Interfaces/Data exchange/Integration options

bitfarm-Archiv – Enterprise V3.6.2

Table of contents

I. Document import

1. Drop into monitored folder

In the configuration file `scripts.ini`, located in the bitfarm-Archiv program directory on the server, you can specify the following under

`ScannerImportPath=`
and/or
`ExtendedImport=`
you can specify the path to a folder that is monitored by *the* bitfarm-Archiv *spool service*. Files placed here are placed in the archive queue and then indexed by *the archiving service*, sorted according to rules, tagged according to specifications, if necessary, and then archived. Further steps, such as *workflows*, can then be carried out. If no sorting rules are specified and no plugin has any influence here, the documents are archived as "undistributed."

It is possible to create subfolders in the monitored folders, which must have the same name as the templates for the archives to which direct archiving is to take place. When documents are placed in these subfolders, the software searches for a template with the same name in the *%bitfarm-archiv%*\templates directory and, if found, places the document in the archive specified in the template.

The target archive or user can also be determined via the file name.

Common file types that can be processed in this way are `TIF, .PDF, .JPG, .GIF, .BMP, .DOC, .XLS, .PPT, .MSG, .EML, .DWG, .PLT, .DXF, .RTF, .HTM, .HTML, .ASC, .TXT`

2. Data transfer with .job files

A similar procedure to data transfer with XML files exists with .job files, which bitfarm-Archiv also uses internally for information transport. Here, a .job file with the same name must exist for each document file to be archived. The import is carried out by inserting the .job file and then inserting the document file into the main transfer folder, which is referred to in the profile file (name of the CON file) as the client pooler or UNC client pooler.

The .job file is generated from a template file. This template is automatically generated by the DMS when the archive is created. It is an empty framework that already specifies which fields are valid in an archive. The template must then be copied by the importing application, filled in, and renamed to .job. An example of such a template file, which is located in *the %bitfarm-archiv%* program directory on the server in the templates subfolder (ArchiveName.tpl):

```
[Version]
Version=36
[Archive]
Profile=sample-gmbh
Name=Accounts-Payable
Table=28052015172108
Arcid=10
[Document]
DOCID=
gdoc_id=
Title=
Original=
Copy=
Source=
User=
userid=
Fulltext=
Status=
StatStr=
OCR_QF=
OCR_Type=OMP
Keywords=
Date=
Filter=
Pages=
[Hash]
SHA1=
[Reference]
Count=0
[SVN]
SVN_Link=
SVN_Version=
SVN_Revision=
SVN_Date=
SVN_Info=
[Tasks]
Date=
Alarm=
Processed-by=
Note=
TimerOptions=0
[Additionalfields]
Fields=17
AdditionalTitle1=Order-number
AdditionalField1=add_5
AdditionalValue1=
AdditionalTitle2=Invoice-number
AdditionalField2=add_3
AdditionalValue2=
AdditionalTitle3=Invoice-date
AdditionalField3=add_2
AdditionalValue3=
AdditionalTitle4=Invoice-Amount
AdditionalField4=add_14
AdditionalValue4=
AdditionalTitle5=Auditor
…
```

The following fields can be filled in by the transferring system:

- Title

- User

- Additional fields

The transferring system can also supply an .ftx file with the same name, which is used for full-text searches. In this case, OCR type TXT must be entered so that no further OCR is performed.

An .slw file can also be provided, the contents of which are then stored in the global keywords.

Please note: There is some data that cannot be supplied by the transferring system. This includes:

- Date

- References

- Appointment

- Alarm

- Processor

- Note

3. Print Archive

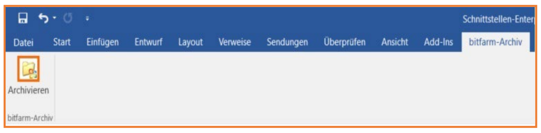
It is possible to set up virtual software printers for the archiving function . These can be set up on the client (see the relevant section in the system manual). When setting up the printer, the target archive and a real printer (which must be installed on the server) can be defined for additional paper output.

Print data can be read and indexed with a very high degree of accuracy using rules for keyword assignment in the .wfd file. The disadvantage of this method compared to XML data import or import via the .job file is that only information that is displayed on the document itself or can be derived from the display can be transferred to the DMS (see plugins).

4. Import of ASCII data (COLD)

For systems (mostly UNIX host-based ERP applications) that cannot export PDF/TIF/DOC or print to Windows printers, it is possible to import print data via COLD. To do this, the data sent to the printer must be written to a file and copied to one of the monitored folders mentioned above. The file extension can be omitted or ASC can be specified. Currently, an Epson printer emulation is hard-coded. For other control commands, please contact bitfarm-Archiv software support.

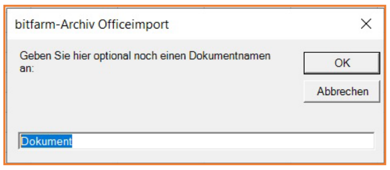
5. MS Office add-ins



For the Microsoft Office package, specifically for Word, Excel, and Outlook, add-ins are available up to and including Office 365 for quickly archiving the original file from the respective application to the DMS. The add-ins are installed automatically with the client installation, but can also be installed manually. For manual installation, please call up `bfaOfficeSetup.msi` on your DMS server.

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

In Word or Excel, select the add-in for archiving the original file.


bitfarm-Archiv Officeimport

Geben Sie hier optional noch einen Dokumentnamen an:

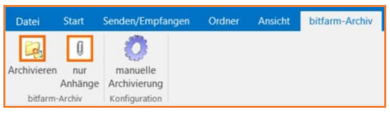
OK

Abbrechen

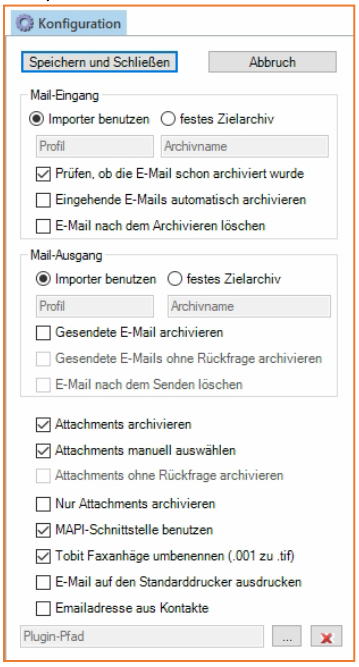
Dokument

If the document is still unnamed, you can give it a name before it is imported into the DMS. Of course, the name can also be changed or specified afterwards.

Note: In the bitfarm-Archiv DMS, documents can also be found by their name.



With the add-in for Outlook, you have the option of archiving only the attachments in addition to archiving the original file (msg file). Click on *manual archiving* to make any settings that suit your way of working.


Konfiguration

Speichern und Schließen | Abbruch

Mail-Eingang
- (●) Importer benutzen ( ) festes Zielarchiv
- Profil | Archivname
- [✓] Prüfen, ob die E-Mail schon archiviert wurde
- [ ] Eingehende E-Mails automatisch archivieren
- [ ] E-Mail nach dem Archivieren löschen

Mail-Ausgang
- (●) Importer benutzen ( ) festes Zielarchiv
- Profil | Archivname
- [ ] Gesendete E-Mail archivieren
- [ ] Gesendete E-Mails ohne Rückfrage archivieren
- [ ] E-Mail nach dem Senden löschen

- [✓] Attachments archivieren
- [✓] Attachments manuell auswählen
- [ ] Attachments ohne Rückfrage archivieren
- [ ] Nur Attachments archivieren
- [✓] MAPI-Schnittstelle benutzen
- [✓] Tobit Faxanhäge umbenennen (.001 zu .tif)
- [ ] E-Mail auf den Standarddrucker ausdrucken
- [ ] Emailadresse aus Kontakte

Plugin-Pfad ... ✗

Automatic archiving of emails does not usually make sense for user mailboxes because it is difficult to know what type of document it is.

Tip: Mailboxes such as Rechnungen@IhrUnternehmen.de can be conveniently archived using the bfaPop3 connector or the imap4 service, as described in the next chapter.

You can configure some settings in the add-in for MS Outlook that relate to your personal incoming and outgoing mail. As mentioned above, in most cases it does not make sense to automatically archive a user mailbox without confirmation. Spam is not usually something you want to archive, and it is difficult for the system to determine which emails are purely correspondence, an offer, an inquiry, etc. etc., which is why you should select the importer (default) and thus determine yourself in which archive the email and/or attachment should be archived. If you activate the option *Select attachments manually*, the attachments of an email are displayed in a separate window and only the attachments that you specify are archived.

6. Automatic email import (POP3)

With the bitfarm email service and the bfaPop3 connector, you can automatically retrieve important email accounts and archive them precisely.

Note: bitfarm advises against archiving all company emails or collective mailboxes such as info@ihrunternehmen.de in this way, but rather specific mailboxes such as rechnungen@ihrunternehmen.de or bewerbungen@ihrunternehmen.de, where it is already clear from the mailbox what type of documents are involved and in which archives they can then be automatically stored.

POP3 service as a service

Please note: If the POP3 service is running in the service context, the mailbox will be emptied as soon as the emails have been transferred to bitfarm.

First, create a *bfaEMail* directory in the bitfarm-Archiv directory and add all files from the directory ...*\bitfarm-Archiv\install\Bitfarm-Tools\bfaEMail*.
In the attached `bfapop3.ini file`, change all necessary entries for your environment accordingly:

```
[Main]
POP3 Server=Mail server name or IP
Username=Testmail@Mailserver.de
Password=test
Transfer=c:\bitfarm-archive\import\Incoming invoices
UseTLS=true
TLSUser=TLS user
```

Use the `POP3 Server` command to specify the server name or IP address of the POP3 server. Enter the account name under `Username`. The `Transfer` command lets you select the archive into which the documents are to be imported. Make sure that the *import* is monitored by the spool service, otherwise the emails will not be imported into the DMS. `UseTLS` allows you to decide whether the login must be done via TLS. The possible values here are `true` or `false`. The `TLSUser` command specifies the (Windows) user of the POP3 account. In addition, if the port has been changed (the default is 110), the `Port=` command can be added, followed by the port used.

Run the command `bfaEMailService.exe -install` in an administrator console and the POP3 service will be installed. In the service context, the login via bitfarm user must still be assigned to it.

As soon as you start the service, it checks the configured mailbox every 5 minutes and sends the emails it finds directly to the path specified under `uebergabe`. You can also specify `–t:<sec>` as a start parameter to change the retrieval interval.

Note: In the service context, the POP3 service writes logs, which are placed in the ...\bitfarm-Archiv\bfaEMail\\logs directory.

Caution: If you want to retrieve multiple POP3 accounts, this is not possible via this service context, as only one mailbox can be specified in the INI file.

Tip: Create a batch file that contains the retrieval of the account you want to retrieve in each line. All parameters are explained and listed below, and an example call is also provided. Once you have created the .bat file, link bfapop3.exe via Windows Task Scheduler and pass the batch file as a parameter.

POP3 service as console call

The POP3 service can also be started as a console call. The possible parameters can be requested with the call `bfapop3.exe –h`:

```
bfapop3.exe <POP3 server|-auto><User><Pass><Transfer path>
{-TLS}{-TLSUser:Name}{-p:Port}{-nodelete}{-log}
```

The meaning of the individual parameters is as follows:

`POP3 server`	Name or IP address of the POP3 server
`-auto`	Parameter for automatic retrieval. Uses the connection stored in bfapop3.ini.
`User`	User of the POP3 account
`Pass`	Password for the POP3 account
`Transfer path`	Directory where emails are stored.

