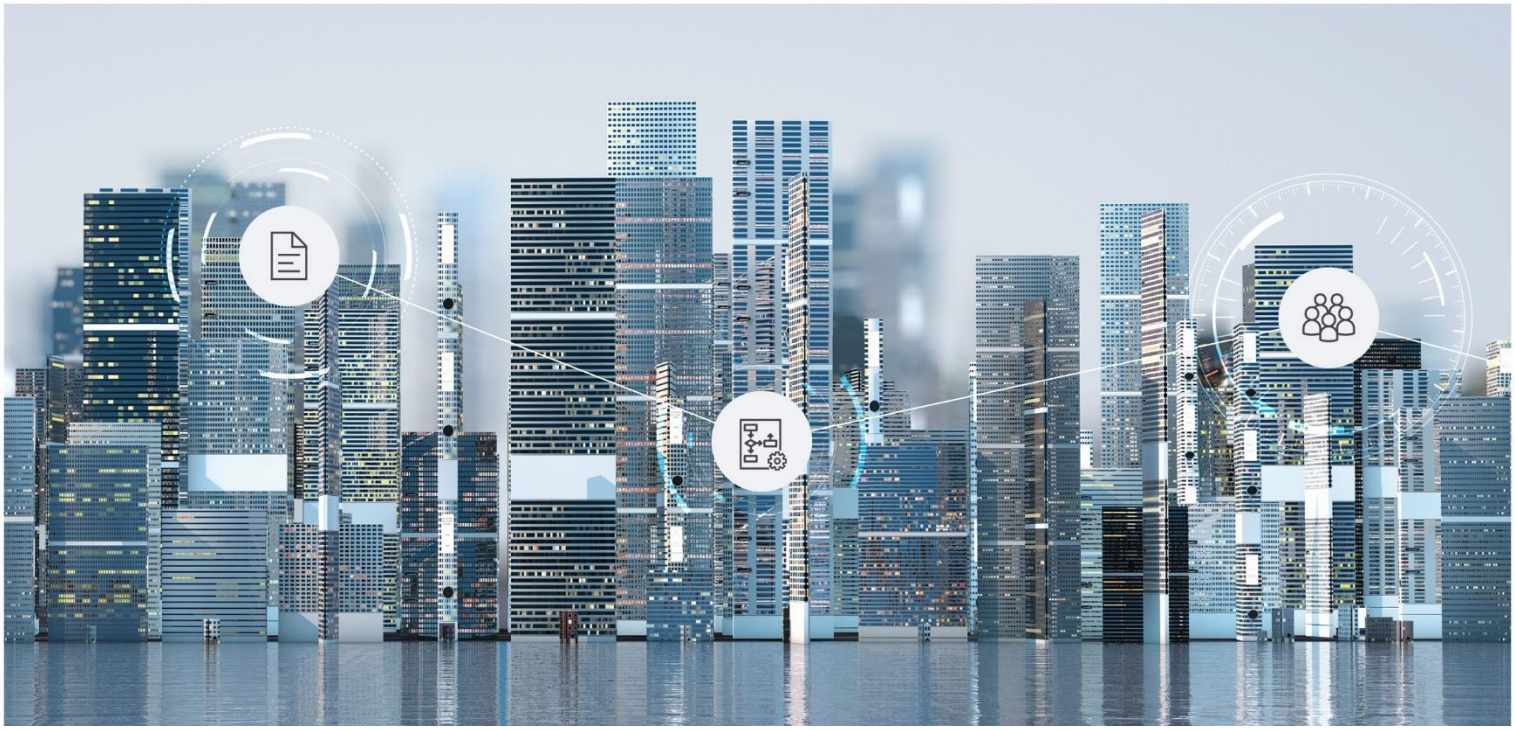
Fabasoft



White Paper

Fabasoft Folio Content Addressed Storage

Fabasoft Folio 2022 Update Rollup 1

Contents

1 Introduction

Fabasoft software products include the storage system “content addressed storage (CAS)” for Fabasoft Folio MMC Areas.

Content addressed storage ensures that certain content is only saved once. It is possible to reduce disk space usage, especially in larger installations with many content files, because without using a CAS system the same files are generally saved several times.

