

1) TrueFile Specs

TrueFile Specs are used by a growing number of Markzware products. A TrueFile Spec is a simple, text-based file format which allows anybody to store job-related information.

The format is both human- and machine-readable.

The TrueFile Spec file format is very similar to the .INI file format: a TrueFile Spec consists of a number of sections. Each of the sections contains a number of entries.

A simple protocol is used in order to avoid clashes between various suppliers and consumers of data: each section name in the TrueFile Spec starts with the Internet domain name of the company or person who defines the structure of that particular section.

As an example: below part of a TrueFile Spec as used by MarkzNet:

```
[markzware.com TrueFile-Remote]
Host=216.167.57.153
JobNumber=2226
FileName=2226
LocalJobNumber=1
MarkzNetVersion=1.0r17
DirPath=/pub/markznet/
Images=Images_Folder
Fonts=Fonts_Folder
MinSaveSize=100000
CanSave=1
CanSend=1
FeedBackURL=http://216.167.57.153/cgi-bin/feedback.pl
```

```
[testcompany.com TrueFile-Our Own Job Ticket]
EMail=john.doe@somewhere.else
CustomerCode=13323
PaperSize=US Letter
PaperQuality=Newsprint
PaperColor=White
JobColorType=Black and White
SpotColorCount=0
Urgency=48 Hours
Comment=
```

Here you see two sections - one with a structure defined by Markzware, and one with a structure defined by a fictive 'Testcompany'.

Only the party whose domain name is contained in the section can define what entries are possible in that section, what data can be stored there, and how that data is to be encoded. The encoding of the data can be kept private or can be publicized - that is totally up to whoever defines the section.

In the example above, Markzware has defined the layout of the 'Remote' section; Testcompany has defined the layout of the 'Our Own Job Ticket' section.

If the entry names and data-encoding mechanism is publicized, then outside parties can read or write data in the section; this allows the TrueFile Spec to be used as a means of communication.

For example, the MarkzNet Client relies on a few specific sections in the TrueFile Spec. The layout and encoding mechanism of these sections is defined and published by Markzware; but the actual data stored in those sections is defined by whoever runs the MarkzNet Server.

That means: Markzware defines the entries and their meaning for MarkzNet. A non-Markzware party (e.g. a web designer) then fills in the data in these entries to communicate with and control the remote MarkzNet Client.

2) Job Flattening/Unflattening

2.1) What is it ?

MarkzScout 2.0 can disassemble a job folder with nested subfolders, and convert it to a number of 'loose' files.

These files can then all be processed separately by MarkzScout.

At a certain point in time, MarkzScout can be instructed to re-assemble the original job folder structure from the loose files.

2.2) What can you use this feature for ?

The easiest way to explain is by using an example. Suppose you're using a workflow where job folders are coming in. Each of these job folders contains at least one XPress document, and a number of linked images in TIFF format.

All the files have been preflighted to make sure the images are not scaled up or down inside the XPress documents - they are all placed at 100%. The TIFF files all have a resolution of 300dpi or higher.

We want to resample all the TIFF files for all these jobs, and bring them down to 150 dpi.

By flattening the jobs, we can process all TIFF files transparently through a simple Photoshop action layout, without having to worry about what job these TIFF files are part of, or where in the job folder structure they belong.

After processing the TIFF files, we can reconstitute the exact same job folder structures.

Two example job folders could contain the following files:

Jobfolder1:MainDoc.qxd
Jobfolder1:image1.tif
Jobfolder1:moreImages:Image2.tif
Jobfolder1:moreImages:Image3.tif
Job2:OtherMainDoc.qxd
Job2:image2.tif
Job2:somemages:Image2.tif
Job2:someImages:Image9.tif

The Job Flattening function of MarkzScout would convert these two folder structures into the following separate files (example):

00A5Z001.qxd
00A5Z002.tif
00A5Z003.tif
00A5Z004.tif
00A60001.qxd
00A60002.tif
00A60003.tif
00A60004.tif

After flattening, all files are 'loose' - they are separate from each other. That means, for example, that it is not useful to try and open the XPress documents in their flattened state, and update the links - the links are disrupted at this point in time.

On the other hand, we can separate off and process all the TIFF files. Potential problems like name clashes etc... have been resolved by the automatic renaming function which is part of the job flattening function.