Optional:

`-TLS`	Activation of login via TLS

`-TLSUser`	(Windows) username of the POP3 account
`-p`	Port (default is 110)
`-nodelete`	Emails are not deleted from the mailbox
`-log`	Writes a log to a specially created subdirectory

Example call:

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

7. Email archiving with the bitfarm-imap4 service

First, create a directory *called bfaimap4* in the bitfarm-Archiv directory and add all files from the directory ...*\bitfarm-Archiv\install\Bitfarm-Tools\bfa_imap4*.

To install the tool as a service, run an administrator console in the newly created directory. In this console, run the command `bfa_imap4_service.exe install`. The bitfarm service user is determined directly in the console. Enter the password for this user and the service will be installed under this user.

Note: If no bitfarm service has been installed yet, the user must be entered manually, as the user is determined based on the bitfarm services that are already installed.

After installing the service, you can proceed with the configuration. To do this, use the file `bfa_imap4_example.ini` and rename it to `bfa_imap4.ini`, as this is the only name under which it will be called in the service context.

The configuration file is divided into two sections. The `[main]` section defines the technical details:

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

`logfile` defines the path and name of the log file in which information about the program run is written, `loglevel` (`debug`, `info`, `error`) defines the logging level, and `logbackup` defines the retention period for the individual log files. The `service_interval` switch defines the query time in seconds for the installed imap4 service.

Please note: Errors related to the service are logged in the event log, while errors during execution are logged in the log file. For error analysis, we recommend using the log file, which contains detailed messages.

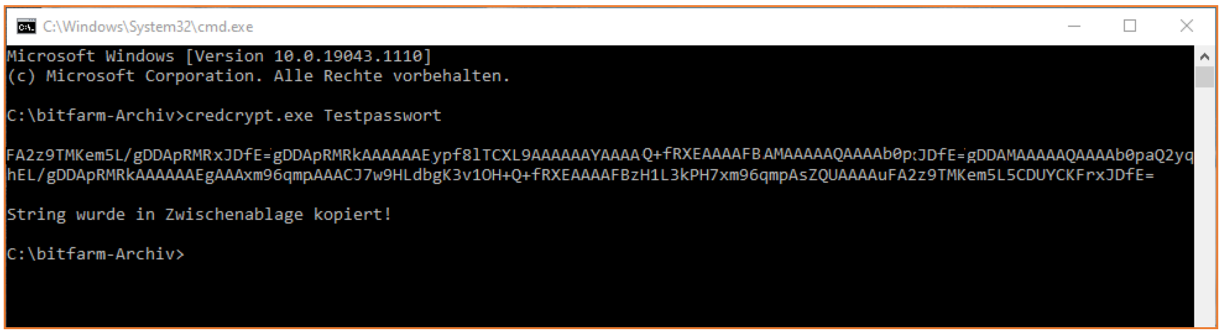
The second section defines the technical details of the mailbox, the storage location, and how to handle already archived emails and attachments. Multiple sections can be created, each monitoring a mailbox. Below is an example configuration and explanation:

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

First, define a section name in square brackets. This must be unique. Several subsequent sections can also be used for different mailboxes. Under `host`, enter the IMAP server on which the mailbox is to be monitored. The `port` command defines the port used. The default here is `993` or `143` for imap4 accounts, depending on the encryption method. This can also be defined in the following `encryption_method` switch. `ssltls`, `starttls`, and `none` are available. This information can be provided by the relevant mail provider.

This may also require a query as a defined user. You define this with `username` and with `password` the respective password. The `use_dpapi` switch can be used to decide whether the password should be stored in encrypted form in the configuration file.

Note: To encrypt the password, the `credcrypt.exe` tool from the bitfarm-Archiv directory must be used. Call this in a console with the term to be encrypted as the following parameter, and the output will be the encrypted term, as shown in the following figure:



```
Microsoft Windows [Version 10.0.19043.1110]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

C:\bitfarm-Archiv>credcrypt.exe Testpasswort

FA2z9TMKem5L/gDDApRMRxJDfE=gDDApRMRkAAAAAAEypf8lTCXL9AAAAAAYAAAAQ+fRXEAAAAFBAMAAAAAQAAAAb0p:JDfE=gDDAMAAAAAQAAAAb0paQ2yq
hEL/gDDApRMRkAAAAAAEgAAAxm96qmpAAACJ7w9HLdbgK3v1OH+Q+fRXEAAAAFBzH1L3kPH7xm96qmpAsZQUAAAAuFA2z9TMKem5L5CDUYCKFrxJDfE=

String wurde in Zwischenablage kopiert!

C:\bitfarm-Archiv>
```

The `fetch_method` switch defines whether all emails (`all`) or only unread emails (`unseen`) should be archived. With the value `unseen`, only unread emails are saved; emails that have already been saved are not retrieved again.

With `delete_after_fetch`, you can also specify whether the emails in the mailbox should be deleted after archiving in bitfarm (`true`) or remain in the mailbox (`false`). In addition, the `maildirs` switch can be used to decide which subdirectories of the mailbox should be monitored. Several directories can be queried, separated by commas. The notation must correspond to that on the mail server. The exact notation of the subdirectories can be found in the log after execution (with `loglevel=debug`):

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

Note: If the subdirectory contains special characters or spaces, the name must be enclosed in quotation marks.

Please note: If you have defined `fetch_method=all` as the retrieval method and do not delete the mailbox, it will re-archive ALL emails during each process.

To define where the emails are stored, the `savefolder` switch can be used. Here, a directory that is monitored by the bitfarm spool service and already exists must be selected. If the emails are to be stored directly in the subfolders defined in the `maildirs`, the switch `maildir_as_subfolder=true` must be set. If necessary, the subfolders in the defined `savefolder` are created automatically and the emails are placed directly in the corresponding folder.

Example:

```
savefolder=c:\bitfarm-archive\import\
maildir=inbox, "inbox.private invoices"
maildir_as_subfolder=true
```

In this case, two additional directories would be generated in the import folder, if not already present, with the names *inbox* and *inbox.private invoices*. The emails would be stored in the respective subdirectory.

Note: The subdirectories should be defined as templates for import into the bitfarm-Archiv DMS, otherwise the emails could end up in the undistributed directory.

The `python_plugin` switch can be used to call a plugin to further process the email before it is archived, for example, to separate the email from its attachments. To do this, the name of the plugin must be specified in the subdirectory of the same name provided for this purpose. A maximum of one plugin can be called in each section.

Note: The tool can also be called in a batch file or in the console. Simply run `bfa_imap4.exe` in the console. Information about the process will be entered in the console according to your defined `log level`.

Example plugin call by the imap4 service

The `extract_att` tool is used as an example of `a Python plugin` for the imap4 service. This allows you to extract attachments from the email and discard the email. It is located under the path *bfa_imap4\python_plugins\extract_att\*. This contains a configuration file that we can use to set up the plugin. First rename it to `extract_att.ini` so that the plugin can read it later.

Note: The file must always be named `extract_att.ini`. Also, make a backup of the sample configuration.

Note: You can also call the plugin multiple times if you give it a different name and thus map different cases. To do this, the folder must be copied and renamed accordingly. The configuration file must always have the same name in the subdirectory.

The configuration file can contain different, uniquely named sections. These are symbolized by square brackets. Here is an example of a section:

```
[Delivery_notes_Müller_GmbH]
att_regex =  .*delivery_note.pdf
sender_regex = .*@mueller-gmbh.com>
subject_regex = .*delivery note
importfolder = Delivery notes
continue_on_match=False
```

This section deals with delivery notes from Müller GmbH. The `att_regex` command searches for the name of the attachment using a regular expression. Here, the attachment must have the title `Delivery note.pdf`, or more precisely, end with `lieferschein.pdf`. Since the sender may vary, `sender_regex`, which checks the sender, only searches for the company name here; the specific sender can be variable. If the command remains empty, the sender is irrelevant and every incoming email is considered. The command `subject_regex` defines the subject of the email and has been set to `delivery note` here. If the word delivery note is in the subject line, this rule also applies. Since it must then clearly be a delivery note, the import folder has been defined more precisely here using the command `importfolder`, independent of the imap4 service folder that is executed. If no specific folder needs to be selected, it can also be filled with `NONE` and the default import folder of the imap4 service will be used.

Finally, the `continue_on_match` command can be used to decide whether the plugin should continue to run after the section or only end after the section if a match is found.

This allows you to define multiple sections for different documents.

Important: All regular expressions must match, otherwise the attachment will not be archived via the plugin. The three commands `att_regex`, `sender_regex`, and `subject_regex` can only match against the email header. Please note the correct spelling! The expressions are case-sensitive.

Here is another example:

```
[JPG Invoice]
att_regex =Invoice\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
```

If an invoice with the title "`Invoice123456.`jpg", where the numbers are random numbers, is sent from the address `...@bitfarm-archiv.de`, the attachment is archived. With `continue_on_match`, the following rule is also applied and every document with a PDF extension from the address `rechnungen@bitfarm-archiv.de` is archived in the same way. No special import directory was selected for either file type. Additional sections are also considered after the PDF rule, as `continue_on_match` is set to `True` here.

Note: The sender of an email is not just the name of the email address. In most cases, the sender looks more like this `=?iso-8553-1?Q?Peter_M=FCller?= p.mueller@bitfarm-archiv.de`. Keep this in mind when creating the regular expression in `sender_regex`.

Note: You can also use empty commands if, for example, the subject or sender is irrelevant and only the name or file type of the attachment is important.

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

a) Description

The bitfarm-Archiv SAP interface is a so-called content server (CS) in accordance with the interface description 'SAP Content Server http 4.5'. It has been implemented entirely in accordance with the interface description. The interface description can be accessed via this link:

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

Essentially, this interface is a web service (web server service) that can receive documents from SAP via the HTTP protocol or send them to SAP on request. To do this, SAP sends HTTP requests to the content server to transfer or request documents.

Only general, international standards are used when using this interface, such as the HTTP protocol 1.1 (RFC 2068), HTML (HyperText Markup Language), URL (Uniform Resource Locators, RFC 2396), UTC (Universal Time Coordinated), and digital encryption methods based on 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. The repositories are assigned a network address and port. This address defines a content server for SAP, in our case the bitfarm-Archiv Content Server (bfaSAP). In the bitfarm Content Server, an assignment of repositories to archives in the DMS is defined.

Communication via the interface can optionally be digitally signed. To do this, SAP signs all requests to the content server. The content server can use a public key transmitted by SAP to verify that the requests are authentic.

b) Archiving and provision of documents

When a document is created in SAP, SAP sends an HTTP request to the CS interface in the form of a URL. This URL contains the command, the necessary parameters, and, optionally, a signature. If, for example, a document is to be created, this is a create request with details of the repository, SAP's own document ID, access authorization, etc. The actual document, usually in the form of a PDF file, is then contained in the body of the HTTP request. This file is then stored by the content server in the archive storage in the audit-proof area and a database entry (document) is created in the bitfarm database. The SAP document ID is also stored in a separate table (sys_sap). Finally, the content server transmits an HTTP return or error code back to SAP.

If the document is now requested by SAP, SAP transmits a get request with the ID of the requested document. The corresponding document is determined in the bitfarm database using the SAP ID and transmitted to SAP in the body of the server response.

In addition, the CS processes further commands for updating, supplementing, searching, and deleting documents, as well as administrative commands for querying information about documents or the content server, or for transmitting certificates for the respective repositories.

c) Audit compliance

The bitfarm content server runs on the bitfarm-Archiv server as a Windows service. This service is started with the rights of the bitfarm-Archiv service user, a generic user account that is set up exclusively for bitfarm services. This service user is the only one with write access to the audit-proof area of the archive storage.

The audit-proof area is a directory structure managed by bitfarm-Archiv.