This document describes the new storage possibility of saved content files in Fabasoft Folio.

2 Software Requirements

System environment: All information contained in this document implicitly assumes a Microsoft Windows environment or Linux environment.

Supported platforms: For detailed information on supported operating systems and software see the software product information on the Fabasoft distribution media.

3 Configuration

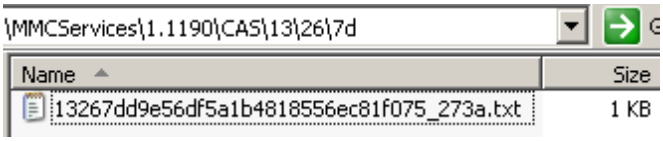
The chosen storage type for content files is set in the Fabasoft Folio MMC Areas in the corresponding Fabasoft Folio MMC Service object. It is not possible to change the storage type later on. To change the storage from “One Directory per Day (Change Date)” to “Content Addressed Storage (CAS)“ (or vice versa) the old Fabasoft Folio MMC Area has to be closed and a new one has to be created.

With the content addressed storage technology of the Fabasoft Folio MMC Services it is possible that several Fabasoft Folio MMC Services are using the same data pool. Thus, content managed by several Fabasoft Folio MMC Services is only stored once. Otherwise several Fabasoft Folio MMC Services may use different CAS areas (e.g. each Fabasoft Folio Tenant has an own CAS area)

4 Hash Value and Filing Structure

A hash value serves as base for the path of the respective content. Therefore, an MD5 hash value is calculated out of the content (MD5: 128 bit = 16 bytes = 32 signs (hexadecimal)). An MD5 hash value is a compromise between uniqueness and performance.

Two characters of the hash value act as directory. Dependent on the hash value, altogether three directories are created. Therefore, up to 256 directories on one level may be created.



The hash value, a random number (that is computed at the first writing of the content) and the file extension (in this example “txt”) represent the filename. If the same two content files have different file extensions only one file is stored in the Fabasoft Folio MMC Area with the file extension of the first file.

Fabasoft Folio Kernel stores the hash values, which belong to certain content files in the aggregate `COOSYSTEM@1.1:objlogcontmap`.

5 Cleaning a CAS Area

Changes the content of an existing Fabasoft Folio object, a new hash value is calculated and stored for the modified content. But the previous content is not deleted. To delete content files, which are not in use, call the action "Cleanup Content Areas" via the tool `fscadmin`. All content with hash values that are not referenced by a Fabasoft Folio object are deleted through this operation.

Fabasoft Folio Kernel reads all referenced hash values from the database (from all Fabasoft Folio COO Services) during cleaning the Fabasoft Folio MMC Areas and compares this set of values with existing filenames in the file system. File names which are not included in the set of hash values are not in use anymore and are deleted during cleanup.

Only all-lowercase CAS paths are expected for cleanup. At the first CAS level only CAS directories are considered. At the second and third CAS level only directories are considered. At the fourth CAS level only files are considered. For unexpected paths a warning is written to the log. To fix the "Skip uppercase directory" warning the directory has to be renamed to all-lowercase.

Files getting deleted are logged in the corresponding MMC Areas and backup directories. These logs are saved in `logs/casdeletion`.
If the option `--casdeletion <path>` is passed to `fscadmin`, the deletion logs are additionally stored at `<path>`.

Attention: Before cleaning a CAS area, a state-of-the-art backup is absolutely necessary.

Note:

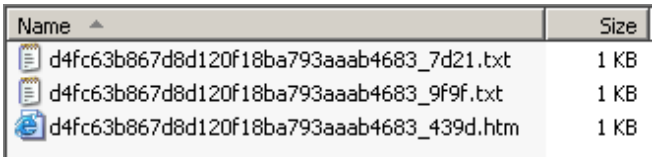
- The cleaning process needs, dependent on the extent of a Fabasoft Folio MMC Area, long time and should be planned accordingly. To restrict the cleaning process to specific directories of the MMC Area, the option `casrange` (e.g. `--casrange 00-3f`) can be used.
- To make successful cleaning possible, all Fabasoft Folio COO Services have to be available (important to run the query successfully).
- Content of objects that are located in inactive services are deleted, too.

