

Common-Inbox V1.0 Documentation - 02.08.2018

Disclaimer / Liability:

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or any later version. This program and its documentation is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details: [GNU General Public License](#)

Table of Contents

1. About

2. Features

3. Installation

3. 1. Linux

- 3. 1. 1. [Software Prerequisites](#)
- 3. 1. 2. [Create User](#)
- 3. 1. 3. [Unpack tarball](#)
- 3. 1. 4. [Checkout from repository](#)
- 3. 1. 5. [Start-up Parameters](#)
- 3. 1. 6. [Log Levels](#)
- 3. 1. 7. [Access](#)

3. 2. Windows

4. Configuration

4. 1. Basics

- 4. 1. 1. [Inbox Structure](#)
- 4. 1. 2. [Temporary Directory](#)
- 4. 1. 3. [Item IDs](#)
- 4. 1. 4. [Naming limitations](#)
- 4. 1. 5. [Incoming Files/Folders](#)
- 4. 1. 6. [Output](#)
- 4. 1. 7. [Placeholder Variables](#)
- 4. 1. 8. [Default Workflow](#)
- 4. 1. 9. [Tasks](#)
 - 4. 1. 9. 1. [FilesWait](#)
 - 4. 1. 9. 2. [DirListCSV](#)
 - 4. 1. 9. 3. [Clean Filenames](#)
 - 4. 1. 9. 4. [PreProcs](#)
 - 4. 1. 9. 5. [FilesValid](#)
 - 4. 1. 9. 6. [FilesMustExist](#)
 - 4. 1. 9. 7. [HashGenerate](#)
 - 4. 1. 9. 8. [HashSearch](#)
 - 4. 1. 9. 9. [CopyToTarget](#)

- 4. 1. 9. 10. [HashValidate](#)
 - 4. 1. 9. 11. [RenameTarget](#)
 - 4. 1. 9. 12. [HashOutput](#)
 - 4. 1. 9. 13. [PostProcs](#)
 - 4. 1. 9. 14. [LogFileCopy](#)
- 4. 1. 10. [Custom Tasks](#)
- 4. 1. 11. [Task States - Return Codes](#)
- 4. 1. 12. [Task Modes](#)
- 4. 2. [Configuration File: cinbox.ini](#)
 - 4. 2. 1. [Inbox Configuration Options](#)
 - 4. 2. 2. [Cool-off mechanism](#)
 - 4. 2. 3. [\[INBOX\]](#)
 - 4. 2. 4. [\[Directory\] Sections](#)
 - 4. 2. 5. [Directory Configuration options](#)
 - 4. 2. 6. [\[DEFAULT\]](#)
 - 4. 2. 7. [\[UNDEFINED\]](#)
 - 4. 2. 8. [\[.\]](#)
 - 4. 2. 9. [All Configuration Options](#)
- 4. 3. [Plug-Ins](#)
 - 4. 3. 1. [MailNotification](#)

5. [Running Common-Inbox](#)

- 5. 1. [Manual mode](#)
- 5. 2. [Automatic mode](#)
- 5. 3. [Logging](#)

6. [Examples](#)

- 6. 1. [Example 1: Simple wav+mp3 import](#)
 - 6. 1. 1. [Scenario](#)
 - 6. 1. 2. [Workflow design](#)
 - 6. 1. 3. [Configuration](#)
 - 6. 1. 4. [Process details](#)
- 6. 2. [Example 2: Different target directories and existing checksums](#)
 - 6. 2. 1. [Scenario](#)
 - 6. 2. 2. [Workflow design](#)
 - 6. 2. 3. [Configuration](#)
 - 6. 2. 4. [Process details](#)

7. [Troubleshooting](#)

- 7. 1. [Support](#)
- 7. 2. [The cinbox.php script can't be executed](#)
- 7. 3. [Changing language does not work](#)
- 7. 4. [Executing cinbox.php results in a black screen](#)
- 7. 5. ["bad interpreter" error](#)
- 7. 6. [ERROR Folder does not exist](#)
- 7. 7. [Inbox contains 0 Items](#)
- 7. 8. [CleanFileNames: Empty or invalid character map](#)
- 7. 9. [ERROR Invalid configuration 'CONFIG_OPTION': Must be an array.](#)
- 7. 10. [Inbox is reporting the same error after the issue has been resolved](#)
- 7. 11. [Undefined variable: age](#)

1. About

The Common-Inbox (CIN) is a very flexible engine for migrating data. It ensures data integrity and can run in most environments. Further, it allows for Plug-Ins, pre- and postprocessing scripts and the definition of a rule set (configuration) for each Inbox. Depending on the configuration, the data files, metadata, hashes and directory structure will be handled to migrate automatically and only have user interaction in error cases. Additionally, multiple languages are supported (Currently: EN, DE).

2. Features

- Easy setup and configuration
- Ready-to-use default Workflow
- Easy to use
- Filename cleaning/normalizing
- File verification and hash code handling
- Detailed logging
- Support for importing existing hash codes
- Support for splicing an Item to multiple output paths
- Support for custom pre- and post-processing scripts
- Support for Plug-Ins
- Support for multiple languages

3. Installation

The installation requires some basic Linux/IT know-how. It is strongly recommended to set up CIN in a sandbox environment and try configurations with a test set of data before using it on live systems. This documentation expects that the user knows how to operate the system shell and use text editors and other tools, such as an SSH client.

3.1. Linux

A Ubuntu Server (16.04) environment will be used as example for the setup. Depending on your Linux flavor, some steps might be different, but most things should be very similar. CIN is tested on php5 (and might also work on php7).

3.1.1. Software Prerequisites

CIN requires the basic system packages and PHP with the CLI extension installed. Currently, PHP5 is supported and PHP7 should work (but is not officially supported yet). Also, rsync is required but should come with the basic system utilities. To prevent shell session aborts, CIN should be run in a "[GNU screen](#)" session, especially in automatic mode.

Packages required (Ubuntu Server 16.04):

- php
- php-cli
- rsync
- screen

Packages optional (Used in examples):

- samba-server




3.1.2. Create User

Use the system utilities to create a user. In this example, we create the user 'cin' by executing `$ sudo adduser cin`. It might be required to additionally fill in some fields after supplying a password for the new user. After the user has been created, log on with it and check if the home directory was correctly created by executing `$ cd; ls -la`. You should see some files and folders in your home directory, on most systems the command prompt will look like this if you're in your home directory (~): `cin@sandbox:~$`.

3.1.3. Unpack tarball

If you have a tarball of the latest CIN build, upload it to your home directory using SFTP or FTP, depending on your setup. [\[1\]](#) can be used for both methods. After the upload, unpack it by using the command `$ tar -xzf cinbox.tar.gz` (the exact filename might be slightly different). This should create a directory called 'cinbox' in the home directory of the user 'cin', containing the following:

```
home
├── cin
│   └── cinbox
│       ├── bin
│       ├── doc
│       └── locale
```

Directory	Description
 bin	Contains the CIN scripts, default Tasks, Plugins, etc.
 doc	The CIN code documentation, generated by Doxygen.
 locale	Available translations.

3.1.4. Checkout from repository

⚠ Add repository checkout information.

3.1.5. Start-up Parameters

Execute the script and you should get the parameter help output: `cd ~/cinbox/bin; ./cinbox.php -h`.

Parameter	Description	Required
-i /path/to	Inbox source folder	mandatory
-c /path/to/file	Configuration file. If none specified, the default 'cinbox.ini' will be looked for in the specified Inbox path.	optional
-h	Show the help output	optional
-p /path/to	Processing folder, contains logs and state-keeping directories.	optional
-v	Enable verbose logging (LOG_LEVEL = INFO)	optional
--debug	Debug mode (LOG_LEVEL = DEBUG)	optional
--logstyle	Set output format of logfile	optional
--lang en_GB	Set the language. Currently supports en_GB, de_AT.	optional
--log /path/to/file	Set the log file for the Inbox (Items have their own logs).	optional
--forever	Continue checking the Inbox for Items until interrupted by user (automatic mode).	optional
-n 5	Items at once. Overrides the config file setting.	optional
-w 5	Wait-time in minutes. How long to wait between iterations.	optional

3.1.6. Log Levels

String	Numeric
LEVEL_NONE	0
LEVEL_DEBUG	1
LEVEL_INFO	2
LEVEL_NORMAL	3
LEVEL_WARNING	5
LEVEL_ALWAYS	10
LEVEL_ERROR	20

3.1.7. Access

It is the choice of the system administrator how to provide access to the necessary shares. The machine running Common-Inbox must be able to access the configured Inbox paths as well as the archive paths with correct user/access rights. This can be done by uploads over SFTP, which is working out of the box with the 'cin' user, but not very user friendly. Another option would be to have Inboxes as samba shares, so people can conveniently drag and drop files to the Inbox as needed. It might be necessary to mount network shares to local mount points for easier handling. In most examples, we will use /mnt/inbox and /mnt/archive as base source and target path, respectively.

For a test environment, a minimal samba configuration without access control might do. For the sandbox in the examples below, we'll use the following `/etc/samba/smb.conf`:

```
[global]
workgroup = Testing
netbios name = SANDBOX
security = user
map to guest = Bad User
usershare path =

[inbox]
comment = CIN Test Inbox
path = /mnt/inbox
force user = cin
force group = cin
read only = No
guest ok = Yes

[archive]
comment = CIN Test Archive
path = /mnt/archive
force user = cin
```

```
force group = cin
read only = No
guest ok = Yes

[CIN_Home]
comment = CIN Home
path = /home/cin
force user = cin
force group = cin
read only = No
guest ok = Yes

[cache]
comment = CIN Test Cache
path = /mnt/cache
force user = cin
force group = cin
read only = No
guest ok = Yes
```

When accessing `\\sandbox` from a file explorer tool (e.g. Windows explorer, not to be confused with Internet Explorer), the shares should be visible and read/write/delete should work.

3.2. Windows

Common-Inbox is currently not supporting installations on Microsoft Windows.

4. Configuration

This section will cover configuration options with some examples. CIN is built for very high flexibility so many cases can be handled without code modifications. Depending on the actual project, some pre- and postprocessing scripts can be used if needed. Some configuration examples can be found further below in the article.

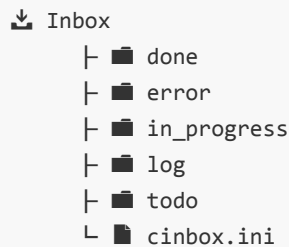
The fresh CIN installation will come with a file called `cinbox.conf.example` that can be used as an example template. Of course it is also possible to create a new text file and write the configuration into it.







4.1. Basics

It is most important to be aware of the expected input, wanted processing and desired output structure as well as the surrounding archive rules and formats used. A road map of what types of data are expected in what formats and possible naming schemes, how this data should be treated and where it should be written to and verified at in the end, can be very helpful. CIN also allows to configure rules how unexpected data should be handled. Thought well put into the configuration saves a lot of trouble later.

4.1.1. Inbox Structure

The Inbox is the directory that contains the configuration and structure required to process a specific workflow . There must be one configuration file per Inbox, with the filename **cinbox.ini** in the root. Additionally, the following directories must exist and be accessible:



(Sub-)Directory	Description
 Inbox	The Inbox root. Must contain the 'cinbox.ini' file.
 todo	Items dropped into this directory will be processed according to the configured workflow.
 in_progress	Items that are currently being processed will be moved here.
 done	Successfully processed Items will be moved into the done directory. Depending on garbage collection settings, they will reside there for a while and then get deleted.
 error	If processing of an Item has to be stopped due to errors or the configured rules, the Item will be moved to this directory.
 log	Contains the log files for all CIN operations, depending on configuration (see parameter MOVE_LOGFILES) . The default file name is the Item ID + .log, e.g. TEST-123456.log and the default log behavior will have the log files in the Item's working directory.

4.1.2. Temporary Directory

To store generated data (e.g. hash files) before it is written to the archive location or handled otherwise, a directory will be created in the system default temp location. For most linux distributions, this will be /tmp. There, a sub-directory with the name of the inbox is created. This name will be "cleaned" for easier handling, e.g. by replacing white spaces with underscores. If you have an Inbox with the name "Inbox 1", the temp directory used will be /tmp/Inbox_1.

Data of Items that are currently processed will be stored in yet another sub directory within the Inbox temp. If the Archive Item has ID 'ABC-1234', the related temporary files can be found in /tmp/Inbox_1/ABC-1234.

The actual used temp directory can also be found in the Item's log if a Task is using it for operations:


```
[2017-10-10T14:22:57] N =====
[2017-10-10T14:22:57] N Executing task #4: 'HashGenerate'...
[2017-10-10T14:22:57] N =====
[2017-10-10T14:22:57] I Running task 'HashGenerate' on folder: '.'
[2017-10-10T14:22:57] I Task 'TaskHashGenerate': temp folder is '/tmp/Inbox_1/ABC-1234'.
```

If an Item goes into error, the temporary files are **not** deleted. This design decision allows for a workflow to continue from the point it was interrupted once the cause for the error is resolved, without having to re-run the whole workflow. Especially when treating hires video files, it is desirable to rather continue workflows, due to the long run times when processing large files.

In some cases it might be necessary to **completely** reset a workflow. In this case, the Archive Item's temp directory must be deleted to force a 100% re-processing of the Item.

Important:

The `/tmp` directory is usually located in the system root, not in the respective Inbox directory! It might require assistance of an administrator to access this location in order to fully reset a workflow.

4.1.3. Item IDs

Most Archives have a unique identifier for their assets, often known as "Archive Signature", "UUID" or similar: In CIN, these are referred to as **Item IDs**. Often those Item IDs also follow a specific Format, e.g. three chars, a dash and 10 digits (ABC-1234567890) or a numeric value only (1234567890). If CIN should only work on Items (files, directories, ...) with IDs in the correct expected format, a regular expression is used to configure a filter.