When a document is transferred from SAP to the content server via a create request, the server stores the document (the body of the HTTP request) immediately and directly in the audit-proof area. Once stored there, this document can no longer be changed, as no one except the service user has write or change rights to this area.

This document/file remains physically in this location, even if the document is assigned to another archive within the DMS, as this only happens via a corresponding assignment within the bitfarm database.

If SAP issues a delete request, the document is actually only marked as 'deleted' in the database. Here, too, the original file remains unchanged.

The history table in the bitfarm-Archiv DMS can be used to track what has happened to the document after archiving throughout its entire lifetime.

d) Technical requirements

The bitfarm-Archiv Content Server runs on Windows servers and has been tested with Server 2003, Server 2008 R2, and Server 2016. In theory, any platform that can run a Python interpreter is supported.

Python is the underlying programming language of the content server; the version used is Python 2.7.2 or 2.7.3.

The Python framework CherryPy 3.2 is used for web server functionality and must also be installed.

Signatures are verified using SAP certificates in signed mode (if configured) with OpenSSL version 1.0.1L, which must also be available.

e) Configuration

The bitfarm-Archiv Content Server requires two configuration files: `bfaSAP.conf` for specifying the network address and port required by the web server, and `bfaSAP.ini` for making general settings and assigning the repositories to the target archives in the DMS. Each SAP repository is specified with a two-letter identifier, and there are corresponding sections in the ini file for these identifiers where the assigned archive is stored. The structure of the two files is actually self-explanatory.

Note: The `arcid`, `tblname`, and `arcname` information can be found in the `sys_arc` table in the bitfarm database or in the archive templates created in *%bitfarm-archiv%\templates\*.

For example:

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

Example: bfaSAP.conf

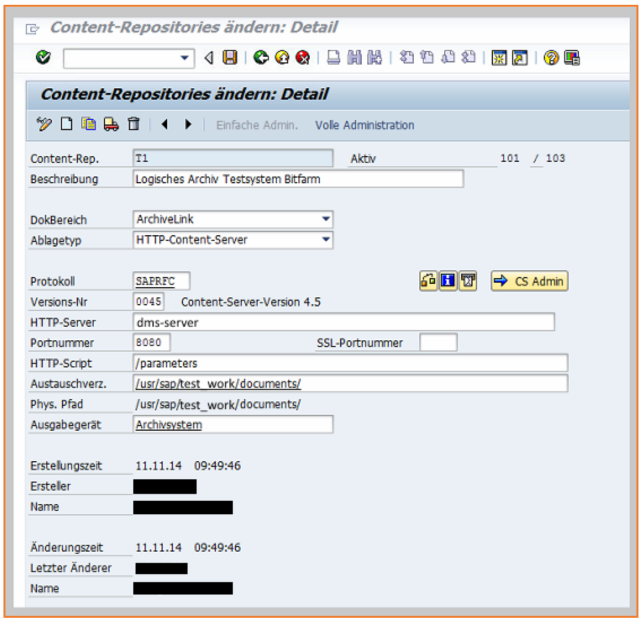
```
[SAP]
; specify multiple repositories with ; separated
repository=T1;T2
server=dms-server
signed=false
[bitfarm]
profile=demo
changeable=.doc;.xls
openssl=c:\OpenSSL\bin\openssl.exe
[T1]
arcid=68
tblname=09032010135935
arcname=Invoices_Credit_Notes_Sales
description=Invoices_Credit_Notes_Sales
[T2]
arcid=111
tblname=28042010161913
arcname=Invoices_Credit_Notes_ET
description=Invoices_Credit_Notes_ET
(…)
```

`Example: bfaSAP.ini`

On the SAP side, the bitfarm Content Server must then be stored in the configuration of the SAP repositories. To do this, create new repositories or change the repositories whose document types are to be archived in bitfarm-Archiv in the future.

(SAP transaction OAC0)

Basically, only the repository name, the HTTP server (name of the bitfarm server), the port number, and the HTTP script (/parameters) are relevant.



Content-Repositories ändern: Detail

Content-Rep.	T1	Aktiv	101 / 103
Beschreibung	Logisches Archiv Testsystem Bitfarm		
DokBereich	ArchiveLink		
Ablagetyp	HTTP-Content-Server		
Protokoll	SAPRFC		
Versions-Nr	0045	Content-Server-Version 4.5	
HTTP-Server	dms-server		
Portnummer	8080	SSL-Portnummer	
HTTP-Script	/parameters		
Austauschverz.	/usr/sap/test_work/documents/		
Phys. Pfad	/usr/sap/test_work/documents/		
Ausgabegerät	Archivsystem		
Erstellungszeit	11.11.14 09:49:46		
Ersteller	[redacted]		
Name	[redacted]		
Änderungszeit	11.11.14 09:49:46		
Letzter Änderer	[redacted]		
Name	[redacted]		

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 for bitfarm-Archiv DMS is available as standard in the documentation folder in the bitfarm program directory.

9. Data transfer in XML format, bitfarm XML Importer

a) General

The Bitfarm XML Importer is used to import documents whose metadata is stored in an XML file. In general, an import is started by calling the program as follows:

```
bfa_xmlimport.exe <path to configuration file> <xml file or path to the directory containing xml file(s)>
```

The configuration can be used to control which archive a document is imported into and how metadata is transferred to bitfarm. The configuration file (ini file) always contains a `[main]` section in which the following basic variables are defined.

- profile=The name of the bitfarm profile
- bfauser=bitfarm user used to log in to bfaServer 36 (encrypted)
- bfapass=Password of the bitfarm user (encrypted)
- rootnode=Root tag
  All XML nodes specified in the configuration use the node defined here as their origin. The node must be specified including the surrounding tags, e.g. <rootnode>.
- document=file or <name of the node in which the document is referenced>
  This controls how the document is found:
  - The document has the same file name as the imported XML file and is located in the same directory: document=file
  - The path to the document is specified in a node of the XML file: `document=`<name of the node>

    Note: The path can be an absolute path (with directory); if only the file name is specified, the document is expected to be in the same path as the XML file.
  - bitfarmpath=Path to the bitfarm-Archiv program directory (server)
  - templatepath=Path to the templates folder
  - uebergabe=Path to *bitfarm-Archiv\uebergabe*
  - logfile=Path to log file
  - loglevel=info or debug
  - deletexml=False or True

    After successful import, the XML file is deleted (True) or not deleted (False).
  - deletedoc=False or True

    After successful import, the document file is deleted (True) or not deleted (False).

b) Setting bfauser/bfapass

The bitfarm user to be used and their password are stored in encrypted form in the configuration. To do this, start a command console on the computer on which the import is to be carried out in the future, using the Windows user under which the import is to be started. Start the program `bfa_xmlimport.exe` in the console and enter the following parameters: `-cred <path to the configuration file>`.

You will now be prompted for the user name and password. Please enter both in the console window and confirm with "Enter." The user name and password will then be automatically saved in the configuration in `[main]`.

Please note: If the computer on which the import is performed changes, or if the Windows user for the import is changed, the process must be repeated to update the access data.

c) Sorting

In the "Archive" section of the configuration, conditions are defined that specify into which archive (name of the template) a document is imported. If none of the specified conditions apply, the xml file is ignored and the document is not imported. In each line below the "Archive" section, the base name of a template and the respective condition must be specified. For example, if a document is to be imported into the archive with the template name "Incoming invoices" when the node `<Header><DocType>` in the XML file has the value "Invoice," the following line is entered in the section:

Incoming invoices=(`<Header><DocType>=Invoice`)

Logical statements can be used in the condition, and the keywords "and," "or," "`and` not," "`or` not," and parentheses can be used to enable constructs such as ((`<Header><DocType>`=Invoice) and (`<Header><Subject>`=Renovation)). Each individual comparison, as well as the entire expression, should be placed in parentheses. The operators "=", "`<`", "`>`", "<=", "`>=`", "contains," and "`contains` not" can be used to compare values.

d) Metadata

For each specified archive (more precisely: template name), the following determines how which additional field and status field values are determined from the XML data. For additional fields, a new section "zus" <`name of the template`> is created for this purpose, and for status fields, a section "stat" <`name of the template`> is created.

e) Status fields

For the above example, let's assume that the "Incoming Invoices" archive (`IncomingInvoices.tpl`) has a status field called "Posted." "Posted" should be set if the node `<Header><Posted>` in the XML file has the value "Yes." The corresponding section would then look like this.

```
[stat_incoming_invoices]
posted=(<Header><posted>=Yes)
```

The same keywords and operators can be used for the condition as in the "Archives" section. If the specified condition evaluates to True, the status field is set to active, otherwise it is not.

f) Additional fields

Like the status field configuration, the configuration for additional fields is also done per archive in a separate section, whose options are generally structured as follows:

```
<Name of the bitfarm additional field>=<Value>
```

In the simplest case, <value> can be a specific node of the XML file. For the above example, let's assume that the "Incoming Invoices" archive has an additional field called "Invoice Amount." The field should receive the value from the XML node `<Header><InvoiceAmount>`. The corresponding section would then look like this:

```
[zus_incoming_invoices]
InvoiceAmount=<Header><InvoiceAmount>
```

The value can also be extended with any free text and/or composed of the values of several nodes. Let's assume the following XML structure:

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

The value for the additional field "Test" should now be set to "t1 - t2" for this xml. The configuration for this is as follows:

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

Nodes are always enclosed in "<" and ">" in the configuration. If two nodes appear directly after each other in the configuration, the second node is interpreted as a child element of the first node. If the nodes are separated from each other, they are treated as independent nodes on the same level. In the example XML, "test1" is therefore a child element of "subnode". If the following were stored in the configuration:

Test=<subnode> <test1>

the corresponding XML would look 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 configuring the additional field values, in addition to simply mapping an element value, you can also use a "List" statement or an "If" statement to read multiple nodes with the same name (List) or to determine a node value conditionally (If).

g) List statement for additional fields

An XML file may contain several nodes with the same name that contain different data. In order to determine from which node an additional field value should be retrieved, a condition must be defined that uniquely identifies the node.
For illustration purposes, let's assume the following example XML structure:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Indexing>
    <Header>
        <Address>
            <RecordType>Creator</RecordType>
            <AddressType>Employee</AddressType>
            <Name1>Homer Simpson</Name1>
            <Street>742 Evergreen Terrace</Street>
            <City>Springfield</City>
        </Address>
        <Address>
            <RecordType>Purchased</RecordType>
            <AddressType>Seller</AddressType>
            <Name1>Morticia A. Addams</Name1>
            <Name2/>
            <Street>1313 Mockingbird Lane</Street>
            <City>Mockingbird Heights</City>
        </Address>
        <Address>
            <Address Type>Recipient</Address Type>
            <AddressType>Customer</AddressType>
            <Name1>Wile E. Coyote</Name1>
            <Street>Desert Road 12</Street>
            <City>Desert</City>
        </Address>
    </Header>
</Indexing>
```

Here we see three nodes labeled "Address," whose data is to be mapped to an additional field. Let's assume that the target archive contains three additional fields, "Employee," "Salesperson," and "Customer," which are to be filled with the corresponding value of "Name - City" from the XML file. Expected values:

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

The general syntax of an if statement is:

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

The configuration for the example is then:

```
 Employee=If(<AddressType>=Employee, <Header>, <Address>) <Name1> - <City>
    Salesperson=If(<AddressType>=Salesperson, <Header>, <Address>) <Name1>
- <City>
    Customer=If(<AddressType>=Customer, <Header>, <Address>) <Name1> -
<City>
```

When specifying the "condition," you can again use the familiar keywords and operators.

If the "address" node only appears once in the example XML and we simply want to read a value under certain conditions, the `<startnode>` and `<iternode>` specifications can be omitted from the If statement. Only the condition is required. If this is evaluated as `true`, the specified value is read, otherwise it is not. However, when using this syntax, only the first node found is analyzed; other nodes with the same name are ignored.

h) Plugin configuration

A "Plugins" section can be created in the configuration file. For each archive (again, note that these are actually template names), the path to a plugin (executable file) can be specified here, which is started before the document + job file is transferred to the Bitfarm server. The generated job file is transferred to the plugin.

A useful application for a plugin would be, for example, when a MYSQL timestamp is read from an XML file and needs to be transferred to a Bitfarm date field. For successful import into Bitfarm, the date in the job must be specified in the format DD.MM.YYYY. A plugin could ensure that the MYSQL timestamp is converted into a suitable date before being transferred to Bitfarm.

II. Transfer of document information from the DMS to other systems

1. Job file

In `scripts.ini`, you can use
`autoexportpath=`
can be used to define a path for metadata export. For each new document that enters the DMS, the information written to the database is placed in the export folder in the form of a .job file. This information can then be used in other systems and enables, for example, the configuration of ERP/accounting systems as leading applications. The external system must also take care of cleaning up the read files. Here is an example of such an exported .job file:

```
[Version]
Version=36
[Archive]
Profile=sample-gmbh
Name=Accounts Payable
Table=28052015172108
Arcid=10
[Document]
DOCID=10.531
gdoc_id=531
Title=Invoice Ingram Micro-09586-11
Original=%bfastore0%\data-rs\2019\09\20\1992011957NBLK236825147.tif
Copy=
Source=
User=m.mustermann
userid=
Full text=%bfastore0%\data-rs\2019\09\20\1992011957NBLK236825147.txt
Status=0
StatStr=Document indexed and archived.
OCR_QF=-1
OCR_Type=OMP
Keywords=
Date=2019-09-20 11:09:34
Filter=
Pages=1
[Hash]
SHA1=
[Reference]
Count=0
[SVN]
```

```
SVN_Link=
SVN_Version=
SVN_Revision=
SVN_Date=
SVN_Info=
[Tasks]
Date=
Alarm=
Processed by=
Note=
TimerOptions=0
[Additionalfields]
Fields=17
AdditionalTitle1=Order-number
AdditionalField1=add_5
AdditionalValue1=09586-11
AdditionalTitle2=Invoice-number
AdditionalField2=add_3
AdditionalValue2=44-3281708
AdditionalTitle3=Invoice-date
AdditionalField3=add_2
AdditionalValue3=10/15/2019
AdditionalTitle4=InvoiceAmount
AdditionalField4=add_14
AdditionalValue4=622.37
AdditionalTitle5=Auditor
AdditionalField5=add_14
AdditionalValue5=m.mustermann
...
```

The gdocid may be of interest for external systems, as it allows the document to be accessed directly from external systems via the DMS client. The additional fields, which can also contain a barcode number, for example, can be used to link the document to external databases.

2. Excel export via configurable viewer plugin

Metadata contained in additional fields can generally be exported by right-clicking, e.g., to Excel or as a csv file. However, this always exports all additional field values and document-specific metadata, which you may not need for your export. With the Excel export viewer plugin, you can define exactly which additional field values should be exported.

a) Preparation

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

Important: To edit the configuration files, be sure to use an editor that supports UTF-8 encoding (not Windows Editor).

b) configuration

`Excel Export.vbs`

Note: Use ANSI encoding for `Excel-Export.vbs`

`exportpath="askuserfile"`

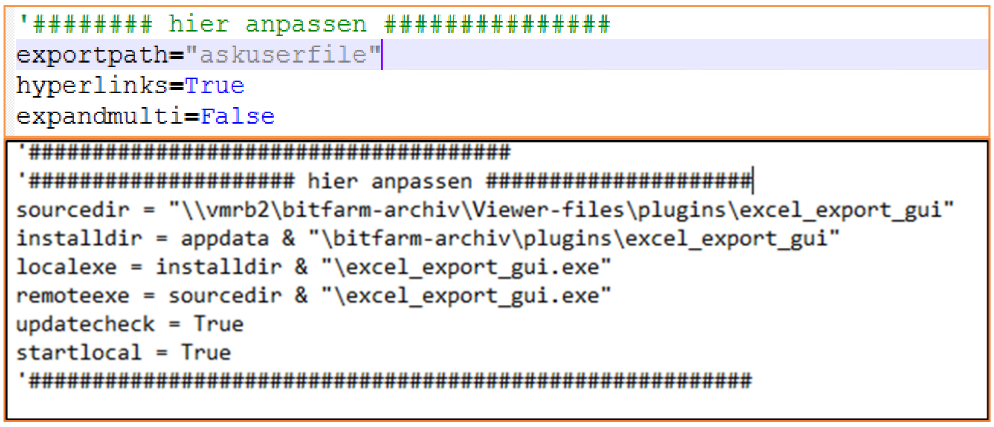
You can choose between three different options. With "askuser," the user is asked for a path to which the file should be exported. "askuserfile" asks for the path + the file extension. Here, it is important that the extension is also specified each time. Specify a UNC path and it will be saved directly to the specified path without prompting.

`hyperlinks=True`

Hyperlinks are used to go directly to the document in bitfarm DMS by clicking on the link. If this is set to `True`, the hyperlinks are also exported; if set to False, only the Gdoc IDs without links are included in the file.

`expandmulti=True`

If multiple selection fields are to be exported to individual columns, you can specify True here; if this is not desired, specify `False`.
In the area below, the paths of your environment must also be adjusted accordingly.

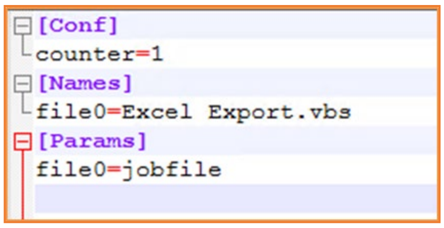


```
'######## hier anpassen ###############
exportpath="askuserfile"
hyperlinks=True
expandmulti=False
```

```
'########################################
'##################### 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
'########################################################
```

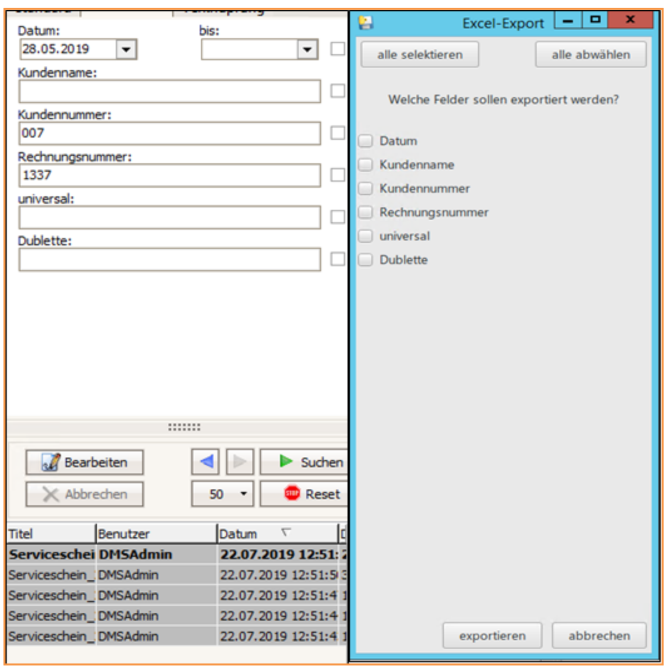
`Plugins.ini`

The `counter` must be adjusted in `plugins.ini`. This value is increased by one for each plugin.



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

The script to be executed is specified under `[Names]`, in our example `Excel Export.vbs`



Datum: 28.05.2019 bis:
Kundenname:
Kundennummer: 007
Rechnungsnummer: 1337
universal:
Dublette:

Bearbeiten | Suchen | Abbrechen | 50 | Reset

Titel	Benutzer	Datum
Serviceschei	DMSAdmin	22.07.2019 12:51:
Serviceschein_	DMSAdmin	22.07.2019 12:51:5
Serviceschein_	DMSAdmin	22.07.2019 12:51:4
Serviceschein_	DMSAdmin	22.07.2019 12:51:4
Serviceschein_	DMSAdmin	22.07.2019 12:51:4

Excel-Export

alle selektieren | alle abwählen

Welche Felder sollen exportiert werden?

- ☐ Datum
- ☐ Kundenname
- ☐ Kundennummer
- ☐ Rechnungsnummer
- ☐ universal
- ☐ Dublette

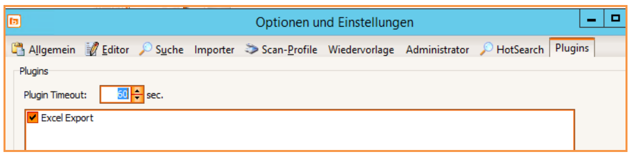
exportieren | abbrechen

Note: Enter the complete name of the file to be executed.

file0 specifies the first plugin. A new entry is created for each additional plugin, so the second plugin is called file1. Parameters are passed to the plugin in the [Params] section. In this case, the job file for the respective document on which the plugin is executed is passed to the plugin. You can find out which transfer parameters are available in the system manual or by contacting bitfarm-Archiv software support.

c) Execution of the plugin

If you have one or more documents in the hit list, you can select the plugin either by right-clicking on the context menu or via a corresponding button in the viewer. If you want a viewer plugin to be displayed as a "button" in the viewer, you must configure this in the viewer options.



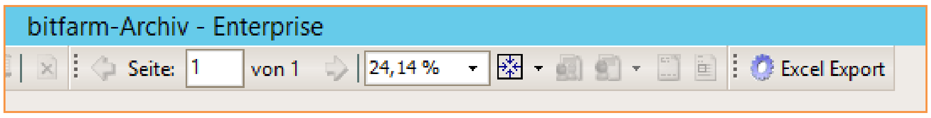
Optionen und Einstellungen

Allgemein | Editor | Suche | Importer | Scan-Profile | Wiedervorlage | Administrator | HotSearch | Plugins

Plugins

Plugin Timeout: 30 sec.

☑ Excel Export



bitfarm-Archiv - Enterprise

Seite: 1 von 1 | 24,14 % | Excel Export

The first time you start the plugin takes a little longer, as the plugin is installed on the client during the first start. Once it has started, you will see a selection window in which you can select and deselect additional fields.

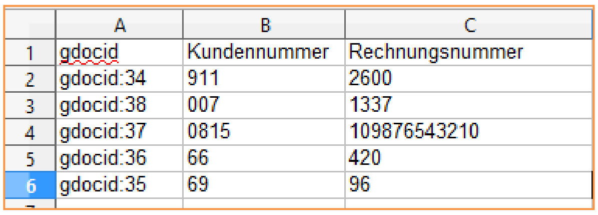
Once the additional fields have been selected, click on "Export." Depending on the configuration, you will see a new window in which you can select where the file should be saved. You can then open the Excel file containing the metadata you have selected and exported.

3. Advanced CSV export via configurable viewer plugin

There is often a need to make archived data and/or its metadata available to another program, e.g., financial accounting software. The desired metadata for the document can be exported as a .csv file using the bfa_csv_export tool, optionally with an image file and status setting.

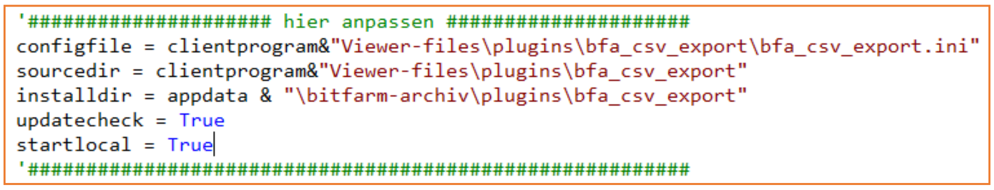
a) Preparation

Note: A compatible version of Bitfarm-Archiv DMS must be installed to set up the tool. If an outdated or incompatible version is installed on your system, please update it using your customer account on the bitfarm-Archiv website and the respective Enterprise version packages.



	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

The tool is located in the directory *%bitfarm-archiv%\install\Bitfarm-Tools\Viewer-Plugins\bfa_csv_export.*