6 Collisions Handling

Theoretical it is possible that different documents have the same MD5 hash value. If you assume that a hash value is unique, this could lead to data loss.

Therefore, Fabasoft Folio compares the content if the hash value is equally. If Fabasoft Folio determines that two different documents have the same hash value, the system adds a new random number to the existing hash value and saves the modified hash value to the content.

Example for a collision:

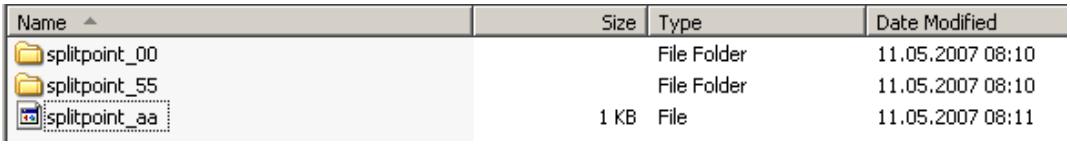


7 Scaling with Split Points

To enable a distribution of a CAS area, so-called split points can be created. The distribution of data is processed depending on the calculated hash values.

Example for split point configuration:

To distribute a CAS area to three different storage devices, three split points can be created directly in the defined Fabasoft Folio MMC Area. A split point can be a directory or a file. If the split point is a directory then this directory will be used for saving the corresponding data. In case of a split point file the path specified in this file will be used. Such a file can contain a network path or a local directory path. A split point has the following syntax: `splitpoint_<xx>`. `<xx>` represents the first two characters of the hash value.



In this example the split points are "00", "55" and "aa". By means of this structure all hash values starting between "00" and "54" are stored in the directory `splitpoint_00`, all hash values between "55" and "a9" in the directory `splitpoint_55` and all hash values from "aa" in the directory specified in the `splitpoint_aa` file. According to this illustration it is possible to create up to 256 split points.

If the `splitpoint_00` split point is not created the content is located directly in the defined Fabasoft Folio MMC Area directory.

Split points can be created or deleted as necessary. The right distribution of the content (changes in the file system have to be processed manually) and a restart of Fabasoft Folio MMC Services (the split point structure is only read at first access) are important for successful modification.

Note: The consistence of split points is checked after first access to a CAS area. In cases of inconsistence (files are in the wrong split point directory) an error message is written to the event log.

In a Linux system environment, so-called "symbolic links" can be used to locate split point directories on different storage devices. In a Microsoft Windows system environment, a so-called "mount point" can be created on these directories.

If file shares are used as MMC or split point directories, the system administrator must ensure that these directories are mounted before the MMC service or the Kernel accesses the CAS area for the first time. In Linux the configuration file `/etc/fsctab` can be used to specify all mounts. If one mount requires another, the option `x-systemd.requires-mounts-for` can be used in the `fstab` config file.

Example `fstab` entries:

```
//10.10.10.1/mmc /data/mmc cifs domain=mydomain,username=user,x-systemd.requires-mounts-for=/data/splitpoints 0 2
//10.10.10.2/splitpoints /data/splitpoints cifs domain=mydomain,username=user 0 2
```

Furthermore, it is possible to define dependencies for service units to ensure required mountpoints are available before starting a service. To add a specific mountpoint as dependency for the `fsc.service` an `override.conf` file needs to be created in the `fsc.service.d` folder. This can be done by executing the following command:

```
systemctl edit fsc.service
```

For every required mountpoint the option `After=` and `Required=` with the associated mount file needs to be added in the `[Unit]` section.

Example `override.conf` file:

```
[Unit]
After=data-mmc.mount
Requires=data-mmc.mount
```

The mount files (units) are automatically generated by `systemd` for every `fstab` entry. To get the unit names of all mountpoints, execute the following command:

```
systemctl list-units --type=mount
```

Reload the `systemd` daemon to make the changes take effect.

```
systemctl daemon-reload
```

8 Configuring CAS Cache

For performance reasons each Fabasoft Folio Kernel instance has a CAS cache by default. When requesting a content, the Fabasoft Folio Kernel first tries to read the content from the CAS cache. If the content identified by a hash can be found in the CAS cache the content can be used directly from the cache and the Fabasoft Folio MMC Service is not required for reading.