Important:

In the Common-Inbox default environment, the Item is always a sub-directory of the Inbox, e.g. `/inbox/ABC-123` contains all files for the Item with ID 'ABC-123'. Also see "Incoming Files/Folders"

4.1.4. Naming limitations

Depending on the operating system and other factors, there are limits for what characters can be used to name files and folders. CIN provides the Task *Clean Filenames* to handle incoming files and check them for invalid characters. There are several configuration options that will treat a specific set of unwanted characters and replace them according to the following tables:

whitespace		
" "	⇒	-

slashes		
\	⇒	-
/	⇒	-

illegal		
?	⇒	-
*	⇒	-
:	⇒	-

picky		
#	⇒	-
,	⇒	-
;	⇒	-
&	⇒	and

quotation		
"	⇒	"
"	⇒	"
'	⇒	'
`	⇒	'

quotation2		
"	⇒	-
"	⇒	-
'	⇒	-
`	⇒	-
"	⇒	-
'	⇒	-

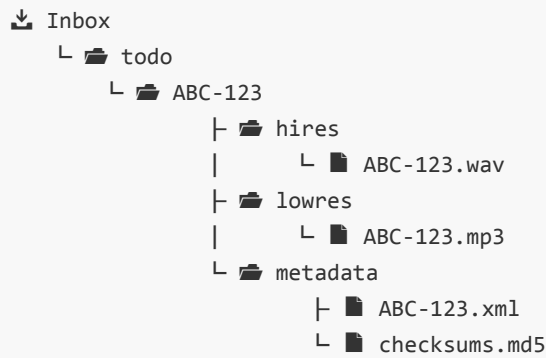
umlauts		
ä	⇒	ae
ü	⇒	ue
ö	⇒	oe
ß	⇒	ss
Ä	⇒	AE
Ü	⇒	UE
Ö	⇒	OE

brackets		
(⇒	-
)	⇒	-
[⇒	-
]	⇒	-
{	⇒	-
}	⇒	-
<	⇒	-
>	⇒	-

4.1.5. Incoming Files/Folders

Ideally, it is known what structure incoming data has, e.g. a directory having the Item ID as name, containing the files or sub-directories to be imported and (optionally) already existing hashes for verification. There might also be metadata files accompanying the archive files. In some cases, it happens that unexpected data is found and needs to be handled as well, e.g. additional booklet or cover scan images.

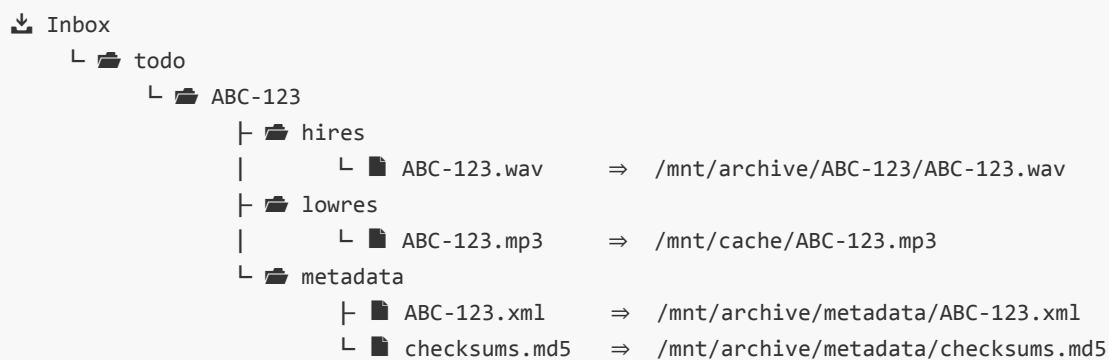
Example:



4.1.6. Output

The data set will be processed as configured and written to the final path(s). This might be the actual archive or a storage where the archive gathers data from later on. It is possible to split a data set into different output paths, e.g. one path for metadata and another one for audio files. All files should be verified at the target using either existing hashes or those generated by CIN.

Example:



4.1.7. Placeholder Variables

These variables can be used in configuration options. Most variables will only return data of the Item currently in progress while some are of a more global nature.

Variable	Description	Returns / Example
[@ITEM_ID@]	The ITEM_ID that is currently processed, exactly as is (mixed-case enabled)	Test-123456
[@ITEM_ID_LC@]	The ITEM_ID in lower-case only	test-123456
[@ITEM_ID_UC@]	The ITEM_ID in upper-case only	TEST-123456
[@BUCKET@]	This requires the /plugins/objid_to_path.sh plugin to be enabled. The plugin will generate a path and return this variable for further use.	A path created by the plugin script, e.g. /TEST/123456
[@DATETIME@]	System clock: Date + Time	
[@DAY@]	System clock: Day	27
[@DIR_BASE@]	Returns the current Item's base folder	/mnt/inbox/todo/item-1234
[@DIR_SOURCE@]	Returns the current Item's subfolder (source folder)	hires
[@DIR_TARGET@]	Returns the current Item's subfolder (target folder)	hires
[@DIR_TEMP@]	Returns the path of the temp folder for the currently processing Item.	/tmp/inbox-item-1234
[@FILE_IN@]	A filename used in the current Item process (in/source)	hires/item-1234.wav
[@FILE_OUT@]	A filename used in the current Item process (out/target)	/mnt/archive/item-1234/hires/item-1234.wav
[@FILENAME@]	A filename used in the current Item process (without path)	item-1234.wav
[@HASHCODE@]	A hashcode	72e96e9b88351cd233fa25d23f46e68d
[@HASHTYPE@]	A hashcode type (md5, sha1, crc, etc)	md5
[@HOUR@]	System clock: Hour	6
[@LOGFILE@]	Name of a logfile	/mnt/inbox/log/item-1234.log
[@MINUTE@]	System clock: Minute	43
[@MONTH@]	System clock: Month	5
[@OPTIONS@]	Options/arguments (provided to e.g. external commands)	
[@TASK_LABEL@]	Label of task	
[@TASK_NAME@]	Classname of task	
[@YEAR@]	System clock: Year	2017

4.1.8. Default Workflow

If the Tasklist (= Workflow) is not specified in the configuration file of the inbox, the default Workflow will be executed. Changes to the workflow order may have severe impact, as some Tasks are depending on others running before them. The default Workflow and execution order is:

Step	Task
1	FilesWait
2	DirListCSV
3	Clean Filenames
4	PreProcs
5	FilesValid
6	FilesMustExist
7	HashGenerate
8	HashSearch
9	CopyToTarget
10	HashValidate
11	RenameTarget
12	HashOutput
13	PostProcs

4.1.9. Tasks

This section describes existing Tasks that can be used in CIN Workflows and some example log output on the default log level. Please note that the output you get when running CIN manually in the shell might be different from the default log level, so the log files might hold more or less information in comparison.

4.1.9.1. FilesWait

Due to the nature of this Task, it has to be the first Task to run. It checks if there is any FILES_WAIT[] configuration in the Inbox settings. If not, the Workflow continues. If there are FILES_WAIT[] settings, FilesWait will check if the required files are already present and continue the Workflow if this is the case. If not, the Item is being postponed. Please note: This is not an error State.

This is very handy in cases where an Item is incomplete for a longer period of time, e.g. the media file (.wav) is present, but the metadata (some .xml or text file) will be added later by an operator. The Item is not considered ready for processing until certain files are present.

Logfile example:

```
=====
Executing task #1: 'FilesWait'...
=====
(TEST-123456): Task 'FilesWait' ran successfully.
```

4.1.9.2. DirListCSV

This Task takes a 'Snapshot' of the Items directory and file structure. This is very useful to preserve information of the exact structure of the data before processing by CIN started. Popular example is to keep this information as part of the provenance metadata.

The filename and path for the stored .csv file can be configured with the Inbox parameter DIRLIST_FILE.

Logfile example:

```
=====
Executing task #2: 'DirListCSV'...
=====
Creating directory listing for '/mnt/inbox/in_progress/TEST-123456'...
Wrote directory listing to '/mnt/inbox/in_progress/TEST-123456/metadata/TEST-123456-20170116-
list.csv'.
Task 'DirListCSV' is marked as 'once per Item'. Applied only to '.'.
(TEST-123456): Task 'DirListCSV' ran successfully.
```

4.1.9.3. Clean Filenames

This Task is for streamlining file and path names. It can be configured to be more or less strict, using the configuration parameter CLEAN_SOURCE[]. Most characters that are considered invalid will be mapped to an underscore (). *For Details on the mapping and possible values, please see [_Naming limitations](#).*

CLEAN_SOURCE[] is an array, allowing for multiple configuration lines.

Logfile example:

```
=====
Executing task #3: 'CleanFileNames'...
=====
(TEST-123456): Task 'CleanFileNames' ran successfully.
```


4.1.9.4. PreProcs

This task can be used to execute any user-defined commands. As the word "PRE" in the name indicates, these commands are intended to be executed before the Item is copied to its final destination (See Task *CopyToTarget*). The commands will be executed in the order they are configured in the array PREPROCS[].

All pre-processing scripts must return a positive status for this Task to run successfully.

⚠ Add PreProc Example

Logfile example:

```
=====
Executing task #4: 'PreProcs'...
=====
(TEST-123456): Task 'PreProcs' ran successfully.
```

4.1.9.5. FilesValid

This Task checks if files in the configured folders are valid or invalid, according to the configuration options CONF_FILES_VALID and CONF_FILES_INVALID. These parameters support wildcards, e.g. '.wav', '.mp3'. The following two cases will be tested:

- Check for invalid files matching 'FILES_INVALID': Returns "true" if the folder contains files matching patterns in 'FILES_INVALID', false if not.
- Check for non-valid files not-matching 'FILES_VALID': Returns "true" if the folder contains files that do NOT match the patterns in 'FILES_VALID', false if not.

Logfile example:

```
=====
Executing task #5: 'FilesValid'...
=====
Folder 'hires' contains only valid files (Matching '*.wav'). Good.
Folder 'lowres' contains only valid files (Matching '*.mp3'). Good.
Folder 'metadata' contains only valid files (Matching '*.xml, *.csv'). Good.
(TEST-123456): Task 'FilesValid' ran successfully.
```

4.1.9.6. FilesMustExist

For each configuration section that contains the parameter MUST_EXIST, a check is done to see if the required files/folders are existing. This parameter supports wildcards and is an array, thus allows multiple entries per configuration section.

Logfile example:

```

=====
Executing task #6: 'FilesMustExist'...
=====
Folder '.' contains 1 files/subfolders that match 'hires'. Good.
Folder '.' contains 1 files/subfolders that match 'lowres'. Good.
Folder 'hires' contains 1 files/subfolders that match '*.wav'. Good .
Folder 'lowres' contains 1 files/subfolders that match '*.mp3'. Good.
(TEST-123456): Task 'FilesMustExist' ran successfully.

```

4.1.9.7. HashGenerate

Hash files are generated for every incoming file and either stored one hash-file per file or one hash-file for all files in a directory. Configuration for this Task consists of several parameters. Additionally, this Task checks for any hash duplicates: If files in the source have identical hashes, this either means there are duplicates or danger of possible hash collision.

Logfile example:

```

=====
Executing task #7: 'HashGenerate'...
=====
Generating hashcodes for files in '/mnt/inbox/in_progress/TEST-123456'...
Generating hashcodes for files in '/mnt/inbox/in_progress/TEST-123456/hires'...
Writing hash file '/tmp/cinbox-inbox_test_1/test-123456/hires/250Hz_44100Hz_16bit_05sec.wav.md5'...
Hashcode for 1 file(s) generated.
Generating hashcodes for files in '/mnt/inbox/in_progress/TEST-123456/lowres'...
Writing hash file '/tmp/cinbox-inbox_test_1/test-123456/lowres/250Hz_44100Hz_16bit_05sec.mp3.md5'...
Hashcode for 1 file(s) generated.
Generating hashcodes for files in '/mnt/inbox/in_progress/TEST-123456/metadata'...
Writing hash file '/tmp/cinbox-inbox_test_1/test-123456/metadata/TEST-123456-20170116-list.csv.md5'...
Hashcode for 1 file(s) generated.
(TEST-123456): Task 'HashGenerate' ran successfully.

```

4.1.9.8. HashSearch

For files matching 'HASH_MUST_EXIST[]', this task searches for existing hashes in files matching 'HASH_SEARCH[]' in the source folder in order to compare the current hashes against it. This allows to validate already existing hashes to document the chain of file transfer. It must be run **after** the Task TaskHashGenerate, because the hashes are read from the temp-files created in that Task. All hashes are generated on all files on purpose to compare against possible existing hashes to ensure file integrity at this point.

If HashSearch is configured to look for must-exist hashes and can't find them, it will put the workflow into error and also delete the temporary hash files. This is necessary to make sure that if files are modified and the workflow is reset, no wrong hashes from the temporary files will be used.

Logfile example:

```
=====
Executing task #8: 'HashSearch'...
=====
Searching existing hashcodes for files in '/mnt/inbox/in_progress/TEST-123456'...
No must exists set. Fine.
Searching existing hashcodes for files in '/mnt/inbox/in_progress/TEST-123456/hires'...
No must exists set. Fine.
Searching existing hashcodes for files in '/mnt/inbox/in_progress/TEST-123456/lowres'...
No must exists set. Fine.
Searching existing hashcodes for files in '/mnt/inbox/in_progress/TEST-123456/metadata'...
No must exists set. Fine.
(TEST-123456): Task 'HashSearch' ran successfully.
```

4.1.9.9. CopyToTarget

This Task checks the section configurations for how to treat files and folders and copy them to the **target location**, e.g. An already existing directory structure is expected (update) or nothing is expected at all (create). For files, there can also be configured 'create_or_update' to handle both cases. A condition mismatch is treated as "STATUS_PBC" (Problem But Continue): This allows an operator to clear all reported warnings/errors before resetting the Item, as the task will try to process all sub-folders and report all possibly mismatching entries.

