QuickSave tool – Help

credits

The tool was made by Bernhard Schaffer, SuperSTEM Laboratory Daresbury/UK.
email: Bschaffer@superstem.org
version: 2012-07-02
documentation update: 2012-07-02

purpose

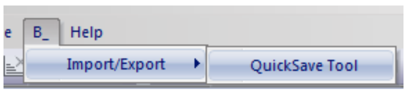
This script is meant to make saving of a lot of images form the microscope easier and quicker by allowing 'one-button-click' saving of new images to a pre-defined folder according to several pre-defined naming arrays. The script has been made flexible, so that different users can define different saving templates. This information is stored in the global tag structure of DigitalMicrograph. The first setup – which is described in detail below – might be a bit complicated, but once the tool is configured, it should be a good, flexible "oneClick-save" solution.
This package works on both GMS 2.x and prior GMS versions.

Contents

Setup

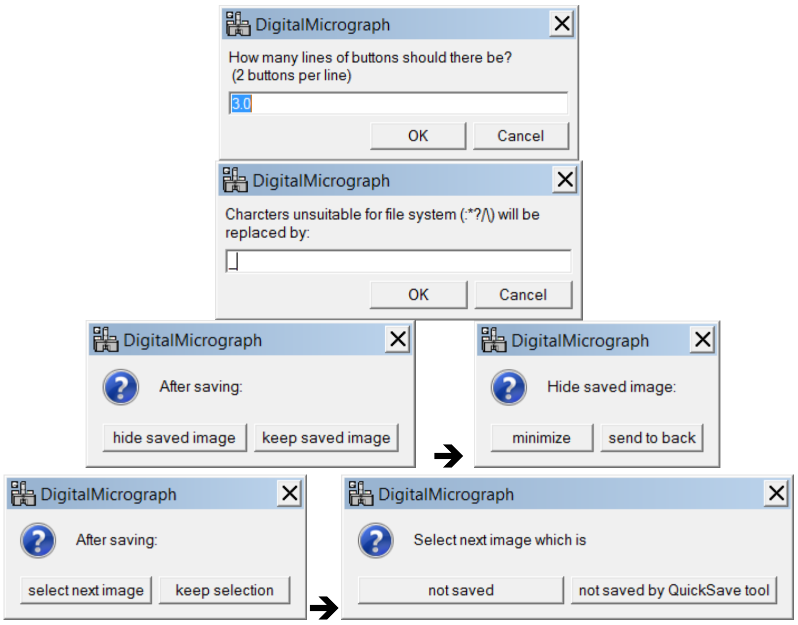
install

Copy the *b_QuickSaveTool.gtk* file into the plugins folder and restart DigitalMicrograph. There should now be a new menu entry:
B_ → Import/Export → QuickSave Tool



first time use

Launching the script for the first time – or when the global tags have been cleared – the script will prompt to ask for four basic parameters:


DigitalMicrograph
How many lines of buttons should there be?
(2 buttons per line)
3.0

DigitalMicrograph
Charcters unsuitable for file system (:*?/\) will be replaced by:
_

DigitalMicrograph
After saving:

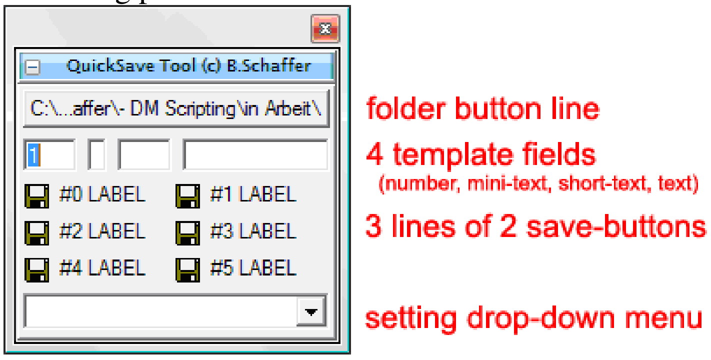
DigitalMicrograph
Hide saved image:

DigitalMicrograph
After saving:

DigitalMicrograph
Select next image which is

- The first defines the basic layout of the tool, i.e. the number of "save buttons" it should show. Each line will have 2 buttons, and it is up to you to choose how many buttons (=different saving templates) should be displayed in the tool.
- The second defines the default separator character into which all "forbidden" file-characters will be automatically changed. By default this is the underscore – sign (_).
- The third dialog asks what the tool should do with the front-image window once it is saved. (If it is closed, this questions is of course redundant.) Either this image stays as it is ("keep") or it is either minimized or send "behind" all other open image windows. In both later cases, the next image automatically becomes the selected front image unless the fourth dialog changes this.
- The fourth dialog ask what image should become the "selected front image" after saving. Either the natural selection is kept, or the tool search for the next "unsaved" image, makes it the front image and selects it. If so, either the next "unsaved" image can be chosen, or the next image which has no "saved by QuickSave tool" tag (but might have been saved before.)