The path of the CAS cache directory can be configured via the environment option `CASCACHEDIR` (e.g. `HKEY_CURRENT_USER\Software\Fabasoft\Environment\CASCACHEDIR` in a Microsoft Windows environment).

By default, all CAS areas of all Fabasoft Folio MMC Services are treated as a separate data pool in the cache. In case that several Fabasoft Folio MMC Services are using the same data pool the configuration can be adopted to combine the areas also within the cache. Therefore configure the property *Shared Cache Name (CAS)* (`COOSYSTEM@1.1:dareasharedname`) in the *MMC Service Areas* aggregate of the Fabasoft Folio MMC Services. Areas with the same shared cache name are treated as one CAS data pool.

In order to prevent an unlimited growth of the CAS cache the Fabasoft Folio Kernel calls periodically a cleanup routine for the CAS cache. Information about the cleanup can be found in the event log. The maximum size and the time interval of the recurring cleanup process can be configured via the environment options `CASCACHEMAXMB` and `CASCACHECLEANUPINTERVAL`, respectively. It is also possible to call the cleanup mechanism directly via the action `COOSYSTEM@1.1:CleanupCASCache`. When doing so the environment option `CASCACHECLEANUPINTERVAL` is ignored.

The CAS cache can be disabled via the environment option `ENABLECASCACHEDIR`.

9 Direct Access to a CAS area

For reliability and performance reasons it is possible to configure more file shares for the same MMC area. File shares can be configured as UNC paths by the property *Path to Directory on Server* (`COOSYSTEM@1.1:darearemdir`). If file shares are configured the Fabasoft Folio Kernel reads and writes all contents from the shares without the Fabasoft Folio MMC Service.

When the Fabasoft Folio Kernel reads a content, the file access occurs on a randomly selected file share. (Load distribution). If the content cannot be found on one share another share is selected. (Reliability)

When writing contents, the Fabasoft Folio Kernel writes parallel to all configured file shares. The transaction is successful when at least on 50 percent of the configured shares the write operation succeeded.

Note:

- File servers must process write requests synchronously or honor filesystem synchronization requests to prevent data loss. In case of Samba (CIFS), the server option `strict sync = yes` is required. In case of Linux/nfsd (NFS), the export option `sync` is required.
- The user context in which the Fabasoft Folio Kernel is running (e.g. Web Service User) must have read and write privileges on the configured shares.
- On Linux systems, the configured UNC paths will be resolved automatically based on currently mounted filesystems.
- Changing the MMC area configuration requires the execution of "Synchronize Registry Entries".
- Changing the MMC area configuration requires a restart of the kernel.
- The number of writer threads can be configured via the environment option `MMCWRITETHREADCOUNT` and is 12 by default.
- For each Fabasoft Folio Kernel a preferred read share can be configured via the environment option `MMCREADDIR_<areaname>` (e.g. `MMCREADDIR_MMCSVC1 = "\\192.168.100.100\CAS2"`).

10 Backup Directory

A backup directory can be configured for online and offline backups. Both are based on a daily directory structure. An online backup directory configured via the properties *Backup Directory on Server (Online)* (`COOSYSTEM@1.1:dareabackupdir`) or *Path to Backup Directory on Server (Online)* (`COOSYSTEM@1.1:dareabackupremdir`) means that contents will be backed up in the directory during normal operations. Additionally, this kind of backup directory can also be synchronized with the configured area paths. On the other hand, an offline backup directory configured via the property *Path to Backup Directory on Server (Offline)* (`COOSYSTEM@1.1:dareaofflinebackupremdir`) will not be used during normal operations. This backup directory will only be used for manual backup tasks.

When using the options `synctobackup` or `syncfrombackup` of the action *Cleanup Content Areas* in the tool `fscadmin`, the configured area paths can be synchronized with the backup directory. Using the option `cleanupbackup`, a cleanup of the backup directory can be started. The cleanup is based on file system operations meaning that each file in the backup directory which is no longer available in one of the configured area paths will be deleted. `cleanupbackup` only makes sense when the operation is executed after removing unreferenced files on the configured area paths using the action `cleanup`.