The actual copy itself is done using 'rsync' to ensure reliable data transfer. This also generates information on transfer times and throughput. At this point, the files and directories may still have their temporary names.

Logfile example:

```
=====
Executing task #9: 'CopyToTarget'...
=====
Creating folder '/mnt/archive/temp_TEST-123456'...
No files to copy in '/mnt/inbox/in_progress/TEST-123456'. Fine.
Creating folder '/mnt/archive/temp_TEST-123456/hires'...
Copying '/mnt/inbox/in_progress/TEST-123456/hires/250Hz_44100Hz_16bit_05sec.wav' to
'/mnt/archive/temp_TEST-123456/hires/250Hz_44
250Hz_44100Hz_16bit_05sec.wav
M          32,768   7%   0.00kB/s   0:00:00 M          444,524 100%   30.21MB/s   0:00:00
(xfr#1, to-chk=0/1)
1/1 file(s) copied from '/mnt/inbox/in_progress/TEST-123456/hires' to '/mnt/archive/TEST-
123456/hires'.
Creating folder '/mnt/archive/temp_TEST-123456/lowres'...
Copying '/mnt/inbox/in_progress/TEST-123456/lowres/250Hz_44100Hz_16bit_05sec.mp3' to
'/mnt/archive/temp_TEST-123456/lowres/250Hz_
```

```

250Hz_44100Hz_16bit_05sec.mp3
M          32,768  40%    0.00kB/s    0:00:00  M          80,666 100%   45.68MB/s    0:00:00
(xfr#1, to-chk=0/1)
1/1 file(s) copied from '/mnt/inbox/in_progress/TEST-123456/lowres' to '/mnt/archive/TEST-
123456/lowres'.
Creating folder '/mnt/archive/temp_TEST-123456/metadata'...
Copying '/mnt/inbox/in_progress/TEST-123456/metadata/TEST-123456-20170116-list.csv' to
'/mnt/archive/temp_TEST-123456/metadata/TE
TEST-123456-20170116-list.csv
M          822 100%    0.00kB/s    0:00:00  M          822 100%    0.00kB/s    0:00:00
(xfr#1, to-chk=0/1)
1/1 file(s) copied from '/mnt/inbox/in_progress/TEST-123456/metadata' to '/mnt/archive/TEST-
123456/metadata'.
(TEST-123456): Task 'CopyToTarget' ran successfully.

```

4.1.9.10. HashValidate

All hashes will be validated against the files in the final target path to ensure all files have been transferred bit-proof. This requires the Task 'HashGenerate' to be run before.

Logfile example:

```

=====
Executing task #11: 'HashValidate'...
=====
Checking target folder '/mnt/archive/TEST-123456'...
Checking target folder '/mnt/archive/TEST-123456/hires'...
Target '/mnt/archive/TEST-123456/hires/250Hz_44100Hz_16bit_05sec.wav' is valid.
Checking target folder '/mnt/archive/TEST-123456/lowres'...
Target '/mnt/archive/TEST-123456/lowres/250Hz_44100Hz_16bit_05sec.mp3' is valid.
Checking target folder '/mnt/archive/TEST-123456/metadata'...
Target '/mnt/archive/TEST-123456/metadata/TEST-123456-20170116-list.csv' is valid.
(TEST-123456): Task 'HashValidate' ran successfully.

```

4.1.9.11. RenameTarget

Takes care to rename the temporary folder on the target path to their final name and merge everything into the final destination.

Logfile example:

```

=====
Executing task #10: 'RenameTarget'...
=====
Creating folder '/mnt/archive/TEST-123456'...
Merging files from '/mnt/archive/temp_TEST-123456' into '/mnt/archive/TEST-123456'...
Creating folder '/mnt/archive/TEST-123456/hires'...

```

```

Merging files from '/mnt/archive/temp_TEST-123456/hires' into '/mnt/archive/TEST-123456/hires'...
Moving '/mnt/archive/temp_TEST-123456/hires/250Hz_44100Hz_16bit_05sec.wav' to
'/mnt/archive/TEST-123456/hires/250Hz_44100Hz_16bit
Creating folder '/mnt/archive/TEST-123456/lowres'...
Merging files from '/mnt/archive/temp_TEST-123456/lowres' into '/mnt/archive/TEST-123456/lowres'...
Moving '/mnt/archive/temp_TEST-123456/lowres/250Hz_44100Hz_16bit_05sec.mp3' to
'/mnt/archive/TEST-123456/lowres/250Hz_44100Hz_16b
Creating folder '/mnt/archive/TEST-123456/metadata'...
Merging files from '/mnt/archive/temp_TEST-123456/metadata' into '/mnt/archive/TEST-123456/metadata'...
Moving '/mnt/archive/temp_TEST-123456/metadata/TEST-123456-20170116-list.csv' to
'/mnt/archive/TEST-123456/metadata/TEST-123456-2
Task 'RenameTarget' is recursive itself. Applied only to '..'.
(TEST-123456): Task 'RenameTarget' ran successfully.

```

4.1.9.12. HashOutput

This Task takes care of writing the generated source-hashes to the final target. Format, location, etc. can be set in the configuration sections. Depending on configuration, existing hashcode files with the same name will be overwritten (CONF_HASH_OUTPUT = file) or merged with the new hashcode lines (CONF_HASH_OUTPUT = folder). The hashcode format must be identical, otherwise the resulting files will not work for automatic validation using tools like md5sum and the automatic removal of duplicates will also not work. The three formats supported are:

Format	Hashline	Example
gnu	[@HASHCODE@] [@FILENAME@]\n	b9555c2ddbed44072fdb558e476a43c5 TEST-123456-20170126-list.csv
win	[@HASHCODE@] [@FILENAME@]\r\n	b9555c2ddbed44072fdb558e476a43c5 TEST-123456-20170126-list.csv
macos	[@FILENAME@] [@HASHCODE@]\n	TEST-123456-20170126-list.csv b9555c2ddbed44072fdb558e476a43c5

Logfile example:

```

=====
Executing task #12: 'HashOutput'...
=====
Folder '/mnt/archive/TEST-123456'...
Folder '/mnt/archive/TEST-123456/hires'...
Folder '/mnt/archive/TEST-123456/lowres'...
Folder '/mnt/archive/TEST-123456/metadata'...
(TEST-123456): Task 'HashOutput' ran successfully.

```

4.1.9.13. PostProcs

In case some post-processing is required, the post-processor scripts will run in the order configured in the `POSTPROCS[]` array. All post-processing scripts must return a positive status for this Task to run successfully. The task `PostProcs` is identical to *PreProcs*, except that it is intended to be ran after *CopyToTarget*.

Logfile example:

```
[2017-02-04T16:46:28] N =====
[2017-02-04T16:46:28] N Executing task #13: 'PostProcs'...
[2017-02-04T16:46:28] N =====
[2017-02-04T16:46:28] I Running task 'PostProcs' on folder: '.'
[2017-02-04T16:46:28] N Running 1 post-processing scripts in folder '.' ...
[2017-02-04T16:46:28] N Executing post-processor #1:
'/home/cin/cinbox/bin/plugins/mailnotification.sh'
[2017-02-04T16:46:28] N =====
```

4.1.9.14. LogFileCopy

This task enables support to migrate the generated logfile at the latest point possible (so it contains the maximum of information) to an archive path. The actual Common-Inbox machine will hold a few more lines, stating if the last copy was a success. `LogFileCopy` is an optional Task and not part of the default workflow, which has to be thought of when configuring an inbox.

Logfile example (cv-syntax):

```
"2018-08-01T17:35:29+02:00";"/mnt/inbox_mthk_audio/in_progress/VX-09167";"ADD";"N";"auto-cinbox";"=====
"2018-08-01T17:35:29+02:00";"/mnt/inbox_mthk_audio/in_progress/VX-09167";"ADD";"N";"auto-cinbox";"Executing task #11: 'LogFileCopy'..."
"2018-08-01T17:35:29+02:00";"/mnt/inbox_mthk_audio/in_progress/VX-09167";"ADD";"N";"auto-cinbox";"=====
"2018-08-01T17:35:29+02:00";"/mnt/inbox_mthk_audio/in_progress/VX-09167";"ADD";"I";"auto-cinbox";"Running task 'LogFileCopy' on folder: '.'"
"2018-08-01T17:35:29+02:00";"/mnt/inbox_mthk_audio/in_progress/VX-09167";"ADD";"N";"auto-cinbox";"Logfile target: /mnt/dlp-storage/part2/audio/VX/09/VX-09167/vx-09167.cv"
"2018-08-01T17:35:29+02:00";"/mnt/inbox_mthk_audio/in_progress/VX-09167";"ADD";"N";"auto-cinbox";"Copying logfile from '/mnt/inbox_mthk_audio/in_progress/VX-09167.log' to '/mnt/dlp-storage/part2/audio/VX/09/VX-09167/vx-09167.cv'."
```

4.1.10. Custom Tasks

Common-Inbox offers support for custom created Tasks. The skeleton example can be found in `/cinbox/bin/tasks/TaskSkeleton.php` and contains more detailed information on how to implement new types of Tasks. This is only meant for very advanced users. When the custom Task script has been created, the Workflow can be modified using the `TASKLIST[]` parameter. If the Workflow is changed, the complete new Tasklist has to be configured, e.g.

```
TASKLIST[] = FilesWait
TASKLIST[] = DirListCSV
TASKLIST[] = Clean_Filenames
TASKLIST[] = PreProcs
TASKLIST[] = CustomTask          <----
TASKLIST[] = FilesValid
TASKLIST[] = FilesMustExist
TASKLIST[] = HashGenerate
TASKLIST[] = CopyToTarget
TASKLIST[] = HashValidate
TASKLIST[] = RenameTarget
TASKLIST[] = HashOutput
TASKLIST[] = PostProcs
```

4.1.11. Task States - Return Codes

Return Code	Task State	Description
0	STATUS_UNDEFINED	Undefined. This is the default, unless a task changes it.
1	STATUS_RUNNING	Undefined, but indicates that the task is currently in progress.
5	STATUS_DONE	Success! This means the task completed successfully.
6	STATUS_PBCT	Problem But Continue Task: the TASK may continue, the workflow will then stop.
7	STATUS_PBC	Problem But Continue: there is a problem but the workflow will continue.
10	STATUS_ERROR	An error occurred. Execution is aborted as soon as possible in this case.
11	STATUS_CONFIG_ERROR	This occurs if at least one configuration option was invalid.
15	STATUS_SKIPPED	Indicates that a TASK was skipped.

4.1.12. Task Modes

There are two basic operation modes for Tasks:

Task Mode	Description	Default
ONCE_PER_ITEM	Run this task only once per item.	False
IS_RECURSIVE	False: One task = one folder. True: The task performs actions on subfolders too.	False

4.2. Configuration File: cinbox.ini

Each Inbox requires a configuration file. It must be named 'cinbox.ini' and reside in the root of the Inbox directory. They contain specific settings for the inbox and how to process expected and unexpected data. Furthermore, custom configuration settings can be defined for finer granulation. Sections names are in square brackets and there are four reserved names: [__INBOX__], [__DEFAULT__], [__UNDEFINED__] and [.] - all custom configuration happens in their respective directory configuration blocks. Please check the section documentation below to see what settings are possible or required.

4.2.1. Inbox Configuration Options

Option	Description	Type	Value(s)	Example / Default	Required
BUCKET_SCRIPT	A plug-in script that builds output paths from the Item ID and returns them as [@BUCKET@] to be used later in the target path configuration.	String	Path/Filename	/opt/cinbox/plugins/objid_to_path.sh	optional
COOLOFF_FILTERS[]	By default, all available information about a file is used when calculating the cool-off time to ensure high reliability. In rare cases, some of that might need to be filtered, e.g. when using certain types of storage technology. Allows multiple lines for multiple filters. Available filters can be found here .	String	dev, ino, mode, nlink, uid, gid, rdev, size, atime, mtime, ctime, blksize, blocks	ino	optional
COOLOFF_TIME	To prevent issues with large files (e.g. Hires video) or slower transport methods (e.g. FTP transfer), Common-Inbox will write a temporary file for every Item it discovers in the /todo directory. On the next iteration, information from this file is compared to the status of the Item data. This ensures a most accurate cool-off timer. The temp file that holds this information can be found in /tmp/inbox_name/ITEM_ID/item_changelog.txt	Num	Minutes	5	optional
DIRLIST_FILE	First in the process, the incoming folder and file structure will be written to this file, before anything else happens. This ensures that at any point, it is clear how the incoming data looked like at the begin of the process. This should be a relative path inside the Item directory.	String	Path/Filename	metadata/[@ITEM_ID@]-list.csv	mandatory
INBOX_NAME	A human readable name for the Inbox.	String	Alphanumeric only	Video Testing	mandatory
ITEM_ID_VALID[]	Only Items with an ID matching the regular expression filter will be handled. This filter is applied to the /todo sub-directory of the inbox. Allows multiple lines for multiple filters. Very useful tools to plan regular expression are: http://www.phpiveregex.com/ and http://regexr.com/	String	Regular Expression	/^w{2}-d{10}\$/i	mandatory
ITEMS_AT_ONCE	How many items will be handled per iteration. If many small Items are imported (e.g. mp3s), a higher number can be set. For large file Items (e.g. video), a lower number might be better.	Num	Items	1	optional
KEEP_FINISHED	Keep finished Items in the Inbox sub-directory /done before garbage collection after the configured time. Value in days. If '0' is configured, files get deleted immediately at the next garbage collection.	Num	Days	1	optional
MAX_FOLDER_DEPTH	Maximum recursion depth in the Inbox	Num	Directory depth	5	optional
MOVE_LOGFILES	Defines if logfiles should be written to the logs sub-directory of the Inbox (0), or if the logfiles are moved with the Items (1). If not set, the default (1) will be used.	Num	Boolean	1	optional
MUST_EXIST[]	Configure a subdirectory or file that is expected. Use multiple lines for multiple subdirectories or files. Detail configuration for subdirectories can be specified in their own respective configuration section (e.g. [hires]).	String	Filename, Filemask, directory name	hires	optional
TARGET_FOLDER	Sets the target folder to write files to. Unless configured otherwise, subdirectories will follow their parents to the target path.	String	Path	/mnt/archive	mandatory
UPDATE_FOLDERS	How to handle existing folders. 'create' expects that no target folders are existing yet, while 'update' expects the target folder structure to exist.	String	create, update	create	optional

4.2.2. Cool-off mechanism

To prevent issues with large files (e.g. HiRes video), slow transport methods (e.g. FTP transfer) and network or disk I/O limits, Common-Inbox will write a temporary file containing a change log for every Item it discovers. On every iteration, it will check for new Items and compare data on Items it is already aware of. If a new one is found, another temporary directory will be created in /tmp. There, a temp file will be created that holds information on discovered files. With this information (extracted by using the stat() function from PHP), the cool-off timer can be calculated with a high reliability (compared to simply using the atime, mtime or ctime given by the file system).