```
'##################### 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
'#########################################################
```

Attention: To edit the configuration files of bfa_csv_export, be sure to use an editor that supports UTF-8 encoding (not Windows Editor).

Move the folder and the bfa_csv_export file to the directory *%bitfarm-archiv%\viewer-files\plugins.*

b) Configuration

In the `bfa_csv_export.vbs` file, the section marked "customize here" must be edited. Make sure that you use **ANSI encoding** for this file. If you have not made any changes to the paths, the setting can be retained.

If you have not added the folders to the *Viewer-files*\plugins directory as specified, you must specify the file path to `bfa_csv_export.ini` and the directory where `bfa_export.ini` is located after `configfile=` and `sourcedir=`.

The properties for the client are specified after `updatecheck=` and `startlocal=`. If the tool is to start locally on the client, `startlocal=True` must be entered. `updatecheck=` is used to set whether the tool is synchronized with the server version and updated accordingly when the server version is updated.

c) bfa_csv_export.ini

Caution: Only edit the INI file with UTF-8 encoding.

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

d) [main]

The `[main]` section forms the main part of the ini file. Use the `exportdir` variable to specify the directory in which the tool should save the documents. Specify a path that is accessible from the client.

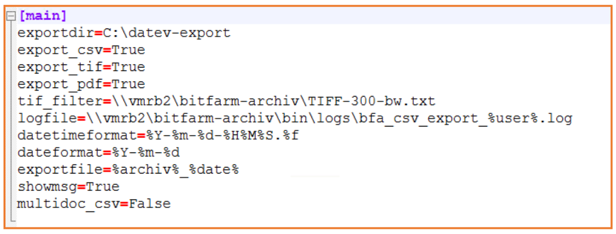
Use the three following switches `export_csv,` `export_tif`, and `export_pdf` to decide which file types you want to save.

With `tif_filter,` you can specify a policy for TIF documents. You will find several sample files (TIFF-100.txt, etc.) in your bitfarm-Archiv directory. You can also write your own filter policies. A description and the corresponding commands can be found in one of the existing filter files.

Logs written when the tool is executed are specified in `logfile`. Specify a path here where the logs should be saved. We recommend saving them on the server in the `Logs` directory. You can specify the user of the tool as a parameter in the file name, as shown in the example illustration. You can change date formats with `datetimeformat` and `dateformat`.

Note: The date format follows a strictly defined structure. You can read about this by looking up the Python function strftime. Do not change anything if you are not familiar with this function.

The files to be exported are named using `exportfile`. You can specify various parameters



```
[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
```

for the file name, such as the archive name, the date, and additional field information. Transfer the corresponding value using a % sign, for example `%archive%.` Other options include `%gdocid%,` `%date%,` `%datetime%`, and `%additional field name`%. If you want a short message to be displayed to the user in a message box after the export, set the `showmsg` switch to `True`. In some cases, it makes more sense to process a CSV file for different posting records instead of importing individual CSV files for each transaction. To ensure that the tool creates a CSV file for all documents selected (marked) in the viewer, set the `multidoc_csv` switch to `True`.

Note: For multidoc exports, `export_TIF` and `export_PDF` must be set to `False`.

e) [csv]

Attention: The various CSV parameters are defined in the `[csv]` section.

As a rule, no changes should be necessary here.

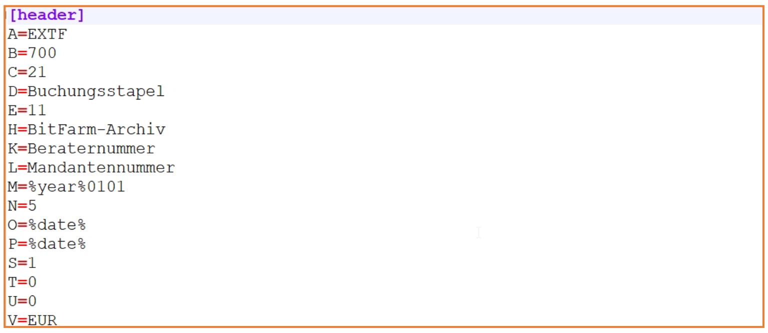
The `[csv]` section is used to adjust separators, escape characters, line endings, and quotes. `delimiter` describes the separator. Make sure that it is always enclosed in quotation marks. `doublequote` controls how an object is marked in the specified character (`quotechar`). If `doublequote` is set to `True`, the object is set in the characters set by `quotechar` with double quoting characters. If set to `False`, it is set in single quoting characters. If `doublequote` is set to `False`, an escapechar must be present. The variable `escapechar=` specifies which character should be used to interpret another character as a special character.

The input for the end of a line is defined in `lineterminator`. The default setting is \r\n. You can use the quotechar command to define the markup character. Always specify this in quotation marks `("'", "#", "")`. With quoting, you specify what should be set in the specified quoting characters. The options are `QUOTE_ALL`, which sets all objects in characters, `QUOTE_MINIMAL`, which sets objects with special chars in quoting characters, `QUOTE_NONNUMERIC` to convert all number types to a float, and `QUOTE_NONE`, which does not set any quoting characters and leads to error messages if no `escape character` is specified.

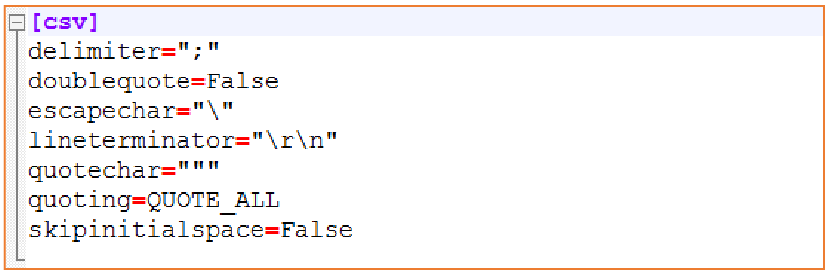
f) [header]

The `[header]` section contains the headers that are set in the CSV file.
The individual letters are used to specify the columns.

Note: If you want to create and generate the CSV file for the DATEV eG. financial accounting system, you need a header that is specified for the respective DATEV version and must be used for every export. Please consult your DATEV expert for more information.
Here is an example of a possible configuration

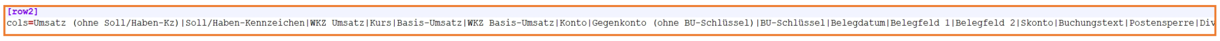


```
[header]
A=EXTF
B=700
C=21
D=Buchungsstapel
E=11
H=BitFarm-Archiv
K=Beraternummer
L=Mandantennummer
M=%year%0101
N=5
O=%date%
P=%date%
S=1
T=0
U=0
V=EUR
```



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

Additional headings can be created in the `[row2]` section. These ultimately represent the names of the database fields that you may have created as additional fields in the bitfarm-Archiv DMS. To ensure that the assignment works smoothly when importing the CSV file into your financial accounting system, be sure to use the names from your financial accounting system here. Below is an excerpt of typical DATEV designations for the `[row2]` section.

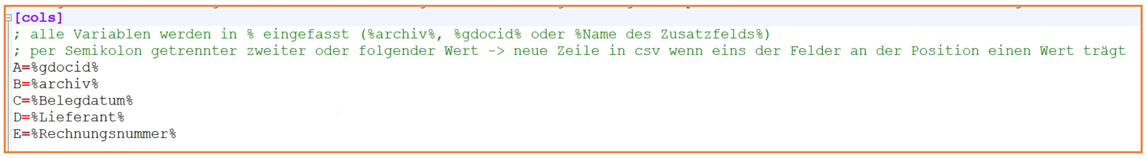


```
[row2]
cols=Umsatz (ohne Soll/Haben-Kz)|Soll/Haben-Kennzeichen|WKZ Umsatz|Kurs|Basis-Umsatz|WKZ Basis-Umsatz|Konto|Gegenkonto (ohne BU-Schlüssel)|BU-Schlüssel|Belegdatum|Belegfeld 1|Belegfeld 2|Skonto|Buchungstext|Postensperre|Div
```

The columns are no longer specified as "A, B, or C," but rather with a vertical line (pipe). Do not use `[row2]` to export the additional field values; this is done in the following section `[cols].` In this section, letters are again specified for the columns.

g) [cols]

The variables are set in percent signs. For example, to display the GDocID, `%gdocid%` is written in the field. Static text can also be written before and after the variables. For additional fields that are to be exported, each additional field must be set in percent signs.

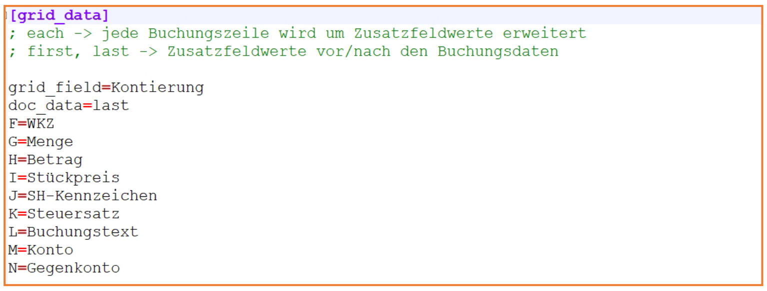


```
[cols]
; alle Variablen werden in % eingefasst (%archiv%, %gdocid% oder %Name des Zusatzfelds%)
; per Semikolon getrennter zweiter oder folgender Wert -> neue Zeile in csv wenn eins der Felder an der Position einen Wert trägt
A=%gdocid%
B=%archiv%
C=%Belegdatum%
D=%Lieferant%
E=%Rechnungsnummer%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

h) [grid_data]

If you also want to export metadata from a grid configured for the archive, in which you have entered your split entries line by line, for example, define the mapping of the corresponding values in the `[grid_data]` section.

Note: The creation of a grid is described in the system manual.



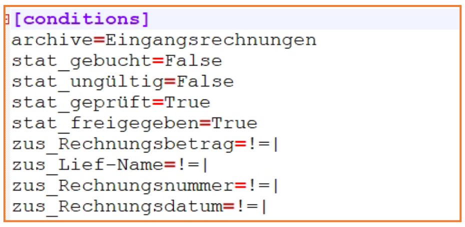
```
[grid_data]
; each -> jede Buchungszeile wird um Zusatzfeldwerte erweitert
; first, last -> Zusatzfeldwerte vor/nach den Buchungsdaten

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
```

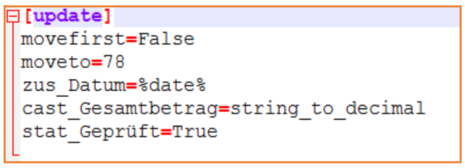
Enter the name of your grid after grid_field=. If you are unsure, you can check in the DMS administration. After doc_data=, you can decide whether the grid data should appear before or after the "normal" additional field data in the CSV file.

i) [conditions]

The [conditions] section is required to create conditions. These are then used to set the prerequisites under which the tool is allowed to export.



