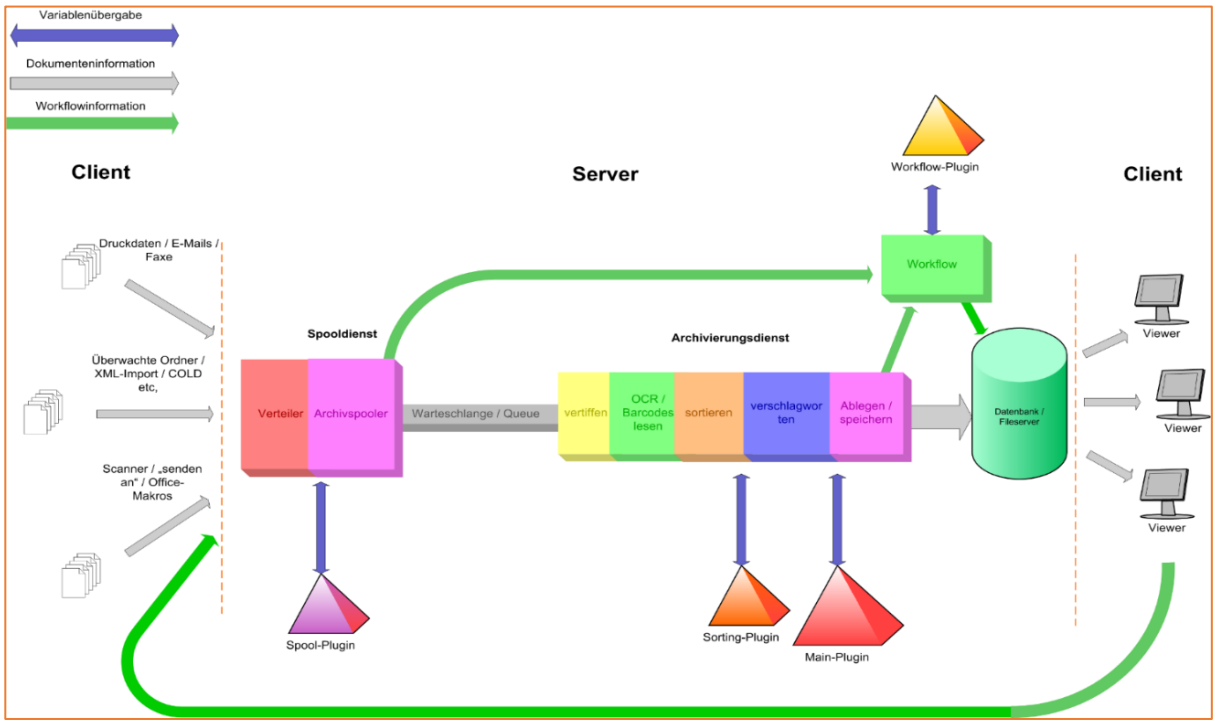
The selection addfield, supplier, (add_78 from example) after comparison.

IV. Individual management via plugins

1. Server-plugins

At many stages of the archiving-process you have multiple ways of using custom programs (plugins) to influence the process, modify variables and field-content, and/or to initiate certain actions. A plugin can be any executable program. The exchange of information with the bitfarm-server happens via a file that is transferred as a parameter at the start. This file is a text-file, that transfers all available variable in the form of `Variablenname=value`. When turning the plugin off the text-file is transferred back to the server. You can find examples for each plugin under ..*bitfarm-archiv**plugins*\ in Visual Basic Script.

The following graphic shows different plugins and at what stage of the process they are located.



2. Viewer-plugins

Viewer-Plugins are saved in...*bitfarm-archiv**Viewer-Files**Plugins* and receive an entry in the `Plugins.ini` which can be found in the same folder. In the section `[Conf]` you can enter the number of plugins, under `[Name]` the name of the file, and under `[Params]` the options. Plugins are shown in the viewer with their base-name (file name without ending).

When starting a plugin the plugin-file is executed and receives the path to the temporary text-file. There you can find the path to the job-file belonging to the document, as well as a preview-tif-file. These files are available for the plugin in the folder *%temp%**Viewer* or *%temp%**Viewer**tif*.

Viewer-Plugins are started either manually via a right-click, or with the related button in the viewer. If you want to do the latter, you need to adjust the configurations of the viewer to show the button.

Possible options for a plugin:

`jobfile`	Export a job-file
`tiffile`	Export the preview-tiff
`waitexec`	The viewer waits until the plugin is closed
`refreshlist`	Refreshes the list of results (search restarted) (also: refresh-list)
`refreshresub`	Refreshes the resubmissions
`resync`	Imports the job-file, the document is updated with the scanned values (save- or move-plugins)

Example: `Plugins.ini`

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

Fore save- or move-plugins you need to enter the base-name of the plugin under `[Conf]` you will also need to check the box ‘plugin active‘ for the archives in the bitfarm-Archiv administrator.