This also means that Items found for the first time will never be processed in the same iteration, but soonest on the next one, as only then a stat temp file exists for comparison. If the file has no value changed for the configured amount of time, it is ready for processing. The temp file that holds this information can be found in /tmp/inbox_name/ITEM_ID/item_changelog.txt

When using certain storage technologies, some of those `stat()` file values might change, even tho the file is all ready to be processed. This might also happen on some network share methods. In this case, it is possible to filter the bogus element(s) using the `COOLOFF_FILTERS[]` configuration option.

Filter name	Description
dev	Device number
ino	Inode number
mode	Inode protection mode
nlink	Number of links
uid	UserID of owner
gid	GroupID of owner
rdev	Device type, if inode device
size	Size in bytes
atime	Time of last access (Unix timestamp)
mtime	Time of last modification (Unix timestamp)
ctime	Time of last inode change (Unix timestamp)
blksize	Blocksize of filesystem IO
blocks	Number of 512-byte blocks allocated

4.2.3. [__INBOX__]

This section contains the configuration and information for the Inbox, e.g. the Name, garbage collection settings and processing throttle.

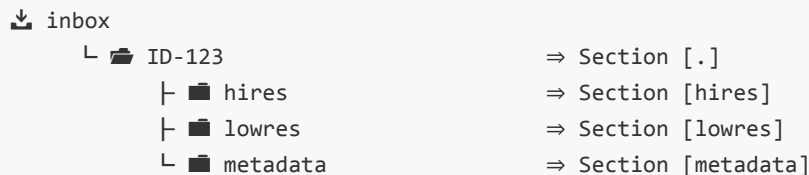
Configuration section example:

```
[__INBOX__]
INBOX_NAME = Testbox
PAUSE_TIME = 60
ITEMS_AT_ONCE = 2
KEEP_FINISHED = 0
WAIT_FOR_ITEMS = 10
COOLOFF_TIME = 5
COOLOFF_FILTERS[] = ino
```

4.2.4. [Directory] Sections

There are three reserved section names: `[.]`, `[_DEFAULT_]` and `[_UNDEFINED_]`. These hold information on how to proceed with different scenarios by setting the default behavior for encountered directories.

All other directory configuration sections can be customized by the user to define the target structure. Ideally, the import folder structure is known and the desired file handling can be configured in detail. A typical example could be for an Item with the ID '123':



In this case, the directory holds the data for the Item ID-123, but it is spread over files in sub-directories. If the structure is expected to always look the same, rules can now be set in sections called `[hires]`, `[lowres]` and `[metadata]`, additionally to the `[.]` section. This gives a lot of control over how files in each sub-directory will be handled.

⚠ Important:

If not explicitly configured, directories will always follow their parents relatively to the new target path! The configuration provided by either the `[.]` or `[_DEFAULT_]` section must contain the archive base path, for subsequent directories it is important to provide a relative path if they should follow their parent directory to the new archive path, or alternatively an absolute path if some files are meant to go to a different storage.

Relative Path Example:

- `TARGET_PATH` in `[.]` is set to `/mnt/archive/[@ITEM_ID@]`
- `TARGET_PATH` in `[hires]` is set to `'hires_audio'`

The incoming file `/inbox/ID-123/hires/id-123.wav` will be treated with the setting in `[hires]` and take the base path from its parent `[.]` - the result is that the file will land on `/mnt/archive/ID-123/hires_audio/id-123.wav`:



Absolute Path Example:

- `TARGET_PATH` in `[_DEFAULT_]` is set to `/mnt/archive/[@ITEM_ID@]`
- `TARGET_PATH` in `[lowres]` is set to `'/mnt/cache/mp3'`

The incoming file `/inbox/ID-123/lowres/id-123.mp3` will be treated with the setting in `[lowres]` and since an absolute path is specified, it will ignore the parent base path given in the `[__DEFAULT__]` section and the file will be written to `/mnt/cache/mp3/id-123.mp3`:

```

  ↓ inbox
    ↳ ID-123
      ↳ lowres
        ↳ id-123.mp3      ⇒      /mnt/cache/mp3/id-123.mp3
```

Configuration section example (relative path):

```
[HIRES]
TARGET_FOLDER = hires/[@ITEM_ID@]
FILES_VALID[] = *.wav
FILES_VALID[] = *.xml
MUST_EXIST[] = *.wav
MUST_EXIST[] = *.xml
HASH_SEARCH[] = *.xml
HASH_MUST_EXIST[] = *.wav
```

4.2.5. Directory Configuration options

Option	Description	Type	Value(s)	Example / Default	Required
CLEAN_SOURCE[]	Replaces characters with an underscore or other meaningful replacements. Multiple lines can be configured to handle different cases. For replacement mapping, please check Naming Limitations.	String	whitespace, slashes, illegal, picky, quotation, quotation2, umlauts, brackets	illegal	optional
COPY_EXCLUDE[]	Defines which files should not be copied to the archive destination. Allows filenames, file masks and wildcards. Multiple lines for multiple settings.	String	Filemask, Filename	MD5SUMS*	optional
FILES_INVALID[]	Files matching this are not processed. Supports wildcards. Set multiple lines for more than one type.	String	Filemask, Filename	*.doc	optional
FILES_VALID[]	Files matching are valid to be processed. Supports wild cards. Set multiple lines for more than one type.	String	Filemask, Filename	*.avi	optional
FILES_WAIT[]	If configured, this setting will keep Items in the 'todo' folder of the Inbox until all configured files are present, e.g. the audio material is already there but it might take a while until an operator adds the required metadata file. Multiple lines for multiple file masks.	String	Filemask, Filename	*.xml	optional
HASH_FILEFORMAT	What format to store the hashes in.	String	gnu, windows, macos	GNU	optional
HASH_FILENAME	Filename for the hash files.	String	Filename, Path	MD5SUMS	optional
HASH_MUST_EXIST[]	If a file exists, there must be a checksum for it. Allows multiple lines for multiple files or file types.	String	Filename, Filemask	*.avi	optional
HASH_OUTPUT	Defines if there should be one hash file for each single file, or one hash file for all files in a directory.	String	file, folder	folder	optional
HASH_SEARCH[]	Which file to check for existing hashes. Can also be configured with a wild card. Supports multiple lines for different files to search in.	String	Filename, Filemask	*.md5	optional
HASH_TYPE	What type of hashing algorithm should be used. Should work with all hash types supported by the PHP installation.	String	md5, sha1, crc32, ... full List at http://php.net/manual/de/function.hash.php	md5	optional
LOG_COPY_DIR	Specifies the directory to write the Item log to.	String	Path	[@DIR_TARGET@]	optional
LOG_COPY_NAME	Sets the name for the Item log file that is written to the archive.	String	Filename	[@ITEM_ID@].cv	optional
MUST_EXIST[]	Configure a subdirectory or file that is expected. Use multiple lines for multiple subdirectories or files. Detail configuration for subdirectories can be specified in their own respective configuration section (e.g. [hires]).	String	Filename, Filemask, directory name	hires	optional
POSTPROCS[]	Call the specified post-processing scripts. Multiple lines can be specified for multiple scripts.	String	Path/Filename	/opt/cinbox/plugins/postproc.sh	optional

Option	Description	Type	Value(s)	Example / Default	Required
PREPROCS[] PREPROCS2[] PREPROCS3[]	Call the specified pre-processing script. Multiple lines can be specified for multiple scripts. A special option is to use PREPROCS2[] and PREPROCS[3] to be able and call more pre-processor scripts at different steps in the workflow.	String	Path/Filename	/opt/cinbox/plugins/preproc.sh	optional
TARGET_FOLDER	Sets the target folder to write files to. Unless configured otherwise, subdirectories will follow their parents to the target path.	String	Path	/mnt/archive	mandatory
UPDATE_FILES	How to handle existing files. 'create' expects that no target files are existing yet. 'update' expects the target files to exist, and 'create_or_update' works in both cases.	String	create , update, create_or_update	create	optional
UPDATE_FOLDERS	How to handle existing folders. 'create' expects that no target folders are existing yet, while 'update' expects the target folder structure to exist.	String	create, update	create	optional

4.2.6. [__DEFAULT__]

This section is for configuring the defaults for several operations: How to handle incoming data, generate filenames, target path, handle hashes and the treatment of already existing files and folders. The default settings are taken if nothing else is configured in a directory specific configuration block later in the configuration file. All parameters that are valid in *Directory Sections* are also valid for the [__DEFAULT__] section.

Configuration section example:

```
[__DEFAULT__]
HASH_TYPE = md5
HASH_OUTPUT = folder
HASH_FILENAME = MD5SUMS.txt
HASH_FILEFORMAT = GNU
UPDATE_FOLDERS = create
UPDATE_FILES = create
COPY_EXCLUDE[] = MD5SUMS
COPY_EXCLUDE[] = *.md5
```

4.2.7. [__UNDEFINED__]

Optional configuration section. Directories encountered that have no configuration block are treated according to the configuration in this section if the DEFAULT behavior is not wanted. All parameters that are valid in *Directory Sections* are also valid for the [__UNDEFINED__] section.

Configuration section example:

```
[__UNDEFINED__]  
TARGET_FOLDER = unknown_data/
```

4.2.8. [.]

This "Dot" section represents the source directory of an Item in the inbox, e.g. in /myinbox/ABC-123, the settings will apply to the directory ABC-123. A recursion will check for further sub-directories and handle them according to additional configurations found in the [__DEFAULT__], [__UNDEFINED__] or [Directory Sections]. The default behavior will copy the existing structure of the Item to the target path and check hashes for all files to verify operations. Child directories will use their parent directories settings if they do not find a configuration suitable. All parameters that are valid in [Directory Sections] are also valid for the [.] section.

Configuration section example:

```
[.]  
TARGET_FOLDER = /mnt/archive  
MUST_EXIST[] = HIRES  
MUST_EXIST[] = LOWRES
```

4.2.9. All Configuration Options

This is an overview about all existing configuration options. Options with square brackets behind them are array options and allow for multiple lines, e.g. CLEAN_SOURCE[] used like the following example will configure both umlaut and illegal characters to be handled:

```
CLEAN_SOURCE[] = umlauts  
CLEAN_SOURCE[] = illegal
```

⚠ Important:

Not all options work in every configuration section. Please consult the section specific documentation and example configurations: *Directory Configuration Options* and *Inbox Configuration Options* .