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



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

These variables can be used as follows:

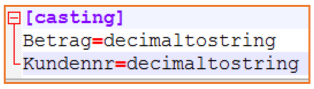
archive= specifies in which archives the tool may be executed. Multiple archives can be specified, separated by semicolons. If nothing is set for all conditions, an export may be started anywhere. To link the export to additional fields, you only need the name(s) of the additional field(s), which can be queried as follows: zus_Rechnungsbetrag=. The first equal sign refers to the variable. The operator after the first equal sign, !=, checks that the additional field is not empty. The vertical bar (pipe) serves as a separator between the operator and the value (in this case, an empty value). As soon as one of the above additional fields is empty, the export is prevented.

Status fields can also be queried. To do this, use stat_checked, where checked can be assigned to any status field name that is to be checked. Other operators for additional field checking are:

= which checks for equality, for example: zus_Kundennr==|57893
< smaller, which checks all values smaller than the specified value
> greater than, which only allows values greater than the specified value
~ contains a character string that must occur in this additional field, for example: zus_Kundennr=~|89. All documents containing 89 are exported, including customer number 57893.
.> begins with, every additional field that begins with, for example, 57
.< ends with, every additional field that ends with, for example, 93.

j) [update]

The [update] section can be used to update document metadata after export. If documents are to be moved before the additional and status fields are changed, set movefirst= to True; if this is not desired, use False. To select the archive to which the documents are to be moved with moveto=, you need the archive ID. This can be viewed in the administration area of the application. Enter the selected archive ID after the variable moveto=. You can use the following variables to change status and additional fields. After exporting, it is advantageous to set a status field that refers to exported documents, such as the status exported. Use the command stat_Exportiert= for this. All existing status fields can be changed by replacing Exportiert with the status field of your choice. This is done by passing True to it. You can also change additional fields after export. With zus_Datum=, for example, you can change the additional field Date, and with %date%, the date of the export day is passed to it.



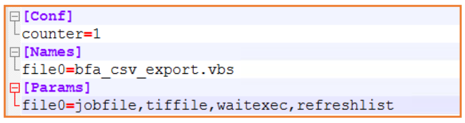
```
[casting]
Betrag=decimaltostring
Kundennr=decimaltostring
```

If you want to change values that need to be converted to a different format, use *the* appropriate *caster*. Make sure that you pass the correct function there. You can define your own casters for different treatments by adjusting the casting.py file accordingly. There are also some predefined functions here that you can use as a guide.

k) [casting]

The [casting] section changes the character format. A change can be made for each additional field that is specified. For example, amount values that are stored in bitfarm as decimal values can be converted to a string for export.

This is done using the example caster decimaltostring. Numeric values should always be converted to a string for export, as otherwise leading zeros may be deleted. This can lead to unwanted behavior and even errors.



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

The structure of the entries in the casting section must be as follows: Additional field name=caster variable.

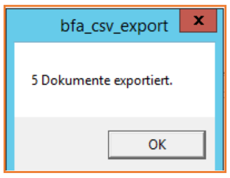
Tip: Provide the user group that is to create this export with a global bookmark that displays all documents that can be exported. Users can then run the viewer plugin for the configured CSV export on this hit list or even just on part of it.

l) Plugins.ini

Viewer plugins are connected via the file %bitfarm-Archiv%\Viewer files\plugins\plugins.ini. The counter may need to be adjusted in plugins.ini. It is increased by one for each tool to be executed.

The script to be executed (with extension) is specified under [Names], in our example bfa_csv_export.vbs. Enter the complete name of the file to be executed here. file0 specifies the first tool; a new entry is created for each additional tool, so the second tool is called file1. Parameters are passed to the plugin in the [Params] section. jobfile passes the metadata of the document to the plugin. `tiffile` specifies that the TIF should be passed to the plugin. `waitexec` pauses the viewer as soon as the export has started. This prevents corrupt data. `refreshlist` reloads the archive as soon as the tool has been executed. If the status and additional fields are changed, these are immediately visible after export. The parameter `tiffile` is considered mandatory. The other transfer parameters and their application are described in this chapter. Viewer-Pl s and gins

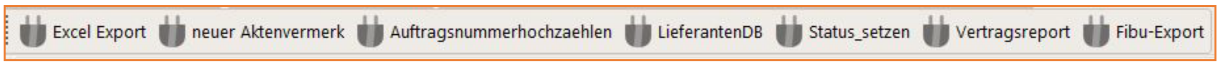
m) Viewer configuration


bfa_csv_export

5 Dokumente exportiert.

OK

Start the viewer and open the options (File -> Options). In the Options window, select the Plugins tab. Here, you can add the tool to the quick



selection by placing a check mark in front of the tool name.

Alternatively, you can run the tool by right-clicking on the selected document under Plugins.

Select one or more documents in an archive and start the tool. After successful export, you should see the following image, provided that the appropriate settings are made in `bfa_csv_export.ini`.

III. Transfer of metadata from external databases

4. bfa_sqlmapper

a) Scope

Usually, the metadata of a document is entered by users or extracted from the full text using indexing rules (wfd file). However, metadata that is not directly available in the document but is stored in external databases is often also of interest. The SQL Mapper can be used to query external databases via ODBC and save the values obtained to the document.

For the most flexible connection possible, the SQL Mapper can be connected in three different ways.

- Connected via a new "sqlmapper" statement in wfd-naming sections
- Started by the end user as a plugin from within the viewer
- As a standalone program, e.g., manually or repeatedly via a scheduled task

b) Application examples

- A customer number is read from a document using a wfd rule. The customer's name is not available in the document; it must be determined from an existing database in which the name is stored for each customer number.

- Quotes are archived in bitfarm-Archiv, and the corresponding quote number is read from the full text and stored in an additional field. In addition to quotes, orders are also archived in the DMS, which are assigned an order number that is also stored in an additional field. In addition to the order number, a field is now to be filled in with the quote number associated with the order. This information is stored in an external database.

c) Settings/Configuration

The sql-mapper is configured via an ini file (by default in *%bitfarm-archiv%\bfa_sqlmapper.ini*). When editing the configuration file, care must be taken to use a suitable editor so as not to change the encoding of the file (UTF-8). Notepad, for example, is known to change the encoding when saving. We recommend using a modern text editor such as Notepad++. In addition to general settings such as logging and connection data for the external data source, the file also stores the corresponding SQL queries and mappings to bitfarm-Archiv. The SQL mapper also requires access data to log in to the bitfarm server.

The following two configuration sections are always required.

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

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

In the `[main]` section, the profile name and the login credentials for the bitfarm server service are stored. The login credentials are stored in encrypted form. To encrypt the data, use the command line tool `credcrypt.exe`, which is located in *the %bitfarm-archiv%* directory on the bitfarm server. Log in to the server with the user under which the bitfarm services are running and open a normal console in *the %bitfarm-archiv%* folder. Enter `credcrypt.exe`, followed by the user name with which bfa_sqlmapper should log in (e.g., DMSAdmin). The encrypted user name is displayed on the console and also saved to the clipboard. Transfer it to the configuration after `bfauser=`. In the same way, generate the encrypted password for the user and enter it after `bfapass=`.

Please note: The Windows Data Protection API is used for encryption and decryption (for more information, see https://msdn.microsoft.com/ e n-us/library/ms995355.aspx). If the bitfarm service user is changed or the server is moved to new hardware, bfauser and bfapass must be re-encrypted, even if neither the user nor the password has changed!

The following options can be set in the `[logging]` section:

- `viewerplugin/standard/standalone`: An individual path for the log file can be specified for each context. Please note that the path to the log file in the viewerplugin context is a path from the client's perspective!
- `level`: This can be used to control how detailed the logging should be. Possible values here are "DEBUG" (detailed) and "INFO" (less detailed).
- `clientlog`: Controls whether information about communication between sql-mapper and bfaServer should also be logged.
- `backupcount`: One log file is created per day. backupcount can be used to control how many log files should be kept as backups.

User-specific configurations are made in further sections. Example configuration (minimal):

```
[hole-customer name]
constring=dsn=kundendb-odbc
sql=select customer name from customerdb where customer number=%customer num-
ber%
Customer name=%0%
overwrite_customer name=True
emptyonmissing=True
emptyonnoresult=True
```

The name of the section can be chosen freely, but must be unique within the configuration.
First, the ODBC data source must be stored in the operating system. The configuration interface for this can be started in Windows using the command
%windir%\syswow64\odbcad32.exe (be sure to start the 32-bit version, even on 64-bit operating systems).
In the configuration dialog, switch to the "System DSN" tab and click "Add." In the following dialog, you must first select the driver (if necessary, this must be installed afterwards; if in doubt, ask the manufacturer of the external database). In the following dialog, further information for ODBC access is stored in addition to the data source name. Click on "Test" to make sure that the configuration is correct and that a connection is possible.
The reference to the newly established ODBC connection must now be stored as a connection string in the "constring" option. What this should look like depends on the type/driver of the data source; if necessary, this information can be obtained from the provider. In many cases, it is sufficient to specify the dsn in the form: constring=dsn=<Data Source Name>. Further information can be found, for example, at https://www.connectionstrings.com.
Once the ODBC connection has been set up and the basic configuration has been created, the sql-mapper can test the accessibility of the external database. To do this, open a console on the bitfarm server in the %bitfarm-archiv% folder and enter the following command:

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

The parameter `-s` all ensures that the connection test is performed for all sections found in the configuration, but it is also possible to specify the name of a single section to be tested

instead of "all". If the connection test is unsuccessful, the error that occurred is displayed. Next, an SQL query is required to provide the desired information. The value to be searched for comes from an additional field. The name of the additional field is enclosed in "%" characters in the query, so in the above example, the search is performed using the value in the additional field "Customer number".
Next, the mapping of the query result to the corresponding additional fields is configured. To do this, select the name of the additional field as the option and use the column index enclosed in "%" characters from the query result as the value. In the above example, the result will only contain a single column (the one with the customer name), so "%0%" (index starts at 0!) is used here. In addition to the index, fixed character strings can also be specified as values. This makes it possible to supplement the value from the database with additional fixed information. If ---%0%--- were specified for the customer name in the above example and the query returned "bitfarm-Archiv GmbH" as the customer name, the stored additional field value would be "---bitfarm-Archiv GmbH---".

The two switches `emptyonmissing` and `emptyonnoresult` control whether the specified target fields should be emptied in the event of an empty search value (`emptyonmissing`) or in the event of an empty query result (`emptyonnoresult`). By default, both switches are set to `False`.

If the target field already contains a value when the process starts, this value is not overwritten by default. If this is desired, an `overwrite_<field name>=True` option can be activated accordingly (in the above example, `overwrite_customer name=True`).
This completes the minimum configuration for the SQL mapper. The section "Advanced configuration" describes additional configuration switches that can be used to cover other special use cases.

d) Connecting the SQL mapper via wfd rule

To enable metadata matching when archiving a document, the SQL mapper must be connected in the WFD file in a `naming section`. In this case, please note that the value to be searched for in the external database (in our example, the customer number) must already be available at the time of archiving, i.e., the customer number must already have been entered during import or can be determined by the archiving naming rule.

In the `naming section`, you can specify conditions for executing the section (e.g., `archivtabelle=incoming invoices`, etc.), after which only the section from the sql-mapper configuration that is to be executed in this context is specified:

```
naming section crm-customers
archive table=customers
sqlmapper=get-customer-name
end section
```

If several sections of the sql-mapper configuration are to be used, this can be done by using multiple sql-mapper statements within the naming section.