After the TIFF files have been processed, they can be joined up again with the other files. Then the 'unflatten' function of MarkzScout can be used to reconstruct the original folders structures with the (now downsampled) TIFF files.

The original job folder name is not restored to avoid potential name clashes - only the job's folder structure is restored. The name is still accessible in the job's TrueFile Spec, and a simple MarkzONE script can restore the original job folder name if so desired.

00A5Z:00A5Z.tfs
00A5Z:MainDoc.qxd
00A5Z:image1.tif
00A5Z:moreImages:Image2.tif
00A5Z:moreImages:Image3.tif
00A60:00A60.tfs
00A60:OtherMainDoc.qxd
00A60:image2.tif
00A60:somemages:Image2.tif
00A60:someImages:Image9.tif

2.2) How is the flattening function implemented ?

2.2.1) Naming of flattened files

The flattened file names are all in 8.3 format and are generated by concatenating a 5-letter job code (00000 - ZZZZZ) and a 3-letter file number (000 - ZZZ), both case-insensitive.

MarkzScout generates these job codes and file numbers as needed. MarkzScout guarantees unique file names by using this 'numbering' scheme - this to avoid potential name clashes by files with the same name. A consequence of the current numbering scheme used is that there can be at most 46656 files in a single job (each position in the file number has 26 + 10 possible values; $36 * 36 * 36 = 46656$), and there can be at most 60466176 concurrently flattened jobs.

2.2.2) TrueFile Specs

MarkzScout remembers the original job structure by using a private section in the TrueFile Spec for the flattened job. It records the original file name and relative file position into the TrueFile Spec.

If the original job folder already did contain an existing TrueFile Spec, then that same TrueFile Spec is used, so any information which was already present in other sections in that TrueFile Spec will be preserved.

If no TrueFile Spec was present in the job folder while it was being flattened, then a new TrueFile Spec is created.

When the job is unflattened, the info stored in the TrueFile Spec is used to resurrect the original file names.

2.2.3) Where are the TrueFile Specs stored

The TrueFile Specs for all flattened jobs are kept in a central location (a subfolder of the Preferences folder on Mac, a subfolder of the MarkzScout folder on Windows).

When the job is unflattened again, the TrueFile Spec is moved into the job folder, so it can accompany the job folder throughout the workflow.

That means that the location of the TrueFile Spec for a particular job depends on whether that job is flattened or not. If the job is flattened, then the TrueFile Spec resides in a central location. If a job is not flattened, then the TrueFile Spec resides in the actual job folder.

3) Additional support for TrueFile Specs

MarkzScout can make use of a TrueFile Spec which either resides at the top level inside a job folder, or in the central location in the case of flattened jobs.

A TrueFile Spec can be used for multiple purposes:

- MarkzScout uses it to record the original file names while flattening a job
- MarkzNet uses it to store job ticket info, preflighting rules, and upload information. MarkzScout can then retrieve this job ticket info from the TrueFile Spec as needed.
- Other applications can store additional information in private sections of the TrueFile Spec.

...

In MarkzScout 2.0, the TrueFile Spec is also used to store user-defined attributes on a per-job and per-file basis - in addition to the attributes defined by the MarkzONE data model, there can be an unlimited number of user-defined attributes. Examples could be: 'customer_name', or 'date_accepted',...

These user-defined attributes are treated like the standard MarkzONE attributes for all intents and purposes - it becomes very easy to write MarkzScout checkpoints with rules that use the values of these attributes, for example to route jobs based on specific customer-related criteria.

4) FlightCheck Checkpoint

MarkzScout 2.0 has a special 'FlightCheck Checkpoint'. This checkpoint allows one to test the outcome of an earlier FlightCheck Actionpoint: if the earlier FlightCheck Actionpoint did not find any errors, then the FlightCheck Checkpoint will allow the job to pass (green, 'N' exit). If the earlier FlightCheck Actionpoint did bring up one or more errors, the the FlightCheck Checkpoint will fail the job (red, 'Y' exit).

By combining regular checkpoints, one or more FlightCheck Actionpoints, and one or more FlightCheck Checkpoints, one can easily create a layout where different FlightCheck ground controls are automatically selected depending on a particular job classification, and jobs are then tested for success or failure with FlightCheck.

5) Improved FlightCheck support

MarkzScout 2.0 adds a new function to the FlightCheck Actionpoint: it allows the automation of the Collect function in FlightCheck.