Option	Description	Type	Value(s)	Example / Default	Required
BUCKET_SCRIPT	A plug-in script that builds output paths from the Item ID and returns them as [@BUCKET@] to be used later in the target path configuration.	String	Path/Filename	/opt/cinbox/plugins/objid_to_path.sh	optional
CLEAN_SOURCE[]	Replaces characters with an underscore or other meaningful replacements. Multiple lines can be configured to handle different cases. For replacement mapping, please check Naming Limitations.	String	whitespace, slashes, illegal, picky, quotation, quotation2, umlauts, brackets	illegal	optional
COOLOFF_FILTERS[]	By default, all available information about a file is used when calculating the cool-off time to ensure high reliability. In rare cases, some of that might need to be filtered, e.g. when using certain types of storage technology. Allows multiple lines for multiple filters. Available filters can be found here.	String	dev, ino, mode, nlink, uid, gid, rdev, size, atime, mtime, ctime, blksize, blocks	ino	optional
COOLOFF_TIME	To prevent issues with large files (e.g. Hires video) or slower transport methods (e.g. FTP transfer), Common-Inbox will write a temporary file for every Item it discovers in the /todo directory. On the next iteration, information from this file is compared to the status of the Item data. This ensures a most accurate cool-off timer. The temp file that holds this information can be found in /tmp/inbox_name/ITEM_ID/item_changelog.txt	Num	Minutes	5	optional
COPY_EXCLUDE[]	Defines which files should not be copied to the archive destination. Allows filenames, file masks and wildcards. Multiple lines for multiple settings.	String	Filemask, Filename	MD5SUMS*	optional
DIRLIST_FILE	First in the process, the incoming folder and file structure will be written to this file, before anything else happens. This ensures that at any point, it is clear how the incoming data looked like at the begin of the process. This should be a relative path inside the Item directory.	String	Path/Filename	metadata/[@ITEM_ID@]-list.csv	mandatory
FILES_INVALID[]	Files matching this are not processed. Supports wildcards. Set multiple lines for more than one type.	String	Filemask, Filename	*.doc	optional
FILES_VALID[]	Files matching are valid to be processed. Supports wild cards. Set multiple lines for more than one type.	String	Filemask, Filename	*.avi	optional
FILES_WAIT[]	If configured, this setting will keep Items in the 'todo' folder of the Inbox until all configured files are present, e.g. the audio material is already there but it might take a while until an operator adds the required metadata file. Multiple lines for multiple file masks.	String	Filemask, Filename	*.xml	optional
HASH_FILEFORMAT	What format to store the hashes in.	String	gnu, windows, macos	GNU	optional
HASH_FILENAME	Filename for the hash files.	String	Filename, Path	MD5SUMS	optional
HASH_MUST_EXIST[]	If a file exists, there must be a checksum for it. Allows multiple lines for multiple files or file types.	String	Filename, Filemask	*.avi	optional
HASH_OUTPUT	Defines if there should be one hash file for each single file, or one hash file for all files in a directory.	String	file, folder	folder	optional
HASH_SEARCH[]	Which file to check for existing hashes. Can also be configured with a wild card. Supports multiple lines for different files to search in.	String	Filename, Filemask	*.md5	optional
HASH_TYPE	What type of hashing algorithm should be used. Should work with all hash types supported by the PHP installation.	String	md5, sha1, crc32, ... full List at http://php.net/manual/de/function.hash.php	md5	optional
INBOX_NAME	A human readable name for the Inbox.	String	Alphanumeric only	Video Testing	mandatory
ITEM_ID_VALID[]	Only Items with an ID matching the regular expression filter will be handled. This filter is applied to the /todo sub-directory of the inbox. Allows multiple lines for multiple filters. Very useful tools to plan regular expression are: http://www.phpliveregex.com/ and http://regexr.com/	String	Regular Expression	/^w{2}-d{10}\$/i	mandatory
ITEMS_AT_ONCE	How many items will be handled per iteration. If many small Items are imported (e.g. mp3s), a higher number can be set. For large file Items (e.g. video), a lower number might be better.	Num	Items	1	optional
KEEP_FINISHED	Keep finished Items in the Inbox sub-directory /done before garbage collection after the configured time. Value in days. If '0' is configured, files get deleted immediately at the next garbage collection.	Num	Days	1	optional
LOG_COPY_DIR	Specifies the directory to write the Item log to.	String	Path	[@DIR_TARGET@]	optional
LOG_COPY_NAME	Sets the name for the Item log file that is written to the archive.	String	Filename	[@ITEM_ID@].cv	optional
MAX_FOLDER_DEPTH	Maximum recursion depth in the Inbox	Num	Directory depth	5	optional
MOVE_LOGFILES	Defines if logfiles should be written to the logs sub-directory of the Inbox (0), or if the logfiles are moved with the Items (1). If not set, the default (1) will be used.	Num	Boolean	1	optional
MUST_EXIST[]	Configure a subdirectory or file that is expected. Use multiple lines for multiple subdirectories or files. Detail configuration for subdirectories can be specified in their own respective configuration section (e.g. [hires]).	String	Filename, Filemask, directory name	hires	optional

Option	Description	Type	Value(s)	Example / Default	Required
PAUSE_TIME	How long to pause processing between two Items.	Num	Seconds	60	optional
POSTPROCS[]	Call the specified post-processing scripts. Multiple lines can be specified for multiple scripts.	String	Path/Filename	/opt/cinbox/plugins/postproc.sh	optional
PREPROCS[] PREPROCS2[] PREPROCS3[]	Call the specified pre-processing script. Multiple lines can be specified for multiple scripts. A special option is to use PREPROCS2[] and PREPROCS3[] to be able and call more pre-processor scripts at different steps in the workflow.	String	Path/Filename	/opt/cinbox/plugins/preproc.sh	optional
TARGET_FOLDER	Sets the target folder to write files to. Unless configured otherwise, subdirectories will follow their parents to the target path.	String	Path	/mnt/archive	mandatory
TASKLIST[]	This enables the user to manually configure a Task-list, e.g. if some tasks are not necessary for the desired workflow or special tasks need to be added. If nothing is set, the default Task-list will be executed.	String	Valid Task name (see Tasks)	HashGenerate	optional
UPDATE_FILES	How to handle existing files. 'create' expects that no target files are existing yet. 'update' expects the target files to exist, and 'create_or_update' works in both cases.	String	create , update, create_or_update	create	optional
UPDATE_FOLDERS	How to handle existing folders. 'create' expects that no target folders are existing yet, while 'update' expects the target folder structure to exist.	String	create, update	create	optional
WAIT_FOR_ITEMS	Time to wait for new Items to appear in minutes.	Num	Minutes	1	optional

4.3. Plug-Ins

These can be found in the cinbox/bin/plugins sub-directory. There are also example files for Pre- and Postprocessor scripts. Plugins are normally called in a PREPROCS[] or POSTPROCS[] configuration. By default, the scripts must return '0' if they run successfully. It is possible to use Placeholder Variables like [@ITEM_ID@] to pass data to Plugins.

4.3.1. MailNotification

This script sends out a notification mail as post-processing step of the Common-Inbox. The subject consists of the ItemID and the message body contains the Item logfile. This script requires a working MTA (Mail Transfer Agent) on the machine running CIN. The script will try to auto-detect the 'mail' application. If this does not work, \$MAILER might have to be set manually.

Required arguments:

```
$1 = ItemID - should be passed over from CIN via the [@ITEM_ID@] variable.
$2 = Email address(es) - supports multiple by separating with a coma.
$3 = log path - Inbox log folder.
$4 = Sender name (Useful for multiple Inboxes). No whitespaces or special chars allowed.
```

Config line example:

```
POSTPROCS[] = /home/cin/cinbox/bin/plugins/mailnotification.sh "[@ITEM_ID@]"
"email@your.tld,email2@your.tld" "/mnt/inbox/log" "Common-Inbox_1"
```

5. Running Common-Inbox

5.1. Manual mode

In some cases it might be desired to run manually start CIN to handle a batch and then stop until the next batch is prepared. It is recommended to start CIN in a screen session and connect to that screen to prevent processing issues by hanging shells or loss of internet connectivity:

```
$ screen -AmdS cin; screen -r cin
```

Either cd to the CIN /bin directory or directly call it with the minimum required parameters:

```
$ cd /home/cin/cinbox/bin; ./cinbox.php -i /mnt/inbox
```

or

```
$ /home/cin/cinbox/bin/cinbox.php -i /mnt/inbox
```

Common-Inbox will look for a valid cinbox.ini file in the specified inbox path and start processing according to this configuration. After the set amount of Items (ITEMS_AT_ONCE) to process, it will stop and exit.

5.2. Automatic mode

Starting CIN with the flag '--forever' will result in a permanent check of the inbox folder for new data. This is a 'daemon mode' for continuous processing of all Items that are dropped in the inbox. CIN will periodically check the inbox according to the INBOX configuration.

It is strongly recommended to start CIN in a screen session if automatic mode is desired:

```
$ screen -AmdS cin; screen -S cin -X stuff "/home/cin/cinbox/bin/cinbox.php -i /mnt/inbox --forever\n"
```

To connect to the CIN shell, use the command `$screen -r cin`; To leave the shell (without interrupting processing), press CTRL+A+D.

5.3. Logging

When specified in the start-up parameters (`--log`), Common-Inbox will also write a log following the given path- and filename. This is the log file of the actual engine and not of the Tasks that are handled.

The logs for the tasks have different modes:

- All Item operations are logged into the /log subdirectory of the inbox
- The logfile is moved with the content through the directory structure, e.g. An Item in progress has the log also in the /in_progress subfolder. If an Item encountered an error, both the log and the Item will be in the /error subfolder.

As of version 1.0, there are two different logging styles supported:

- Flat text ('classic', as mostly used in the below example outputs), which is better human-readable
- The 'cv' (curriculum vitae) format, which uses csv syntax for easier machine-reading

Both logging styles contain the same information in a slightly different format.

To chose the cv logging style, you have to use the '--logstyle' flag when running Common-Inbox, e.g.

```
./cinbox.php -i /mnt/inbox --logstyle=cv
```

⚠ Important:

The logfile of the Common-Inbox-Process itself will always be in classic style! The `--logstyle` flag will only modify the behaviour of the ITEM logs.

6. Examples

This section covers some basics of workflow design, use cases with configuration examples and detailed explanation of what is happening in each step by dissecting the log files. Not all functions are covered by the examples but users should get a good idea of what the engine is capable of.

6.1. Example 1: Simple wav+mp3 import

6.1.1. Scenario

A small audio archive has to migrate a lot of audio media files from an old storage to a new one. The integrity of the data must be secured throughout the process and since there is no hashes existing for a majority of the files, the generated hashes shall be stored with the migrated media files for future use in a streamlined manner. Every archive item has a unique signature formed from four letters, a dash and then 6 digits and consists of a directory with a wave and an mp3 file in it. The CIN is reachable over the windows network as a network share, and operators copy Items to migrate to this shares /todo sub-directory.

```
↓ Inbox
  ↳ 📁 todo          ⇐ '📁 TEST-234561' is moved into the 'todo' folder by the user
```

6.1.2. Workflow design

We assume the following initial situation:

- Archive Signatures (=Item IDs) are 4 characters, a dash and then 6 digits (e.g. TEST-234561).
- Pathes are /mnt/inbox/todo for incoming material and /mnt/archive as output base path.
- Each Item is a directory (e.g. /inbox/todo/TEST-234561)
- Each Item contains a hires (wav) and a low-res (mp3) audio file

Thus, the incoming structure is expected to look like this:

```
↓ Inbox
  ↳ 📁 todo
    ↳ 📁 TEST-234561
      ├── 📄 TEST-234561.wav          ⇒ Must exist
      ├── 📄 TEST-234561.mp3        ⇒ Must exist
      └── 📄 checksums.md5           ⇒ Optional
```

Next, we define some rules for handling incoming data:

- There must be a .wav file in hires and a .mp3 file in lowres
- There might be existing hashes in a *.md5 file.
- The structure will be rebuilt 1:1 on the archive path

The expected outcome of an test item on the archive path should look like this:

```
mnt
├─ archive
│   └─ TEST-234561
│       ├── TEST-234561.wav
│       ├── TEST-234561.mp3
│       ├── MD5SUMS
│       └─ TEST-234561-dirlist.csv
```

6.1.3. Configuration

To achieve this, we need some basic configuration done and put the resulting cinbox.ini file into the root of the /mnt/inbox. All configuration sections explained below are within the cinbox.ini file.

The Inbox Section defines the name of the Inbox (Inbox Example 1) and the COOLOFF_TIME. Here, we wait for files that have not been changed for 10 minutes. We also specify to only process one item per iteration and have a pause of 5 seconds before the inbox is checked for new Items (only in automatic mode, else only one run will be done). ITEM_ID_VALID gives the mask what Items are deemed valid to process. The regular expression here expects a pattern of four alphabet characters followed by a dash and a 6 digit code, e.g. 'TEST-234561' would be valid. With DIRLIST_FILE we configure that a snapshot of the incoming directory/file structure is created and saved with the ITEM ID in the filename.

```
[__INBOX__]
INBOX_NAME = Inbox Example 1
COOLOFF_TIME = 10
PAUSE_TIME = 5
ITEMS_AT_ONCE = 1
DIRLIST_FILE = [@ITEM_ID@]-dirlist
ITEM_ID_VALID[] = /^w{4}-d{6}$/i
```

Using the Default Section, we define how to clean filenames using CLEAN_SOURCE. HASH_TYPE and HASH_FILEFORMAT are set so the file created with the name specified in HASH_FILENAME is formatted as desired. HASH_OUTPUT 'folder' setting means that one hash file with the specified HASH_FILENAME will be generated in each processed directory (here only one that contains all files). UPDATE_FOLDERS in create mode means that the folder structure must not already exist on the archive while UPDATE_FILES in create_or_update allows for files to already exist. Every setting in the Default Section governs the default behavior for all further configured Directory Sections.

```

[__DEFAULT__]
CLEAN_SOURCE[] = illegal
HASH_TYPE = md5
HASH_OUTPUT = folder
HASH_FILEFORMAT = gnu
HASH_FILENAME = MD5SUMS
UPDATE_FOLDERS = create
UPDATE_FILES = create_or_update

```

The 'Dot' Section is for the configuration of how to handle the Item root directory. As we expect all files directly in the Item directory (and not in any sub-directories), we specify what files must exist for valid processing. Expecting a wave and an mp3 file, `MUST_EXIST` is thus configured with `'.wav'` and `'.mp3'`. There might be existing hashes, so `HASH_SEARCH` will check for a `*.md5` file in the directory to find them. Finally, the `TARGET_FOLDER` sets where the data will be copied to and verified at. In this case, the base path `/mnt/archive` will have a directory with the Item ID as name, for each new Item processed. With `COPY_EXCLUDE[]` we ignore the optional `*.md5` file that might or might not be present, as we generate and store hashes in the file `'MD5SUMS'`.

```

[.]
TARGET_FOLDER = /mnt/archive/[@ITEM_ID@]
HASH_SEARCH[] = *.md5
MUST_EXIST[] = *.wav
MUST_EXIST[] = *.mp3
COPY_EXCLUDE[] = *.md5