Now the plugins are executed for every save- or move-process in these archives. The save-plugin runs after the entire saving process has been finished, therefore the updated additional- and status fields are already available. The move-plugin runs right before before moving the object. A return-code = 255 cancels the moving-process. To help with performance only the jobfile is available when using the save- move-plugin.

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

For plugins that run with the options `waitexec` or `resync` you can enter a `plugin-timeout=10` (in seconds) in the `con`-file under `[Optionen]`.

3. Client-control via leading applications

The bitfarm-Archiv-Viewer is the central client in the DMS for viewing and working with documents. Apart from just viewing documents, you can also use it for e-notes, database-information, workflows and distribution, as well as research. The viewer not only gives access to the document itself, but it also supplies you with additional information and functionality. When connecting the DMS to other systems it would make sense to use the bitfarm-Archiv client instead of a simple TIF-viewer when viewing documents. In the following we will describe the interfaces available.

Tip: In many cases it might be useful to use two monitors when displaying documents. Thus the dual-monitor set-up is supported by the bitfarm-Archiv-Viewer.

3. Hotsearch-functions

The Hotsearch-tool consists of one file: `Hotsearch.exe`, which should be run via the autostart of the client-systems. Using the hotkeys *Ctrl-C* and *Ctrl-B* you can move text from any Window-application as a search-string into the viewer, which can then execute a full-text search in the last selected archive or storage. The only requirement is that the application allows for transferring information via *Ctrl-C.* You can use this, for example, to copy the contents of the field 'invoice-number' in an ERP-application via double-click, and have it appear after using *Ctrl-B*.

4. Directly accessing a document via GDocID

Documents in the DMS-databse have the so-called *GDocID* as the global primary-key. This *GDocID* is also included when exporting metadata; this allows external systems to save a reference to a document. The request can be done via a simple parameter in the command-line.

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

If the viewer is already running, the view is updated. If it is not, you are asked to log-on to the system after which the document is presented.

5. Programmatic transfer of search-terms

With other parameters you can start a keyword-search or a full-text search in a specific archive:

```
Viewerv3.exe -L:[Lager] -A:[Archiv] -S:[Schlagworte] -V:[Volltext]
-R:[Verknüpfung] -GO -W: -C: -G:[gdocid] docid:[arc_id.doc_id] gdo-
cid:[gdoc_id] [arc_id].[doc_id] -M:
-ZUS:[Zusatzfeldname]:[Zusatzfeldwert]
```

Example: Search for the keyword: 4711 in a specific storage and archive in the viewer:

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

Example: Search for an additional-field title with the value ‚document' and resubmission open:

```
viewerv3.exe -W: -ZUS:title:document
```

The parameters

`-L:[Lager]` changes the selected storage

`-A:[Archiv]` changes the selected archive. The archive must be located underneath the selected storage, if a request is only done with -A:. If you want to change an archive underneath a different storage, you will need to enter that with -L:.

`-S:[Schlagwort]` fills the keyword field for a search

`-V:[Volltext]` fills the full-text search with one or more term

`-R:[Verknüpfung]` fills the shortcut field, the search via shortcuts cannot be combined with the full-text search nor the keyword search.

`-GO` searches via keyword, full-text, or shortcut.

`-W:` opens the list of resubmissions

`-P:` prints the currently shown document

`-C:` closes the viewer

`-G:[GocID]` opens the selected document via the GdocID

or `gdocid:[GdocID]`

`-ZUS:[Zusatzfeldname]:[Zusatzfeldwert]`

Searches for an additional field in all archives; all archives with that includes that field are searched. The search targets word-components (not in detail) and you cannot search for sections. If you specify a storage or archive before searching (with -L: or -A:), then the search will only be done in that archive. If you want to search for additional fields of a different datatype than universal fields (for example date-, float-, or currency fields), you need to format the search term as the type appears in the database-table. For example, when searching for a date, you need to enter it in the MySQL-date-format:
(YYYY-MM-DD), for example -ZUS:Date:2019-10-12
In the case of decimals, you need to replace the comma with a period. For example -ZUS:Amount:219.77

6. Advanced search with a .FND-file

With the bitfarm-Archiv administrator you can create a 'search-template' for every archive. This .fnd-file is a scaffold comparable to a -tpl-file, which can be filled search-information and is sent as a parameter in the viewer. This way you can perform externally controlled searches via specific additional fields.

V. Contact bitfarm-Archiv software-support

In case of questions or problems, consult the bitfarm-software-support

tel.: +49 (271) 31396-0

E-Mail: support@bitfarm-archiv.de

Business hours:

Mon. – Thu. 8:00 AM to 5:00 PM

Fri. 8:00 AM to 3:00 PM

Please have your contract- or partner-number at hand.