Using the option `offlinebackup` with `synctobackup`, `syncfrombackup`, or `cleanupbackup` means that the configured offline backup directory is used for the specified operation instead of the online backup directory.

Note:

- The synchronization using `synctobackup` or `syncfrombackup` is based on the log files written in the directory *logs* of the configured directory.
- A content will not be written to the daily backup directory when the content already exists in the single instance areas.
- Changing the MMC area configuration requires the execution of "Synchronize Registry Entries".

10.1 Delayed Backup Directory Cleanup

The tool `cleanbackup` removes files from the backup directory by using the deletion log generated during cleaning up an MMC Area. This tool takes the parameters `root`, which specifies the path of the backup directory to be cleaned up and `days` to define the minimum number of passed days since the deletion of a file in the MMC Area for deleting it in the backup directory, too.

Note:

- If a cleanup with `fscadmin` has already been run once and no CAS deletion logs exist, a verbose `cleanupbackup` dry-run needs to be performed with `fscadmin`. The date range should be until yesterday. This output of `fscadmin` needs to be converted to deletion logs using the tool `logtohashlog`. The generated deletion logs need to be placed in `logs/casdeletion` in the associated backup directory.
- The backup can be spread across multiple paths. In this case, *Path to Backup Directory on Server (Online)* (`COOSYSTEM@1.1:dareabackupremdir`) represents the active backup path containing most recent backup data. Old backup data can be moved to other paths that are not configured in Fabasoft Folio. In this scenario, `cleanbackup` must be run for all paths. As CAS deletion logs are only copied to the active backup path, logs in `logs/casdeletion` must be copied to all other paths before executing `cleanbackup` for the active backup path. To override the path to `logs/casdeletion` the option `-casdeletion <path>` can be used.
- On Linux systems `cleanbackup` also deletes immutable files if run as root.

11 Check Content Consistency

The availability and consistency of files in CAS areas can be checked by using the `checkcas` utility on the file server. For every entry in the CAS area, the content address is read and validated. Information about the results is stored as a structured report in the root of the CAS area.

On the file server the `checkcas` utility can be executed as follows.

Example

```
[root@example 1.506]# checkcas -root MMCSVC1/
Fabasoft CheckCAS Version 21.9.0.9
Copyright (c) Fabasoft R&D GmbH, A-4020 Linz, 1988-2021.
CheckCAS:
Start at            : 2021-10-21T11:14:11
Directory           : MMCSVC1/
Report              : /var/opt/fabasoft/lib/mmc/1.506/MMCSVC1/logchecks/2021/10-
21/check-2021-10-21T11-14-11.json
SKIP: directory /var/opt/fabasoft/lib/mmc/1.506/MMCSVC1/logscans
SKIP: directory /var/opt/fabasoft/lib/mmc/1.506/MMCSVC1/temp
SKIP: directory /var/opt/fabasoft/lib/mmc/1.506/MMCSVC1/logchecks
SKIP: directory /var/opt/fabasoft/lib/mmc/1.506/MMCSVC1/logs
FAIL: 79adac8c671790504a142ab2392cf927
[/var/opt/fabasoft/lib/mmc/1.506/MMCSVC1/31/f9/c5/31f9c5546c00cfd90b310f197bb53e
53_5675.cas]
Summary:
Directory           : MMCSVC1/
Directories scanned : 59923
Directories ignored : 4
Files scanned       : 33471
Files ignored       : 0
Files hashed        : 33471
Files failed        : 1
Thread E/D          : 0/10
Start at            : 2021-10-21T11:14:11
```

```
Stop at             : 2021-10-21T11:14:14
Runtime             : 00:00:04 seconds.
Report written      : /var/opt/fabasoft/lib/mmc/1.506/MMCSVC1/logchecks/2021/10-
21/check-2021-10-21T11-14-11.json, 451 bytes
Report updated      :
/var/opt/fabasoft/lib/mmc/1.506/MMCSVC1/logchecks/latest.json
Check completed with errors, found 1 invalid files.
```

The latest log file is saved in the `logchecks` folder.