Note: When processing wfd rules, sql-mapper statements are only applied after all other sorting and naming rules have been executed so that it is possible to read the lookup value via wfd as well.
If scriptdebug is enabled in `archivierung.vbs`, the call to the sql-mapper and the result of the execution are logged in the event log. Of course, you can also find the log messages in the configured log file of the sql-mapper.

e) Connection of the SQL mapper as a viewer plugin

In order for the sql-mapper to run on the client, the ODBC connection must first be set up on the relevant client as described above. Then make sure that you edit the `plugins.ini` file on the server in the *%bitfarm-archiv%\viewer-files\plugins* folder (see the system manual section "Plugin options"). Enter the script `bfa_sqlmapper.vbs` in `plugins.ini` and use "jobfile", "waitexec" and "refreshdocument" as parameters. As with any other viewer plugin, the viewer must be restarted after connecting the sql-mapper. Test whether the sql-mapper is set up correctly by running the plugin once. If the target field does not contain the expected value after execution, you should look for troubleshooting information in the sql-mapper log.

f) Advanced viewer plugin configuration

By default, all sections from `bfa_sqlmapper.ini` are executed when running as a viewer plugin. If this is not desired, but only a specific section is to be evaluated, for example, this can be done by adjusting `bfa_sqlmapper.vbs` (replace the line section="all" with section="hole-customer names", for example).

Alternatively, conditions can also be formulated in `bfa_sqlmapper.ini` for each section, which are then evaluated by the sql-mapper. The section is only executed if all specified conditions are met. For example, if our section [hole-customer names] is only to be evaluated if the document is one from the "Customer invoices" archive, a new section called `[vp_cond_hole-customer name]` can be added to `bfa_sqlmapper.ini`. (vp_cond_<name of section>). The conditions can now be formulated within this section. In our example, it would look like this:

```
[vp_cond_hole-customer name]

archive=Customer invoices
```

Please note: [vp_cond_<name of section>] is only evaluated if the sql-mapper is executed as a viewer plugin.

In addition to the archive, additional or status field values can also be specified as conditions for evaluating the section. Additional field options always begin with `add_` followed by the name of the additional field; status fields begin with `sts_` followed by the status field name. After the equal sign, an operator is now stored for additional fields. (Possible values: "=" (equal), "<" (less than), ">" (greater than), "!=" (not equal), "~" (contains), ">." (begins with), ".<" (ends with). The operator is always followed by a pipe (|), which separates the operator and the following comparison value. There is no operator in the notation for status field conditions; here, you can work with `sts_<name of status field>=True` or `sts_<name of status field>=False`.

In our example, if the sql-mapper as a viewer plugin should only execute the section [get-customer-name] if

1. The document in the customer invoices archive
2. the customer number always begins with a "1"
3. the status "posted!" is set for the document.

the complete condition section looks like this.

```
[vp_cond_hole-customer names]
archive=Customer invoices
add_customer_number=.>|1
sts_posted=True
```

g) sql-mapper as a standalone tool

Under certain circumstances, it may make sense to start the sql-mapper at specific intervals and run the metadata comparison for several documents in the background. When executed as a viewer plugin, the sql-mapper is applied to the documents selected by the user; when connected via the wfd file, it is applied to the document currently being archived. To specify in the standalone context which documents the sql-mapper should compare, use the bitfarm user that was entered in encrypted form in the configuration to create a bookmark that delivers exactly the desired documents. In our example, a bookmark could be created that finds all documents for which the customer name is still empty. The name of the bookmark is then specified in the configuration section as bookmark. If the bookmark is called "without-customer-name," enter bookmark=hole-customer-name in the `[hole-customer-name]` section:

```
[get-customer-name]
constring=dsn=kundendb-odbc
bookmark=no-customer-name
sql=select customer name from customerdb where customer number=%customer num-
ber%
CustomerName=%0%
overwrite_customer_name=True
emptyonmissing=True
emptyonnoresult=True
```

To run the sql mapper, open a console on the bitfarm server in the *%bitfarm-archive%* folder. Then enter the following command:

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

The parameter `-c standalone` specifies that the sql-mapper should be executed in standalone mode, while the parameter `-s hole-customer-name` selects the section [hole-customer-name]. Any problems that occur during execution are displayed in the console and written to the log file configured in the logging section under standalone.

h) Advanced configuration

1. indexlookup

In our example, we search for the customer's name using the customer number recorded for the document. If necessary, several customer numbers may have been recorded for the document, which are then stored in a universal field, separated by semicolons. In this case, the customer number used for the lookup must be stored in the sql-mapper configuration. Here, you have the option of specifying the index of the value to be used (starting at 0!). Add the entry "lookupindex=0" to the corresponding section to always use the first customer number from the additional field. With "lookupindex=1", the second number specified would always be used, etc. If all customer numbers are to be used for the lookup, enter "lookupindex=all" in the section. In this case, the result of the SQL query would return multiple rows of results. Therefore, the additional fields to be filled must be of the type Universal, Selection, or Multiple Selection, as only these fields can contain more than a single value.

2. inputcasting/outputcasting

If SQL errors are displayed/logged when executing the SQL mapper, this may be due to incompatible data types. For example, if the lookup value is determined from a universal field, it is of the unicode string type. However, it may be that the lookup value actually consists only of digits and is stored as an integer value in the external database. If the SQL statement is executed, this results in an error because the data type is incompatible. The additional field value must be converted to an integer before the query can be executed. An inputcaster can be entered in the configuration for this purpose. All casting functions are declared in the `bfa_sqlmapper_casting.py` file. This script already contains several functions in its delivery state, but the script can be expanded with individual casting functions as desired. Each casting function is called with two arguments: first, the value from the bitfarm additional field and, second, the context in which the sql-mapper is executed. The function must be written in such a way that it converts the additional field value to the appropriate data type for the target database and returns it.

For the example given (lookup value must be converted from string to integer), the casting function "toint" can be used, which can be stored in the configuration section as "inputcasting=toint".

If multiple lookup values are used in the query, the casting functions are specified separated by semicolons in the order of the lookup values from the sql statement. For values that do not require conversion, the casting function "pipe" can be used to use the value unchanged in the query.

If the query can be executed without errors and returns a result, it is possible that the result values of the query now also require a type conversion for transfer to a Bitfarm additional field. It is conceivable that the query returns an integer, for example, and the result is to be written to a universal field. Since the sql-mapper knows the data type of the value from the external database and can also determine the target data type, it is usually not necessary to specify a casting function here. However, if the automatic conversion fails, it is also possible to write a function in bfa_sqlmapper_casting.py that performs the conversion correctly. The function is then stored in the section as outputcasting. If more than one column is queried in the query, it is necessary to specify a casting function for each column (in the order of the selected columns from the query). If some columns are to be converted with their own function, but other columns are to be converted automatically, "auto" can be used as the casting function for the latter.

5. add_sync_values

a. Scope

You want to make values available to users in an additional selection field, but you don't want to enter these values manually; instead, you want to compare the possible selection values with, for example, a corresponding table in your ERP database.

b. Application examples

Often, it is supplier names, cost centers, the names of your auditors, construction managers, etc. that are to be made available to users for selection and are already available in one of your databases.

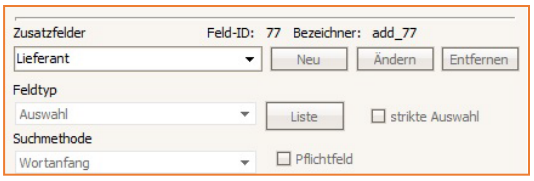
c. Settings/configuration

As with bfa_sqlmapper, configuration is done via an ini file (by default in *%bitfarm-archive%\ Bitfarm-Tools\add_sync_values\ add_sync_values.ini*). When editing the configuration file, care must be taken to use a suitable editor so as not to change the encoding of the file (UTF-8). Notepad, for example, is known to change the encoding when saving. We recommend using a modern text editor such as Notepad++.

As with the configuration of *bfa_sqlmappers*, the login data is encrypted using the command line tool credcrypt.exe, which is located on the bitfarm server in the directory %bitfarm-archiv%. Log in to the server with the user under which the bitfarm services are running and open a normal console in the folder *%bitfarm-archiv%.* Enter `credcrypt.exe`, followed by the username with which the `add_sync_values.exe` tool should log in (e.g., *DMSAdmin*). The encrypted user name is displayed on the console and also saved to the clipboard. Transfer it to the configuration after `user=`. In the same way, generate the encrypted password for 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-archive\Sample-Inc.con
```

```
[add_77]
src=odbc
constring=DSN=lfodbc
sql=SELECT DISTINCT lfname from suppliers
action=add_values
;action=add_values→ old values are retained, 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 additional field to be filled in the bitfarm-Archiv DMS is specified in square brackets, using the notation above. You can find out the ID of the additional field you want to fill in the administrator, among other places, by logging in to the viewer as *dmsadmin*.

Enter the name of your configured 32-bit ODBC connection after `constring=DSN=`. You may need to store a connection string for your specific database here. The format of this string depends on the type/driver of the data source; if necessary, this information can be obtained from the provider. In many configurations, it is sufficient to specify the dsn in the form: constring=dsn=<Data Source Name>. Further information can be found, for example, at https://www.connectionstrings.com

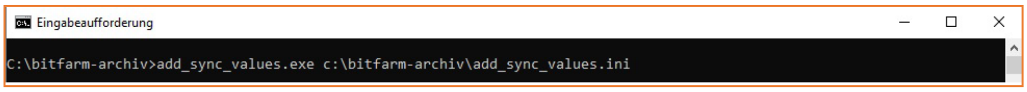
For `sql=`, enter the SELECT that must be executed on the database to be queried in order to obtain the values to be transferred to the additional selection field.

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

Note: If you cannot use an ODBC connection, you also have the option of generating a .jason file with the values to be imported, reading the output either to stdout or saving it in a .jason file and then reading it. A corresponding example section has been created in *add_snc_values.ini* for this case.

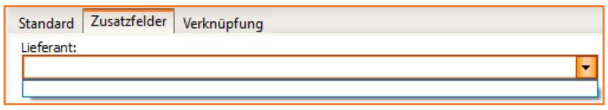
d. Execution

Link the `add.sync.values.exe` file via Windows Task Scheduler and enter the path to the ini file under Arguments to perform a regular comparison between your database table and the additional field in bitfarm. If you want to perform this comparison once, you can also do so via the Windows command prompt.



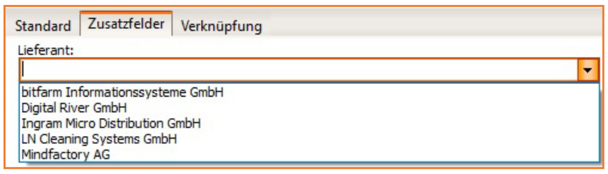
```
C:\bitfarm-archiv>add_sync_values.exe c:\bitfarm-archiv\add_sync_values.ini
```

Once the synchronization has been performed in one of these ways, you can view the result in the corresponding additional field in the bitfarm-Archiv Viewer.


Standard | Zusatzfelder | Verknüpfung

Lieferant:

The selection additional field Supplier (add_77 from the example) before the synchronization.