```

6.1.4. Process details

After an Item was processed by CIN with the Example 1 configuration, a log file will be available in the `/logs` sub-directory of the Inbox. Using the below example we'll explain what steps were happening in more detail. The full log file for reference can be found in the download section at the end of this example. We will now dissect the Common-Inbox output and explain what each step did (even more details can be found in the log file):

```

[2017-01-26T16:35:00] A =====
[2017-01-26T16:35:00] A Item: TEST-234561
[2017-01-26T16:35:00] A 2017-01-26T16:35:00
[2017-01-26T16:35:00] A =====
[2017-01-26T16:35:00] N Checking if item 'TEST-234561' is ready for processing...
[2017-01-26T16:35:00] N Item age is: 13 minutes (cooloff time: 10)
[2017-01-26T16:35:00] N Item ID 'TEST-234561' is valid. Good!
[2017-01-26T16:35:00] N Processing item 'TEST-234561'...
[2017-01-26T16:35:00] I Constructing config for folder '.'
[2017-01-26T16:35:00] I Subfolder structure of '/mnt/inbox/in_progress/TEST-234561':
[2017-01-26T16:35:00] I Array
[2017-01-26T16:35:00] I (
[2017-01-26T16:35:00] I     [0] => /mnt/inbox/in_progress/TEST-234561
[2017-01-26T16:35:00] I )

```

The very first step was the Inbox being checked for new Items. An Item has been found, with the ID 'TEST-234561' being valid. The Item was not touched for 13 minutes while the cool-off time is configured to 10 minutes, which means all prerequisites are met and processing of the item can begin.

```
[2017-01-26T16:35:00] N =====
[2017-01-26T16:35:00] N Executing task #1: 'FilesWait'...
[2017-01-26T16:35:00] N =====
[2017-01-26T16:35:00] I Running task 'FilesWait' on folder: '.'
[2017-01-26T16:35:00] I (TEST-234561): Task 'FilesWait' skipped for '.'.
[2017-01-26T16:35:00] N (TEST-234561): Task 'FilesWait' ran successfully.
```

The first task is "FilesWait". As we're not using this feature in this example, the task is skipped on purpose and thus, reports success so the workflow may continue.

```
[2017-01-26T16:35:00] N =====
[2017-01-26T16:35:00] N Executing task #2: 'DirListCSV'...
[2017-01-26T16:35:00] N =====
[2017-01-26T16:35:00] I Running task 'DirListCSV' on folder: '.'
[2017-01-26T16:35:00] N Creating directory listing for '/mnt/inbox/in_progress/TEST-234561'...
[2017-01-26T16:35:00] N File already exists. Will not overwrite '/mnt/inbox/in_progress/TEST-234561/TEST-234561-dirlist.csv'.
[2017-01-26T16:35:00] N Task 'DirListCSV' is marked as 'once per Item'. Applied only to '.'.
[2017-01-26T16:35:00] N (TEST-234561): Task 'DirListCSV' ran successfully.
```

The next Task is "DirListCSV". It created a csv file that contains information on the directory structure and files of the Item at the time of the process starting. This can be useful to know what exactly was incoming, in case later tasks rename folders/files or modify the directory structure. Here we see what happens if an Item was tried to be processed already and reset, as the csv file already exists. This is not a problem, so the task finishes successful.

```
[2017-01-26T16:35:00] N =====
[2017-01-26T16:35:00] N Executing task #3: 'CleanFilenames'...
[2017-01-26T16:35:00] N =====
[2017-01-26T16:35:00] I Running task 'CleanFilenames' on folder: '.'
[2017-01-26T16:35:00] N (TEST-234561): Task 'CleanFilenames' ran successfully.
```

Next in line, the Task "CleanFilenames" checks all directory and filenames for characters that should be changed according to configuration. Nothing was found that required treatment, so the workflow can continue.

```
[2017-01-26T16:35:00] N =====
[2017-01-26T16:35:00] N Executing task #4: 'PreProcs'...
[2017-01-26T16:35:00] N =====
[2017-01-26T16:35:00] I Running task 'PreProcs' on folder: '.'
[2017-01-26T16:35:00] I (TEST-234561): Task 'PreProcs' skipped for '.'.
[2017-01-26T16:35:00] N (TEST-234561): Task 'PreProcs' ran successfully.
```

PreProcs checks now if there are any scripts configured to execute at this point. This is not the case, so the workflow continues.

```
[2017-01-26T16:35:00] N =====
[2017-01-26T16:35:00] N Executing task #5: 'FilesValid'...
[2017-01-26T16:35:00] N =====
[2017-01-26T16:35:00] I Running task 'FilesValid' on folder: '.'
[2017-01-26T16:35:00] I (TEST-234561): Task 'FilesValid' skipped for '.'.
[2017-01-26T16:35:00] N (TEST-234561): Task 'FilesValid' ran successfully.
```

The Task "FilesValid" is not used here and thus, skips so the workflow continues.

```
[2017-01-26T16:35:00] N =====
[2017-01-26T16:35:00] N Executing task #6: 'FilesMustExist'...
[2017-01-26T16:35:00] N =====
[2017-01-26T16:35:00] I Running task 'FilesMustExist' on folder: '.'
[2017-01-26T16:35:00] N Folder '.' contains 1 files/subfolders that match '*.wav'. Good.
[2017-01-26T16:35:00] N Folder '.' contains 1 files/subfolders that match '*.mp3'. Good.
[2017-01-26T16:35:00] N (TEST-234561): Task 'FilesMustExist' ran successfully.
```

There were two "FilesMustExist" configured: It is required that a wav and an mp3 file are existing. The criteria is matched ok, the workflow continues.

```
[2017-01-26T16:35:00] N =====
[2017-01-26T16:35:00] N Executing task #7: 'HashGenerate'...
[2017-01-26T16:35:00] N =====
[2017-01-26T16:35:00] I Running task 'HashGenerate' on folder: '.'
[2017-01-26T16:35:00] I Task 'TaskHashGenerate': temp folder is '/tmp/cinbox-inbox_test_1/test-234561'.
[2017-01-26T16:35:00] N Generating hashcodes for files in '/mnt/inbox/in_progress/TEST-234561'...
[2017-01-26T16:35:00] I Hashing 'TEST-234561-dirlist.csv'...
[2017-01-26T16:35:00] I Hashcode (md5) for 'TEST-234561-dirlist.csv' = 63a3299cebc810bcddd8fcdafb5366b2
[2017-01-26T16:35:00] N Writing hash file '/tmp/cinbox-inbox_test_1/test-234561/TEST-234561-dirlist.csv.md5'...
[2017-01-26T16:35:00] I Hashing 'TEST-234561.mp3'...
[2017-01-26T16:35:00] I Hashcode (md5) for 'TEST-234561.mp3' = 0c200572407d3d8061eb2dad0a5ddb13
[2017-01-26T16:35:00] N Writing hash file '/tmp/cinbox-inbox_test_1/test-234561/TEST-234561.mp3.md5'...
[2017-01-26T16:35:00] I Hashing 'TEST-234561.wav'...
[2017-01-26T16:35:00] I Hashcode (md5) for 'TEST-234561.wav' = 72e96e9b88351cd233fa25d23f46e68d
[2017-01-26T16:35:00] N Writing hash file '/tmp/cinbox-inbox_test_1/test-234561/TEST-234561.wav.md5'...
[2017-01-26T16:35:00] N Hashcode for 3 file(s) generated.
[2017-01-26T16:35:00] N (TEST-234561): Task 'HashGenerate' ran successfully.
```

Checksums are now calculated by the Task "HashGenerate", following the configuration. The created checksums are saved in a temporary location.

```

[2017-01-26T16:35:00] N =====
[2017-01-26T16:35:00] N Executing task #8: 'HashSearch'...
[2017-01-26T16:35:00] N =====
[2017-01-26T16:35:00] I Running task 'HashSearch' on folder: '.'
[2017-01-26T16:35:00] I Task 'TaskHashSearch': temp folder is '/tmp/cinbox-inbox_test_1/test-234561'.
[2017-01-26T16:35:00] N Searching existing hashcodes for files in '/mnt/inbox/in_progress/TEST-234561'...
[2017-01-26T16:35:00] I searchHashCode: Hashtype is 'md5'.
[2017-01-26T16:35:00] I Searching hashcode '63a3299cebc810bcddd8fcda5366b2' in ...
[2017-01-26T16:35:00] I Searching hashcode '0c200572407d3d8061eb2dad0a5ddb13' in ...
[2017-01-26T16:35:00] I Searching hashcode '72e96e9b88351cd233fa25d23f46e68d' in ...
[2017-01-26T16:35:00] N No must exists set. Fine.
[2017-01-26T16:35:00] N (TEST-234561): Task 'HashSearch' ran successfully.

```

Now, "HashSearch" checks its configuration and tries to find existing hashcodes. The hashcodes it tries to match are the ones created one Task earlier. No match is found, but that is also not required, so the workflow can continue.

```

[2017-01-26T16:35:00] N =====
[2017-01-26T16:35:00] N Executing task #9: 'CopyToTarget'...
[2017-01-26T16:35:00] N =====
[2017-01-26T16:35:00] I Running task 'CopyToTarget' on folder: '.'
[2017-01-26T16:35:00] I Task 'TaskCopyToTarget': temp folder is '/tmp/cinbox-inbox_test_1/test-234561'.
[2017-01-26T16:35:00] N Creating folder '/mnt/archive/temp_TEST-234561'...
[2017-01-26T16:35:00] N Copying '/mnt/inbox/in_progress/TEST-234561/TEST-234561-dirlist.csv' to '/mnt/archive/temp_TEST-234561/TEST-234561-dirlist.csv'...
[2017-01-26T16:35:01] N Copying '/mnt/inbox/in_progress/TEST-234561/TEST-234561.mp3' to '/mnt/archive/temp_TEST-234561/TEST-234561.mp3'...
[2017-01-26T16:35:01] N Copying '/mnt/inbox/in_progress/TEST-234561/TEST-234561.wav' to '/mnt/archive/temp_TEST-234561/TEST-234561.wav'...
[2017-01-26T16:35:01] N 3/3 file(s) copied from '/mnt/inbox/in_progress/TEST-234561' to '/mnt/archive/TEST-234561'.
[2017-01-26T16:35:01] N (TEST-234561): Task 'CopyToTarget' ran successfully.

```

The Task "CopyToTarget" is now taking care of copying the files to the archive location. Note that the target still has a temporary name. As all files could be copied successfully, the workflow continues.


```

[2017-01-26T16:35:01] N =====
[2017-01-26T16:35:01] N Executing task #10: 'RenameTarget'...
[2017-01-26T16:35:01] N =====
[2017-01-26T16:35:01] I Running task 'RenameTarget' on folder: '.'
[2017-01-26T16:35:01] I Task 'TaskRenameTarget': temp folder is '/tmp/cinbox-inbox_test_1/test-234561'.
[2017-01-26T16:35:01] N Creating folder '/mnt/archive/TEST-234561'...
[2017-01-26T16:35:01] N Merging files from '/mnt/archive/temp_TEST-234561' into '/mnt/archive/TEST-234561'...
[2017-01-26T16:35:01] N Moving '/mnt/archive/temp_TEST-234561/TEST-234561-dirlist.csv' to '/mnt/archive/TEST-234561/TEST-234561-dirlist.csv'...
[2017-01-26T16:35:01] N Moving '/mnt/archive/temp_TEST-234561/TEST-234561.mp3' to '/mnt/archive/TEST-234561/TEST-234561.mp3'...
[2017-01-26T16:35:01] N Moving '/mnt/archive/temp_TEST-234561/TEST-234561.wav' to '/mnt/archive/TEST-234561/TEST-234561.wav'...
[2017-01-26T16:35:01] N Task 'RenameTarget' is recursive itself. Applied only to '..'.
[2017-01-26T16:35:01] N (TEST-234561): Task 'RenameTarget' ran successfully.

```

"RenameTarget" renames the directory on the target location as configured. All file renaming operations succeeded, the workflow continues.

```

[2017-01-26T16:35:01] N =====
[2017-01-26T16:35:01] N Executing task #11: 'HashValidate'...
[2017-01-26T16:35:01] N =====
[2017-01-26T16:35:01] I Running task 'HashValidate' on folder: '.'
[2017-01-26T16:35:01] I Task 'TaskHashValidate': temp folder is '/tmp/cinbox-inbox_test_1/test-234561'.
[2017-01-26T16:35:01] N Checking target folder '/mnt/archive/TEST-234561'...
[2017-01-26T16:35:01] I Validating target 'TEST-234561-dirlist.csv' against source hash (md5): 63a3299cebc810bcddd8fcdafb5366b2
[2017-01-26T16:35:01] N Target '/mnt/archive/TEST-234561/TEST-234561-dirlist.csv' is valid.
[2017-01-26T16:35:01] I Validating target 'TEST-234561.mp3' against source hash (md5): 0c200572407d3d8061eb2dad0a5ddb13
[2017-01-26T16:35:01] N Target '/mnt/archive/TEST-234561/TEST-234561.mp3' is valid.
[2017-01-26T16:35:01] I Validating target 'TEST-234561.wav' against source hash (md5): 72e96e9b88351cd233fa25d23f46e68d
[2017-01-26T16:35:01] N Target '/mnt/archive/TEST-234561/TEST-234561.wav' is valid.
[2017-01-26T16:35:01] N (TEST-234561): Task 'HashValidate' ran successfully.

```

Hashes are generated again from the files at the target location and compared to the hashes created earlier. All hashes are valid, so the files have been copied successfully and the workflow continues.

```

[2017-01-26T16:35:01] N =====
[2017-01-26T16:35:01] N Executing task #12: 'HashOutput'...
[2017-01-26T16:35:01] N =====
[2017-01-26T16:35:01] I Running task 'HashOutput' on folder: '.'
[2017-01-26T16:35:01] I Task 'TaskHashOutput': temp folder is '/tmp/cinbox-inbox_test_1/test-234561'.
[2017-01-26T16:35:01] N Folder '/mnt/archive/TEST-234561'...
[2017-01-26T16:35:01] I Target hash output: /mnt/archive/TEST-234561/MD5SUMS
[2017-01-26T16:35:01] N (TEST-234561): Task 'HashOutput' ran successfully.