Example

```
/var/opt/fabasoft/lib/mmc/1.506/MMCSVC1/logchecks/latest.json
{
   "checkpath":"MMCSVC1/",
   "checkfile":"/var/opt/fabasoft/lib/mmc/1.1250/MMCSVC1/logchecks/2021/10-
22/check-2021-10-22T06-37-58.json",
   "checkstartat":"2021-10-22T06:37:58",
   "checkendat":"2021-10-22T06:38:03",
   "success":false,
   "checkeddirectoriescnt":29129,
   "ignoreddirectoriescnt":3,
   "checkeddocumentscnt":15218,
   "faileddocumentscnt":1,
   "ignoreddocumentscnt":0,
   "faileddocumenthashes":[
      {
         "hash":"d555ddb08e463407614524896034bc1c_6d69",
         "detail":"Invalid hash."
      }
   ]
}
```

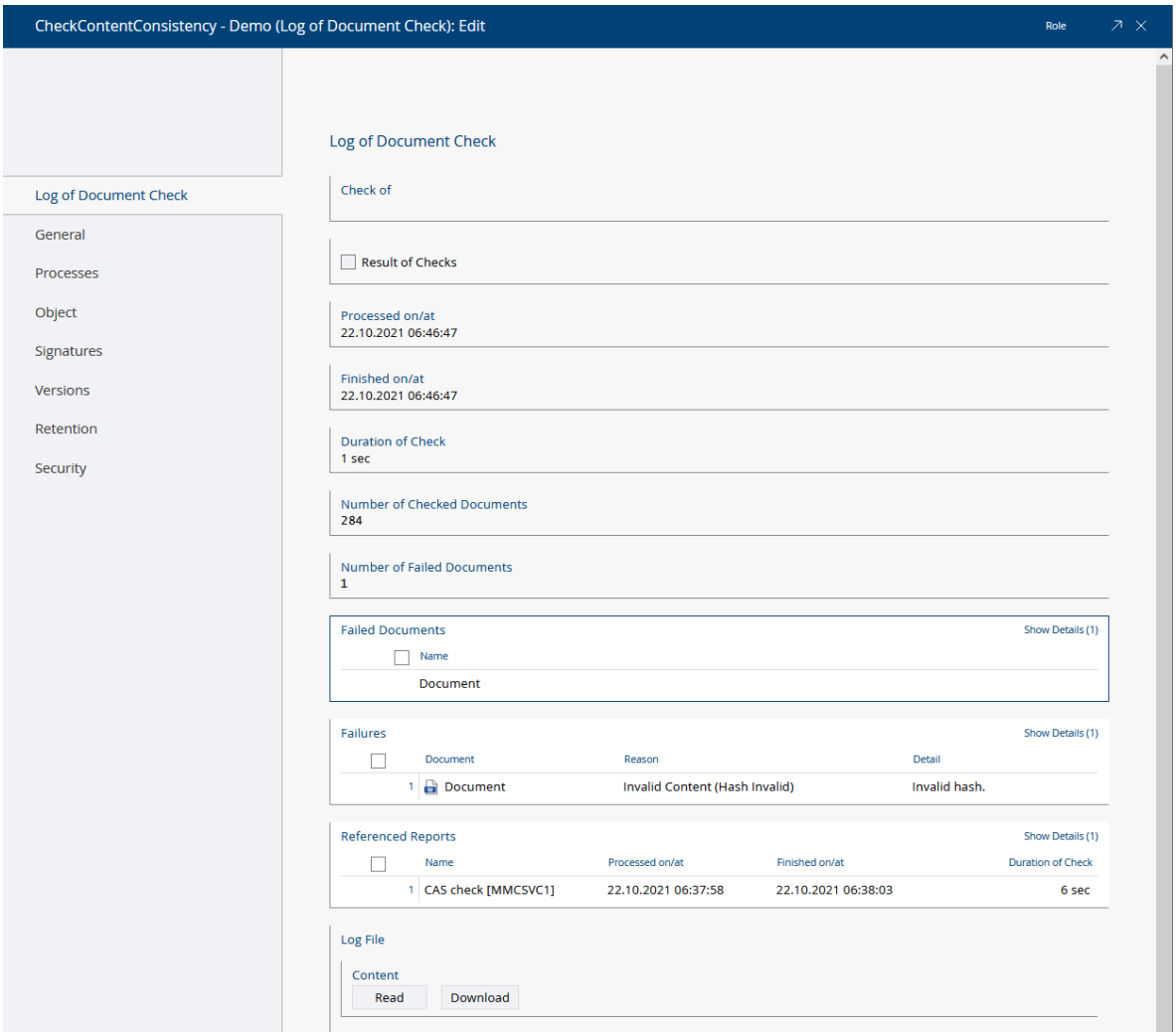
Failed files are listed in `faileddocumenthashes` whereby `hash` indicates the failed file name (without extension) and `detail` contains the corresponding information.