Standard | Zusatzfelder | Verknüpfung

Lieferant:

bitfarm Informationssysteme GmbH
Digital River GmbH
Ingram Micro Distribution GmbH
LN Cleaning Systems GmbH
Mindfactory AG

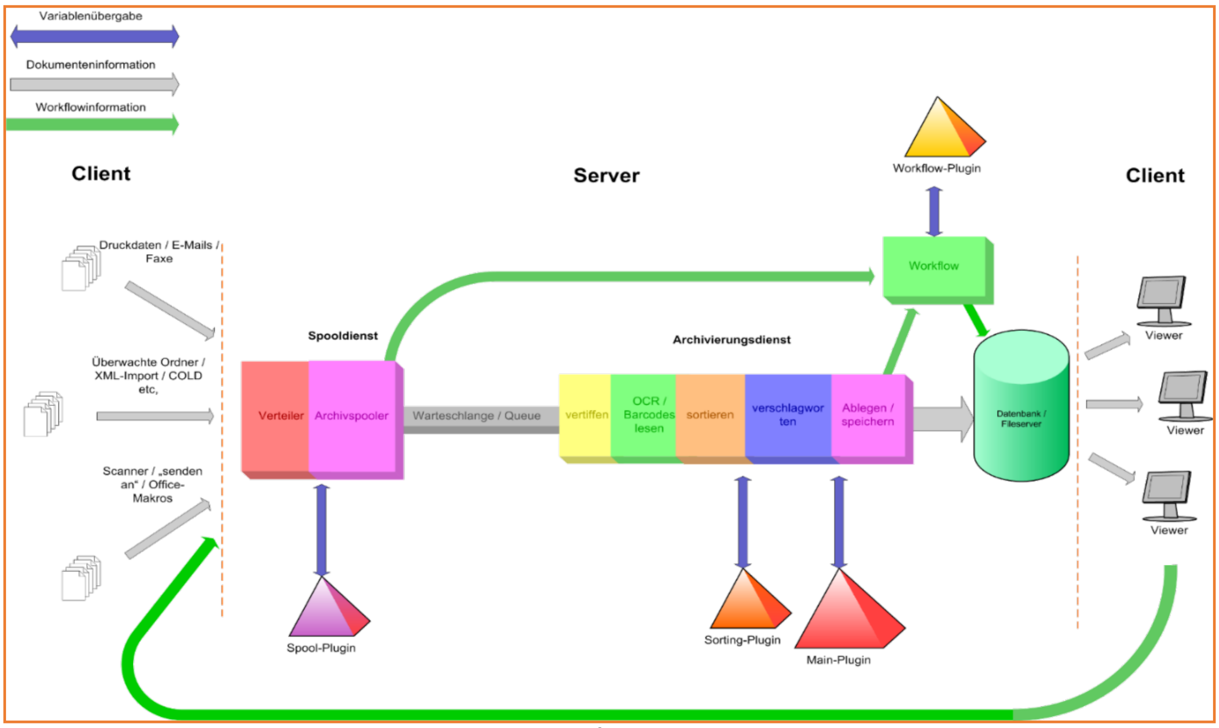
The selection additional field Supplier (add_77 from the example) after the synchronization

IV. Individual controls with the help of plugins

6. Server plugins

At various stages of a document's passage through the archiving or workflow process, there are options available to influence the processes, change variables and field contents, and/or trigger certain actions using individually customized programs (plugins). Plugins can be any executable programs. Information is exchanged with the bitfarm-Archiv server via a file that is passed as a parameter at startup. This file is a text file that passes all available variables in the form of `variable name=value`. The text file is read back into the server after the plugin has finished. Examples for the individual plugins are available in Visual Basic Script under *..\bitfarm-archiv\plugins\*.

The following illustration shows the possible plugins and their position in the document processing timeline.



7. Viewer plugins

Viewer plugins are stored in *...\bitfarm-archiv\Viewer-Files\Plugins* and entered there in `Plugins.ini`. The number of plugins is entered in the `[Conf]` section, the file names under `[Name]` and the options under `[Params]`. The plugins are displayed in the viewer with their base name (file name without extension).

When a plugin is launched in the viewer, the plugin file is executed and the path to the temporary text file is transferred to it. This file contains the path to the job file belonging to the document and, if applicable, the preview TIF file. These files are made available for the plugin in the *%temp%\Viewer* or *%temp%\Viewer\tif* directory.

Viewer plugins are started either manually by right-clicking or via the corresponding button in the viewer interface. If you want to start viewer plugins via "Buttons" in the viewer, you must first display them. This can be configured as desired in the viewer options.

The possible options for a plugin:

`jobfile`	A job file is exported
`tiffile`	The preview tiff is exported
`waitexec`	The viewer waits until the plugin has finished
`refreshlist`	The hit list is updated (search is performed again) (also: refreshliste)
`refreshresub`	The resubmission is updated
`resync`	The job file is imported, the document is updated with the imported 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
```

For a save or move plugin, the base name of the plugin must be specified under `[Conf]` and the "Plugin active" checkbox must be selected in the bitfarm-Archiv administrator for the desired archives.

The plugins are then executed in these archives every time a save or move operation is performed. The save plugin is executed at the end of the entire save process, so newly entered additional field values or status fields are already available. The move plugin is executed immediately before the move; a return code = 255 aborts the move operation. For performance reasons, only the job file is available for the save or move plugin.

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

For plugins that are executed with the `waitexec` or `resync` option, you can specify a `plugintimeout=10` (in seconds) in the con file under `[Options]`.

8. Client control by leading applications

The bitfam archive viewer is the central client program of the DMS for displaying and handling documents. Its functionality goes far beyond simple display and enables the handling of electronic notes, database information, workflows and distribution, research, and much more. The viewer not only allows access to the actual document, but also provides all additional information and functions. When connecting the DMS to other systems, it therefore makes sense to use the bitfarm-Archiv Client directly for document display rather than a simple TIF viewer. The interfaces available here are described below.

Tip: In many cases, a second monitor is useful for document display. This operating mode is therefore also supported by the bitfarm-Archiv Viewer.

9. Hotsearch functions

The Hotsearch tool consists of a file `called Hotsearch.exe`, which should be started on the client systems via autostart. Using the key combination *Ctrl-C* and then *Ctrl-B*, text from any Windows application can be moved to the viewer as search text, which then immediately performs a full-text search in the last selected archive or storage location. The only important thing is that the Windows application allows information to be transferred using *Ctrl-C*. For example, the user can select the content of the "Invoice number" field in their ERP application by double-clicking and then, after *pressing Ctrl-C and Ctrl-B*, the invoice is immediately displayed on the screen.

10. Direct access to a document via the GDocID

Documents in the DMS database have the so-called *GDocID* as a global primary key. The *GDocID* is also supplied when exporting metadata, thus providing external systems with the option of saving a specific reference to a document. The call is then simply made as a single parameter via the command line.

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

If the viewer is already open, the display is updated accordingly. If no viewer is open, the user is prompted to log in to the system and the document is then displayed.

Note: To be able to display a document via the external call of Gdocid, the user must have search rights in the corresponding archive.

11. Programmatic transfer of search terms

Other parameters can be used to trigger a keyword search or full-text search in a specific archive:

```
Viewerv3.exe -L:[Storage] -lid:[ID of the desired storage] -A:[Archive] -aid:[ID of the desired archive] -S:[Keywords] -V:[Full text]
-R:[Link] -GO -W: -C: -G:[gdocid] docid:[arc_id.doc_id] gdocid:[gdoc_id] [arc_id].[doc_id]
-ADDITIONAL:[additional field name]:[additional field value]
```

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

```
viewerv3.exe -L:Purchasing -A:Delivery notes -S:4711 -GO
```

Example: Search in the viewer for additional field title with the value "Document" with Follow-up open

```
viewerv3.exe -W: -ZUS:Title:Document
```

Tip: The parameterized call can also be performed via the hotse a rch.exe. Since this is smaller, the call is faster.

The parameters in detail

`-L:[Storage]` Switches to the specified warehouse
`-lid:[Unique ID of the warehouse]`
`-A:[Archive]` switches to the specified archive. The archive must be located below the
current storage location if a call is made with -A: only. If you want to switch to an archive below another storage location, this must be specified beforehand with -L:.

`-aid:[Unique ID of the archive]`
`-S:[Keyword]` fills the keyword field for a search
`-V:[Full text]` fills the full-text search with one or more terms
`-R:[Link]` fills the link field; the link search cannot be combined with a full-text search or keyword search.

`-GO`	Performs the search using keyword, full text, or link.
`-W:`	Opens the follow-up list
`-P`	Prints the currently displayed document
`-C:`	Closes the viewer
`-G:[GocID]`	Opens the specified document via the GdocID

or `gdocid:[GdocID]`
`-ADDITIONAL:[Additional field name]:[Additional field value]`
Performs an archive-wide search in the additional field, i.e., all archives in which the field

occurs are searched. The search is performed by word fragment (not exact) and no range searches can be performed. If a warehouse or archive is specified beforehand (with -L: or -A:), the search is restricted to this warehouse or archive. If you want to search for additional fields that have a different data type than universal fields, e.g., date, floating point, or currency fields, the search term must be formatted as the data type is stored in the database tables, i.e., for a date search, the date must be specified in the MySQL date format: (YYYY-MM-DD), e.g. -ZUS:Date:2019-10-12
For numbers with decimal places, the comma must be replaced by a period, e.g. -ZUS:Amount:219.77

a. If search authorization has been revoked on the archive side

The *programmatic transfer of search terms* described above always requires that the users making this call also have search authorization for the archive where the document is located.

In practice, however, it is often the case that users are only allowed to see certain documents in an archive.

Example: Lischen Müller is only allowed to see the invoices assigned to her in *the* "Creditors" *archive*, e.g., only the invoices where her name appears in an *additional field* called "Checker." In the DMS, this is mapped by revoking her search permission in the "Vendors" *archive*, but creating a so-called *global bookmark* for her instead. This is ultimately a saved SQL search query (described in the system manual). This type of search query can be made available to the relevant users by the DMS administrator.

These *global bookmarks* can be used to provide the respective users with a set of documents on which they can then perform an advanced search.

Note: You can also combine the external call of a bookmark with the call of a bookmark XML or string or a gdocid search. This allows you to perform targeted searches externally based on the global bookmark.

The parameters in detail

`-gs:<name of a global bookmark>` Executes a *global bookmark*, which must be available to the logged-in user and, by definition, returns the restricted set of results that this user is allowed to see.

Note: The user should not have another *global bookmark* with the same name.

Based on this, you can use

`-so:<name of an XML file>`

`-so:<name of an XML bookmark string>`

`-so:<name of a local bookmark>`

or with the parameter

`gdocid:ID`

.

Example: `ViewerV3.exe -gs:My invoices gdocid:4711`

- ➔ Displays the document with *gdocid* 4711, even if the user does not have search authorization for the corresponding archive, provided that the document is accessible via the global bookmark. This maintains the authorization concept in the DMS while also enabling "searches" from outside.

Note: The global bookmark created for this purpose must not be an "empty search" in the respective archive.

Note: When using the `-so:` parameter, you can use complete bookmark XML files. These are executed based on the global bookmark. This also allows searches for additional criteria, such as additional and/or status fields. In the bookmark editor, you can display and save the bookmark XML by clicking on the info icon.

However, you can also enter a bookmark string directly. This must be valid XML with a `<bookmark>` and an underlying `<fav>` node. Since only the SQL Where string of the bookmark is used in the combined search with a global bookmark, such an XML string would look like this in its minimal version:

Example: `<?xmlversion=‘1.0‘?><bookmark><fav><sqlwhere>(sd.gdocid=4711)`

`</sqlwhere></fav> </bookmark>`

In the example, the search is for *gdocid* 4711

Note: Single quotation marks must be used within the XML.

Example: `-so:<?xmlversion=‘1.0‘?><bookmark><fav><sqlwhere>`
`(add_2=‘search text‘)</sqlwhere></fav></bookmark>`

12. Advanced search with an .FND file

With the bitfarm-Archiv Administrator, a "search template" can be created for each archive. This file with the extension .fnd is an empty framework similar to a .tpl file, which can be filled with search information and then transferred to the viewer as a parameter. This allows externally controlled searches to be carried out using specific additional fields.

V. Contact bitfarm-Archiv Software Support

If you have any questions or problems, please contact bitfarm software support

Phone: +49 (271) 31396-0

Email support@bitfarm-archiv.de

Support is available by phone:

Mon. – Thu. 8:00 a.m. to 5:00 p.m.

Fri. 8:00 a.m. to 3:00 p.m.

Please have your contract or partner number ready.