```

Since it is now clear that the hashcodes are valid, they are stored in a file called 'MD5SUMS', as configured. This file will be written to the target path, next to the migrated files.

```
[2017-01-26T16:35:01] N =====
[2017-01-26T16:35:01] N Executing task #13: 'PostProcs'...
[2017-01-26T16:35:01] N =====
[2017-01-26T16:35:01] I Running task 'PostProcs' on folder: '.'
[2017-01-26T16:35:01] I (TEST-234561): Task 'PostProcs' skipped for '.'.
[2017-01-26T16:35:01] N (TEST-234561): Task 'PostProcs' ran successfully.
```

Now the Task "PostProcs" checks if any aftercare scripts need to be executed. None are configured, so the workflow continues.

```
[2017-01-26T16:35:01] N Item 'TEST-234561': 13/13 task(s) processed successfully!
[2017-01-26T16:35:01] I Trying to remove temp folder '/tmp/cinbox-inbox_test_1/test-234561'...
[2017-01-26T16:35:01] I Removed temp folder.
[2017-01-26T16:35:01] I Updating Item timestamp to: 2017-01-26T16:35:01+01:00
```

The workflow has ended successfully. All 13 tasks have been finished, so some garbage collection is done and the migration of data for this Item is complete.

6.2. Example 2: Different target directories and existing checksums

6.2.1. Scenario

A media company has to migrate a lot of audio media files from an old storage to a new one. The integrity of the data must be secured throughout the process and since there is no hashes existing for a majority of the files, the generated hashes shall be stored with the migrated media files for future use in a streamlined manner. Every Item has a unique ID that is created from two characters, a dash and then 10 digits (e.g. CD-1234567890). The CIN is reachable over the windows network as a network share, and operators copy Items to migrate to this shares /todo sub-directory. Further, the mp3 files are not to be copied to the new archive target, but to a cache that is used by a webserver. There must be a metadata XML file in a metadata sub-directory with the Item ID as name (e.g. CD-1234567890.xml).

6.2.2. Workflow design

We assume the following initial situation:

- Archive Signatures (=Item IDs) are 2 characters, a dash and then 10 digits (e.g. CD-1234567890).
- The path for incoming Items is /mnt/inbox
- Each Item is a Directory (e.g. /inbox/todo/CD-1234567890)
- Each Item contains a hires, lowres and metadata sub-directory:

Thus, the incoming structure is expected to look like this:

```

Inbox
└─ todo
   └─ CD-1234567890
      ├── hires
      │   └─ CD-1234567890.wav
      ├── lowres
      │   └─ CD-1234567890.mp3
      └─ metadata
         ├── CD-1234567890.xml
         └─ checksums.md5

```

Next, we define some rules for handling incoming data:

- There must be a .wav file in hires and a .mp3 file in the lowres directory
- The metadata directory must contain an XML file where the name of the file matches the Item ID (e.g. CD-1234567890.xml)
- There might be existing hashes in an .md5 file in the metadata subdirectory
- The data will be put on different archive paths:

```

mnt
└─ archive
   └─ [@ITEM_ID@]
      ├── hires
      └─ metadata

```

```

mnt
└─ cache
   └─ [@ITEM_ID@]

```

The expected outcome of an test item on the target paths should look like this:

```

mnt
└─ archive
   └─ CD-1234567890
      ├── hires
      │   ├── CD-1234567890.wav
      │   └─ MD5SUMS
      └─ metadata
         ├── CD-1234567890.xml
         ├── CD-1234567890-dirlist.csv
         └─ MD5SUMS

```

```

mnt
└─ cache
   └─ CD-1234567890
      ├── CD-1234567890.mp3
      └─ MD5SUMS

```

6.2.3. Configuration

As in Example 1, all configuration sections explained in this example are in the cinbox.ini file.

In the **Inbox section**, the name of the Inbox as well as some timers are configured. It also specifies where the directory-listing csv file will be written to. The ITEM_ID_VALID filters for Items (= Directories) that are two letters followed by a dash and 10 decimals, e.g. CD-1234567890.

```
[__INBOX__]
INBOX_NAME = Inbox Test 2
COOLOFF_TIME = 1
PAUSE_TIME = 5
ITEMS_AT_ONCE = 1
DIRLIST_FILE = metadata/[@ITEM_ID@]-dirlist
ITEM_ID_VALID[] = /^\\w{2}-\\d{10}$\\i
```

The **Default section** defines some standards and the update policies for files and folders. All directory/filenames will be cleaned of illegal characters. The details for handling hashes are also set here, including the filename for the hash-files (MD5SUMS) and to store one MD5SUMS file per directory (HASH_OUTPUT = folder). The update policies state that folders are not expected to exist (if they do then something went wrong) and files are updated (overwritten) if found.

```
[__DEFAULT__]
CLEAN_SOURCE[] = illegal
HASH_TYPE = md5
HASH_OUTPUT = folder
HASH_FILEFORMAT = gnu
HASH_FILENAME = MD5SUMS
UPDATE_FOLDERS = create
UPDATE_FILES = create_or_update
```

Next, the **[.] section** sets the base archive path and specifies that folders named 'hires', 'lowres' and 'metadata' must exist.

```
[.]
TARGET_FOLDER = /mnt/archive/[@ITEM_ID@]
MUST_EXIST[] = hires
MUST_EXIST[] = lowres
MUST_EXIST[] = metadata
```

Now the **hires section** specifies the configuration for the 'hires' sub-folder of the item. The TARGET_FOLDER given here is relative to the base archive path configured in the [.] section, so the files will end up in /mnt/archive/[@ITEM_ID@]/hires/. A .wav file must exist as well and HASH_SEARCH looks, also in a relative path, for a hash-file that fits the mask .md5 - As 'hires' is next to the 'metadata' directory, we use '../' to step "back" one directory, 'metadata/' to go in that directory and look for the '.md5' file there.

```
[hires]
TARGET_FOLDER = hires
MUST_EXIST[] = *.wav
HASH_SEARCH[] = ../metadata/*.md5
```

A special case is handled here in the **lowres section**: The mp3 files have a different target, so the TARGET_FOLDER is not set relatively but with an absolute path. Besides that, the configuration will also try to find existing hashes for the mp3 files in the same manner as the [hires] section does. Also, there must be an mp3 file existing.

```
[lowres]
TARGET_FOLDER = /mnt/cache/[@ITEM_ID@]
MUST_EXIST[] = *.mp3
HASH_SEARCH[] = ../metadata/*.md5
```

The **metadata section** makes sure the xml file(s) are copied in relation to the base archive path. At least one XML file must be present in the metadata sub-directory of the item. The old .md5 files are not needed and will not be copied to the new storage (and thus, be deleted after the migration).

```
[metadata]
TARGET_FOLDER = metadata
MUST_EXIST[] = [ @ITEM_ID@ ].xml
COPY_EXCLUDE[] = *.md5
```

6.2.4. Process details

After an Item was processed by CIN with the Example 2 configuration, a log file will be available in the /logs sub-directory of the Inbox. Using the below example we'll explain what steps were happening in more detail. The full log file for reference can be found in the download section at the end of this example. We will now dissect the Common-Inbox output and explain what each step did (even more details can be found in the log file):

```
Initializing Inbox...
Loading config from file '/home/cin/cinbox/bin/cinbox_example_2.ini'...
Applying settings for inbox...
Inbox settings successfully applied.
State-keeping base folder is '/mnt/inbox'
Temporary folder: '/tmp/cinbox-inbox_test_2'
Listing item folders in '/mnt/inbox/todo'...
Inbox 'Inbox Test 2' contains 1 items.
Next Item: 1/1 (max 1)
Next item to process: 'CD-1234567890'.
Initializing item 'CD-1234567890'...
=====
Item: CD-1234567890
2017-02-28T16:20:32
=====
Checking if item 'CD-1234567890' is ready for processing...
```

```
Item age is: 17196 minutes (cooloff time: 1)
(CD-1234567890): Changed status from 'todo' to 'IN_PROGRESS'...
Item ID 'CD-1234567890' is valid. Good!
Processing item 'CD-1234567890'...
```

The Inbox checks if any Items are valid with the set regular expression. Since an Item is found, the configured time settings are checked to see if it is fit for processing. This is positive, the workflow begins.

```
=====
Executing task #1: 'FilesWait'...
=====
(CD-1234567890): Task 'FilesWait' ran successfully.
```

FilesWait checks if there is any configuration for it. As this is not the case, it has nothing to do.

```
=====
Executing task #2: 'DirListCSV'...
=====
Creating directory listing for '/mnt/inbox/in_progress/CD-1234567890'...
File already exists. Will not overwrite '/mnt/inbox/in_progress/CD-1234567890/metadata/CD-
1234567890-dirlist.csv'.
Task 'DirListCSV' is marked as 'once per Item'. Applied only to '.'.
(CD-1234567890): Task 'DirListCSV' ran successfully.
```

A snapshot of the incoming file and directory structure is created and written to the configured dirlist.csv file. In this case, the file already existed, but this can happen if an Item needed some correction after an error, so everything is in order.

```
=====
Executing task #3: 'CleanFilenames'...
=====
(CD-1234567890): Task 'CleanFilenames' ran successfully.
```

CleanFileNames checks the configuration and tests against the existing file and directory names. Even tho it is configured to handle characters that are in the 'illegal' table but there aren't any in the Items files nor directories, it has nothing to do and continues the workflow.

```
=====
Executing task #4: 'PreProcs'...
=====
(CD-1234567890): Task 'PreProcs' ran successfully.
```

```
=====
Executing task #5: 'FilesValid'...
=====
(CD-1234567890): Task 'FilesValid' ran successfully.
```

There are no pre-processor scripts or FILES_VALID options configured, so both Tasks have nothing to do.

```

=====
Executing task #6: 'FilesMustExist'...
=====
Folder '.' contains 1 files/subfolders that match 'hires'. Good.
Folder '.' contains 1 files/subfolders that match 'metadata'. Good.
Folder 'hires' contains 1 files/subfolders that match '*.wav'. Good.
Folder 'lowres' contains 1 files/subfolders that match '*.mp3'. Good.
Folder 'metadata' contains 1 files/subfolders that match 'CD-1234567890.xml'. Good.
(CD-1234567890): Task 'FilesMustExist' ran successfully.

```

Several must-exist options are set: The Item directory must contain a 'hires', 'lowres' and 'metadata' folder. In those folders, some files are required as well: A wav and mp3 (with any name due to the wildcard) as well as an XML where the filename is the ITEM ID. All requirements are met by the example dataset, the workflow can continue.

```

=====
Executing task #7: 'HashGenerate'...
=====
Generating hashcodes for files in '/mnt/inbox/in_progress/CD-1234567890'...
Generating hashcodes for files in '/mnt/inbox/in_progress/CD-1234567890/hires'...
Writing hash file '/tmp/cinbox-inbox_test_2/cd-1234567890/hires/CD-1234567890.wav.md5'...
Hashcode for 1 file(s) generated.
Generating hashcodes for files in '/mnt/inbox/in_progress/CD-1234567890/lowres'...
Writing hash file '/tmp/cinbox-inbox_test_2/cd-1234567890/lowres/CD-1234567890.mp3.md5'...
Hashcode for 1 file(s) generated.
Generating hashcodes for files in '/mnt/inbox/in_progress/CD-1234567890/metadata'...
Writing hash file '/tmp/cinbox-inbox_test_2/cd-1234567890/metadata/CD-1234567890-
dirlist.csv.md5'...
Writing hash file '/tmp/cinbox-inbox_test_2/cd-1234567890/metadata/CD-1234567890.xml.md5'...
Writing hash file '/tmp/cinbox-inbox_test_2/cd-1234567890/metadata/checksums.md5.md5'...
Hashcode for 3 file(s) generated.
(CD-1234567890): Task 'HashGenerate' ran successfully.

```

All files found will now be hashed and the hashes will be stored for later comparison. This step happens even if there are hashes existing already, thus integrity is already ensured on the input side.

```

=====
Executing task #8: 'HashSearch'...
=====
Searching existing hashcodes for files in '/mnt/inbox/in_progress/CD-1234567890'...
No must exists set. Fine.
Searching existing hashcodes for files in '/mnt/inbox/in_progress/CD-1234567890/hires'...
1 files containing hash code match for 'hires/CD-1234567890.wav' found (md5 =
72e96e9b88351cd233fa25d23f46e68d)
No must exists set. Fine.
Searching existing hashcodes for files in '/mnt/inbox/in_progress/CD-1234567890/lowres'...
1 files containing hash code match for 'lowres/CD-1234567890.mp3' found (md5 =
0c200572407d3d8061eb2dad0a5ddb13)
No must exists set. Fine.
Searching existing hashcodes for files in '/mnt/inbox/in_progress/CD-1234567890/metadata'...