As administrator it is possible to generate a report based on this log file by executing the `COOSYSTEM@1.1:CheckContentConsistency` action.

Example

```
q="SELECT * FROM ContentObject WHERE true";
r=FSCCHECK@1.1001:GroupContentCheckLog.ObjectCreate();
r.objname = "CheckContentConsistency - Demo";
coort.GetCurrentDomain().CheckContentConsistency(q,r,true);
cootx.Commit();
```

The generated report object can be found and evaluated in Fabasoft Folio.



12 Check Virus Scan

It is possible to block malware in Fabasoft Folio, which was found by an Antivirus software. Therefore, affected documents will be marked as infected and the current content is replaced (the primary content will not get lost). Additionally, these objects can no longer be opened or edited and the user receives a warning.

This feature can be used independently from any kind of Antivirus software but the output must be prepared in a certain structure.

12.1 Requirements and Configuration

The scan output must be provided in JSON format and must be structured as follows.

Example

```
{
  "checkstartat": "2021-10-12T11:23:00",
  "checkendat": "2021-10-12T11:23:45",
  "virusdocumenthashes": [
    {
```

```
      "hash":"cb0b4f7eca5c318724eb001d8099632f_c8f8",
      "detail":"Virus 1"
    },
    {
      "hash":"894d83bf6934c1ee6c989d91ca596995_ee54",
      "detail":"Virus 2"
    }
  ]
}
```

In the JSON, `hash` indicates the affected document name (without extension) found by the Antivirus software and `detail` contains the virus information.

Fabasoft Folio Configuration

In Fabasoft Folio direct access must be configured for the MMC area (see chapter 9 "Direct Access to a CAS area").

The generated JSON file must be placed in the dedicated MMC area in a pre-defined directory called `logscans` and must have the name `latest.json`.

Example

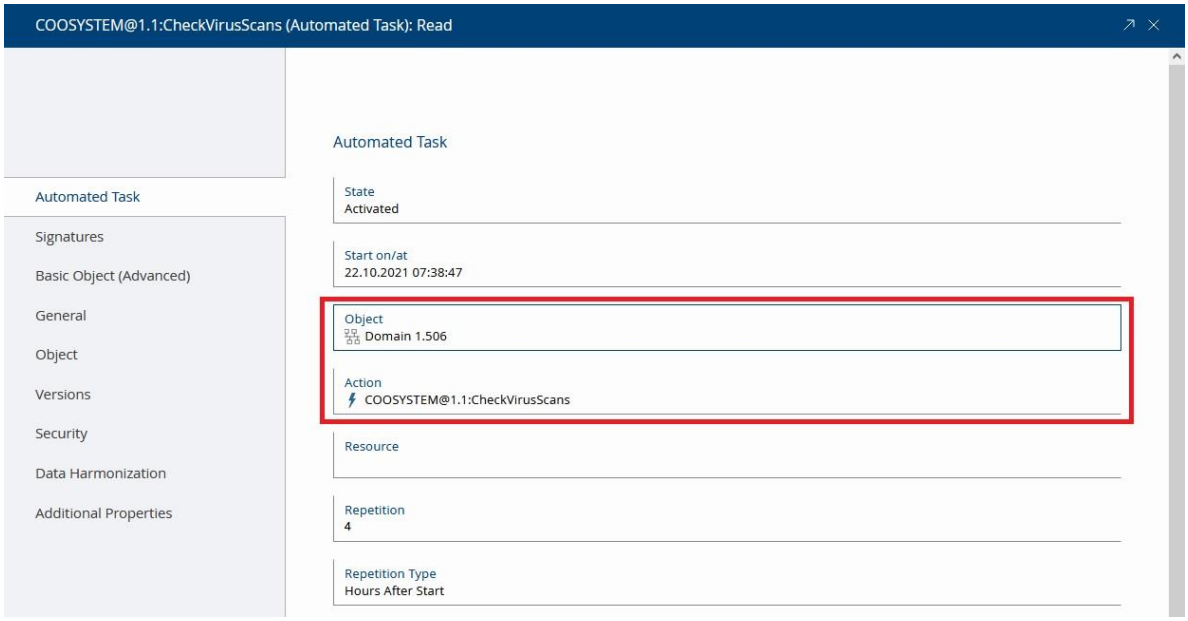
```
[root@foliofileserver MMCSVC1]# pwd
/var/opt/fabasoft/lib/mmc/1.506/MMCSVC1
[root@foliofileserver MMCSVC1]# ls -l logscans/
total 4
-rwxr-xr-x. 1 fscsrv fsc 190 Oct 19 09:29 latest.json
```