```

```
No must exists set. Fine.  
(CD-1234567890): Task 'HashSearch' ran successfully.
```

Now the Task HashSearch checks the configuration and has set to check the /metadata sub-directory of the Item for any files matching '*.md5'. If a file is found, hashes generated earlier will be compared with the found hashes. As in this scenario we do not expect hashes to come with every Item, they are optional and thus, not set as must-exist. Hashes are found and match our media, everything is go for the next step.

```
=====
Executing task #9: 'CopyToTarget'...
=====
Creating folder '/mnt/archive/temp_CD-1234567890'...
No files to copy in '/mnt/inbox/in_progress/CD-1234567890'. Fine.
Creating folder '/mnt/archive/temp_CD-1234567890/hires'...
Copying '/mnt/inbox/in_progress/CD-1234567890/hires/CD-1234567890.wav' to '/mnt/archive/temp_CD-1234567890/hires/CD-1234567890.wav'...
CD-1234567890.wav
      32,768   7%    0.00kB/s    0:00:00
     444,524 100%   78.54MB/s    0:00:00 (xfr#1, to-chk=0/1)
1/1 file(s) copied from '/mnt/inbox/in_progress/CD-1234567890/hires' to '/mnt/archive/CD-1234567890/hires'.
Creating folder '/mnt/cache/temp_CD-1234567890'...
Copying '/mnt/inbox/in_progress/CD-1234567890/lowres/CD-1234567890.mp3' to '/mnt/cache/temp_CD-1234567890/CD-1234567890.mp3'...
CD-1234567890.mp3
      32,768  40%    0.00kB/s    0:00:00
     80,666 100%   45.68MB/s    0:00:00 (xfr#1, to-chk=0/1)
1/1 file(s) copied from '/mnt/inbox/in_progress/CD-1234567890/lowres' to '/mnt/cache/CD-1234567890'.
Creating folder '/mnt/archive/temp_CD-1234567890/metadata'...
Copying '/mnt/inbox/in_progress/CD-1234567890/metadata/CD-1234567890-dirlist.csv' to '/mnt/archive/temp_CD-1234567890/metadata/CD-1234567890-dirlist.csv'...
CD-1234567890-dirlist.csv
      1,128 100%    0.00kB/s    0:00:00
      1,128 100%    0.00kB/s    0:00:00 (xfr#1, to-chk=0/1)
Copying '/mnt/inbox/in_progress/CD-1234567890/metadata/CD-1234567890.xml' to '/mnt/archive/temp_CD-1234567890/metadata/CD-1234567890.xml'...
CD-1234567890.xml
      125 100%    0.00kB/s    0:00:00
      125 100%    0.00kB/s    0:00:00 (xfr#1, to-chk=0/1)
Excluding file '/mnt/inbox/in_progress/CD-1234567890/metadata/checksums.md5'. Matches pattern '*.md5'.
2/3 file(s) copied from '/mnt/inbox/in_progress/CD-1234567890/metadata' to '/mnt/archive/CD-1234567890/metadata'.
(CD-1234567890): Task 'CopyToTarget' ran successfully.
```

At this point the copy operation begins. All data is structured according to the configuration and copied to a temp folder on the target paths. An exception is the checksums.md5 file, which is excluded by using COPY_EXCLUDE. All file copies are successful and the Workflow moves on.


```

=====
Executing task #10: 'HashValidate'...
=====
Checking target folder '/mnt/archive/temp_CD-1234567890'...
Checking target folder '/mnt/archive/temp_CD-1234567890/hires'...
Target '/mnt/archive/temp_CD-1234567890/hires/CD-1234567890.wav' is valid.
Checking target folder '/mnt/cache/temp_CD-1234567890'...
Target '/mnt/cache/temp_CD-1234567890/CD-1234567890.mp3' is valid.
Checking target folder '/mnt/archive/temp_CD-1234567890/metadata'...
Target '/mnt/archive/temp_CD-1234567890/metadata/CD-1234567890-dirlist.csv' is valid.
Target '/mnt/archive/temp_CD-1234567890/metadata/CD-1234567890.xml' is valid.
Excluding file '/mnt/inbox/in_progress/CD-1234567890/metadata/checksums.md5'. Matches pattern
'*.md5'.
(CD-1234567890): Task 'HashValidate' ran successfully.

```

After the copy operation, the files are validated on the target paths. All hashes are valid, with the checksums.md5 file being ignored again. The data was copied without any errors.

```

=====
Executing task #11: 'RenameTarget'...
=====
Creating folder '/mnt/archive/CD-1234567890'...
Merging files from '/mnt/archive/temp_CD-1234567890' into '/mnt/archive/CD-1234567890'...
Creating folder '/mnt/archive/CD-1234567890/hires'...
Merging files from '/mnt/archive/temp_CD-1234567890/hires' into '/mnt/archive/CD-
1234567890/hires'...
Moving '/mnt/archive/temp_CD-1234567890/hires/CD-1234567890.wav' to '/mnt/archive/CD-
1234567890/hires/CD-1234567890.wav'...
Creating folder '/mnt/cache/CD-1234567890'...
Merging files from '/mnt/cache/temp_CD-1234567890' into '/mnt/cache/CD-1234567890'...
Moving '/mnt/cache/temp_CD-1234567890/CD-1234567890.mp3' to '/mnt/cache/CD-1234567890/CD-
1234567890.mp3'...
Creating folder '/mnt/archive/CD-1234567890/metadata'...
Merging files from '/mnt/archive/temp_CD-1234567890/metadata' into '/mnt/archive/CD-
1234567890/metadata'...
Moving '/mnt/archive/temp_CD-1234567890/metadata/CD-1234567890-dirlist.csv' to '/mnt/archive/CD-
1234567890/metadata/CD-1234567890-dirlist.csv'...
Moving '/mnt/archive/temp_CD-1234567890/metadata/CD-1234567890.xml' to '/mnt/archive/CD-
1234567890/metadata/CD-1234567890.xml'...
Task 'RenameTarget' is recursive itself. Applied only to '.'.
(CD-1234567890): Task 'RenameTarget' ran successfully.

```

A move (=rename) is done on the target paths so the Item directory gets its final name. Everything is ok, so the next Task can be started.

```

=====
Executing task #12: 'HashOutput'...
=====
Folder '/mnt/archive/CD-1234567890'...
Folder '/mnt/archive/CD-1234567890/hires'...
Folder '/mnt/cache/CD-1234567890'...
Folder '/mnt/archive/CD-1234567890/metadata'...
Excluding file '/mnt/inbox/in_progress/CD-1234567890/metadata/checksums.md5'. Matches pattern
'*.md5'.
(CD-1234567890): Task 'HashOutput' ran successfully.

```

The HashOutput Task creates a MD5SUMS file for each folder and writes the related hashes to them. The checksums.md5 file is yet again ignored.

```

=====
Executing task #13: 'PostProcs'...
=====
(CD-1234567890): Task 'PostProcs' ran successfully.

```

No post-processing scripts are configured, so PostProcs has nothing to do.

```

Item 'CD-1234567890': 13/13 task(s) processed successfully!
(CD-1234567890): Changed status from 'in_progress' to 'DONE'...
Reached limit of 1 items per run.
Removing finished Items older than 0 days...
Deleted 0 items.
Clean ending of Inbox...

```

All Tasks for the Item were successful, the process is complete.

7. Troubleshooting

7.1. Support

Below, find a collection of the most common issues and how to solve them. For further help, feature/plugin/task requests or bug reports, please initially contact support@av-rd.com for login credentials for the [AV-RD support page](#). Once you have access to the support portal, all bug reports and help requests will be handled there.

7.2. The cinbox.php script can't be executed

- Check if the script is execute-enabled. If not, use `$ sudo chmod +x /home/cin/cinbox/bin/cinbox.php` and retry.
- Does the Common-Inbox directory have the correct owner/group, e.g. cin:cin? Use `chown` to set the correct user and group, e.g. `$ sudo chown cin:cin /home/cin/cinbox -Rv`

7.3. Changing language does not work

Trying to change the locale when running 'cinbox.php --lang de_AT -i /mnt/inbox' results in:

```
Setting language to 'de_AT'.  
WARNING Language 'de_AT' not supported. Is matching locale installed?
```

The locale needs to be installed/generated on the system. Please check how to add locales to the operating system you are using. An example for Ubuntu Server 16.04 systems:

```
cin@sandbox:~/cinbox/bin$ sudo locale-gen de_AT  
Generating locales (this might take a while)...  
de_AT.ISO-8859-1... done  
Generation complete.  
cin@sandbox:~/cinbox/bin$ sudo update-locale
```

This will enable the required locale and the output when running CIN again with the above command should look like:

```
Setting language to 'de_AT'.  
Initialisiere Inbox...
```

7.4. Executing cinbox.php results in a black screen

The php-cli package might be missing. This can be verified by executing `$ php -r "echo 'test';"`; If the cli extension is missing, this command might not work. Install it using `sudo apt-get install php-cli` and retry. Please note: php5 and php7 behave different when it comes to cli extensions.

7.5. "bad interpreter" error

If you execute `cin@sandbox:~/cinbox$./cinbox.php` and get the error `"-bash: ./cinbox.php: /usr/bin/php: bad interpreter: No such file or directory"`, php is most likely not installed on the system. To fix this, install PHP and the PHP CLI package: `sudo apt-get install php php-cli`

7.6. ERROR Folder does not exist

If the inbox is not properly prepared, some folders that need to be initially created might be missing. If this is the case, you will see the following errors:

```
Initializing Inbox...
Loading config from file '/mnt/inbox/cinbox.ini'...
Applying settings for inbox...
Inbox settings successfully applied.
State-keeping base folder is '/mnt/inbox'
ERROR Folder for status 'TODO' does not exist: /mnt/inbox/todo
ERROR Folder for status 'IN_PROGRESS' does not exist: /mnt/inbox/in_progress
ERROR Folder for status 'DONE' does not exist: /mnt/inbox/done
ERROR Folder for status 'ERROR' does not exist: /mnt/inbox/error
ERROR Folder for status 'LOG' does not exist: /mnt/inbox/log
ERROR Could not init Inbox due to errors.
Clean ending of Inbox...
```

To resolve this issue, create the missing directories in the inbox path, e.g.: `mkdir -p /mnt/inbox/{todo,in_progress,done,error,log}`

7.7. Inbox contains 0 Items

Starting cinbox returns no valid Items to work with:

```
cin@sandbox:~/cinbox/bin$ ./cinbox.php -i /mnt/inbox
Initializing Inbox...
Loading config from file '/mnt/inbox/cinbox.ini'...
Applying settings for inbox...
Inbox settings successfully applied.
State-keeping base folder is '/mnt/inbox'
Temporary folder: '/tmp/cinbox-inbox_test_1'
Listing item folders in '/mnt/inbox/todo'...
Inbox 'Inbox Test 1' contains 0 items.
```

Possible Reasons:

- The Items to import have to be put in the /todo sub-directory of the inbox.
- If it is verified that the items are in fact in the inbox' todo directory (in this example /mnt/inbox/todo/), check if the files are accessible by the user running the CIN.
- Another reason might be that the regular expression(s) for ITEM_ID_VALID[] in the [__INBOX__] section of the inbox.ini file is not set or not working. Verify that the regular expression is filtering for Item IDs correctly. Very helpful tools for this are: [\[2\]](#) or [\[3\]](#)

7.8. CleanFileNames: Empty or invalid character map

The Task CleanFileNames exists with an error like in the following log:

```
=====
Executing task #3: 'CleanFileNames'...
=====
PHP Notice:  Undefined index: umlaut in /home/cin/cinbox/bin/tasks/TaskCleanFileNames.php on line
241
PHP Warning:  array_merge(): Argument #2 is not an array in
/home/cin/cinbox/bin/tasks/TaskCleanFileNames.php on line 241
ERROR  Problems with 'TEST-123456'
ERROR  Empty or invalid character map.
TEST-123456: Changed status from 'in_progress' to 'ERROR'...
```

Check the configuration sections for 'CLEAN_SOURCE[]' parameters and make sure all settings used are valid. A typo might be the reason for this error, e.g. 'umlaut' will fail as the correct setting would be 'umlauts'. For details on available options check the parameter documentation.

7.9. ERROR Invalid configuration 'CONFIG_OPTION': Must be an array.

This error is often caused by missing the [] brackets after a configuration option in the cinbox.ini file. To fix this, make sure all options that are arrays (= allow for multiple lines of configuration) have the brackets at the end, e.g. 'HASH_SEARCH = *.md5' ⇒ 'HASH_SEARCH[] = *.md5'.

7.10. Inbox is reporting the same error after the issue has been resolved

To speed up the process, Common-Inbox is trying to avoid unnecessary steps when resuming a workflow after it was in error state. In most cases, resolving the error and then moving the Item back into the todo sub-directory will fix the issue. However, in some cases - especially when there were checksum mismatches of some of the Item's files - this can lead to a state where the Inbox will still be using the already-computed hashes instead of re-computing them for the files that were replaced/fixed by an operator. This is due to some cached data in the Items temporary directory. In order to completely reset an Item, this temp directory has to be deleted before moving the Item back into the 'todo' folder. The location and name of those cached files is computed as follows:

```
Path to System-Temp + / + Cleaned Inbox Name + / + ItemID
```

For the Item "ABC-1234" in the Inbox "Test Inbox 1" on a Debian sytem, the path will look like this:

```
/tmp/Test_Inbox_1/ABC-1234
```

Consequently, the way to completely reset a workflow and have everything including the cache files refreshed, consists of first deleting the Item's temp directory and then moving it from the 'error' back to the 'todo' directory of the Inbox.

7.11. Undefined variable: age

The inbox is reporting a PHP Notice and the item stays at Item age zero all the time and does not get processed:

```
Checking if item 'a-12345' is ready for processing...  
PHP Notice:  Undefined variable: age in /opt/cinbox/bin/include/CIItem.php on line 850  
Item age is: 0 minutes (cooloff time: 10)  
Item not ready for processing: Need to wait 10 more minutes.
```

This happens if the Item directory a-12345 in the /todo of the Inbox is empty and has no files in it or if access is denied. This causes the age variable to stay empty as there is no age to determine, triggering this PHP Notice.

Please check the Item directory in the /todo and make sure that there is actual data present. If data is present, check for access rights being set correctly.