12.2 Execution

If the JSON file is placed in the `logscans` directory and direct access is configured, the execution of `COOSYSTEM@1.1:CheckVirusScans` initializes/updates the attribute `COOSYSTEM@1.1:objscaninfo` based on the JSON input. All files in the CAS area of all versions referenced by objects in Fabasoft Folio are checked.

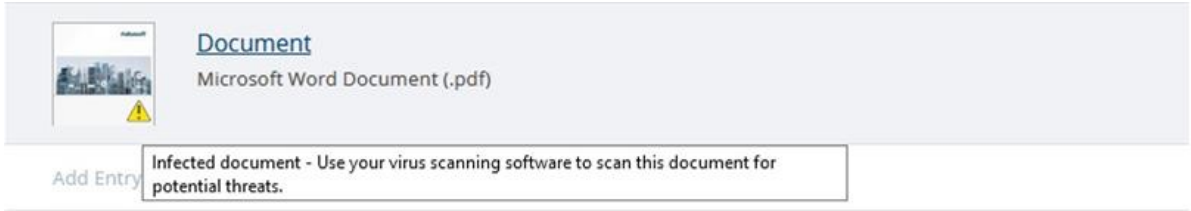
The execution can be accomplished by the AT service. Therefore, a new AT service task must be created and placed in a list of automated tasks.

The object is the current domain and the action is `COOSYSTEM@1.1:CheckVirusScans`.

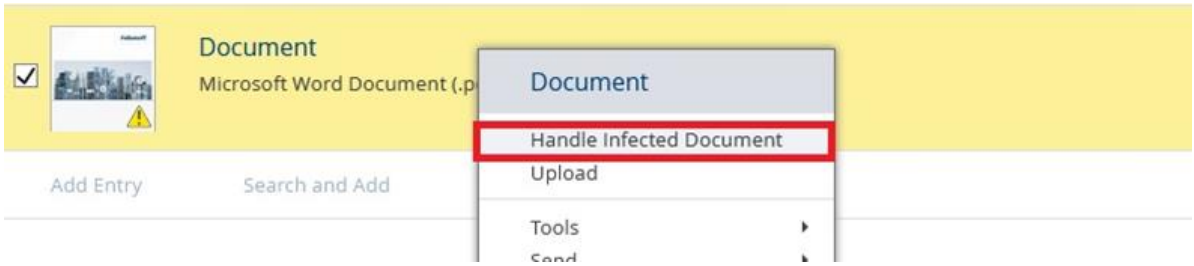


12.3 Handle Infected Documents

In case of a virus, the objects in Fabasoft Folio, which reference this content are marked as infected and consequently cannot be edited or opened by the end user.



The infected document can be handled using the “Handle Infected Documents” context menu command.



This way the user can download, scan and clean the file with an own virus scanning software. The cleaned file can be uploaded again. If the file has been mistakenly classified as a potential threat, the warning can be removed. Alternatively, the document can be deleted.

13 Restrictions

Full-text search in Fabasoft Folio MMC Areas of type “content addressed storage (CAS)” is only supported with Fabasoft Mindbreeze Enterprise. Full-text search based on Microsoft Index Service is not supported by these Fabasoft Folio MMC Areas.