The tool will now open as a floating palette:

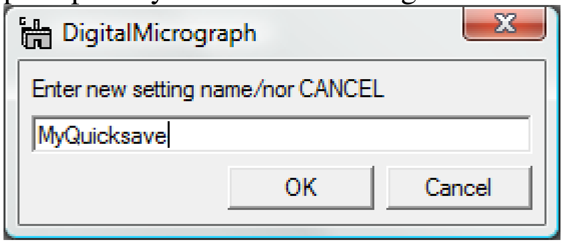

QuickSave Tool (c) B.Schaffer
C:\...affer\- DM Scripting\in Arbeit\
1
#0 LABEL
#1 LABEL
#2 LABEL
#3 LABEL
#4 LABEL
#5 LABEL
folder button line
4 template fields
(number, mini-text, short-text, text)
3 lines of 2 save-buttons
setting drop-down menu

As there is not yet a setting stored, the buttons will have default labels and no effect. Ones first has to create new template settings as explained in detail in the next section.
For a quick start, select "ADD NEW SETTINGS" from the drop-down and enter "default" as a settings name. This will create a standard default setting, which might satisfy some of your needs already…

creating new settings

If needed, launch the script with the ALT key pressed to get a floating palette which shows the setting drop-down. (It will be automatically showed at first time start.)

To create a new setting, use the setting drop-down menu and choose "…add new setting…". You will be prompted by a number of dialog boxes to create a new setting.


DigitalMicrograph

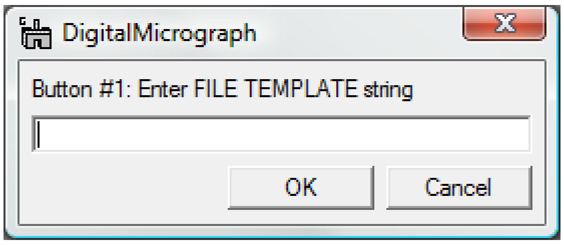
Enter new setting name/nor CANCEL

MyQuicksave

This will be the name of the setting under which it can be loaded from the setting drop-down later on.

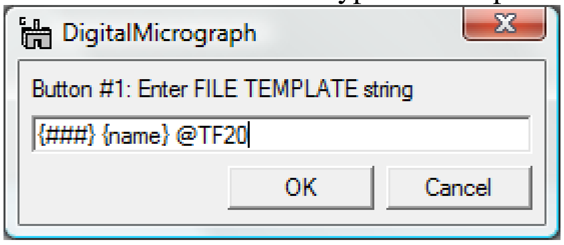
➔ *if you enter "default", a hard-coded default setting will be created.*

For *each* button of the tool there will now be two entries to be made:


DigitalMicrograph

Button #1: Enter FILE TEMPLATE string

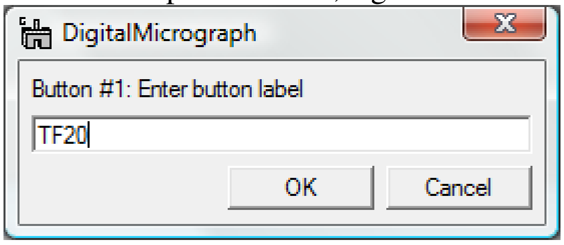
First the *file template* string is defined. This is the name under which an image will be saved when the according button is pressed. The string can contain all standard characters plus a range of *fields* which will be replaced by according information on saving. The fields are explained in the results window or further down in this documentation. A typical example template would f.e. be:


DigitalMicrograph

Button #1: Enter FILE TEMPLATE string

{###} {name} @TF20

Such a template would save all images with a 3-digits serial number, followed by the name of the image (in its title) and a "@TF20".
The second entry is simply the *label* for this quick-save button. It should not be too long, as it has to fit in half of the tool's palette width, e.g.:


DigitalMicrograph

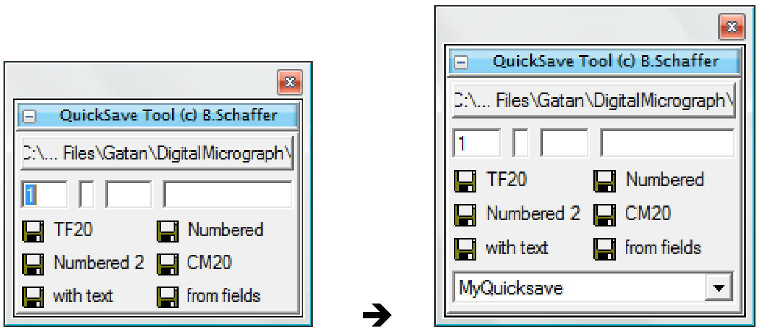
Button #1: Enter button label

TF20

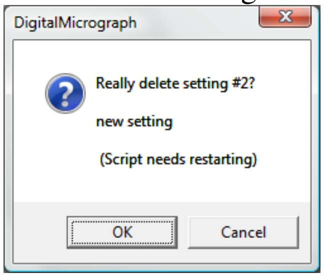
Repeat this template/label entering for all buttons of your setting.
Once finished, select the setting form the setting drop-down to load it.

selecting, changing & deleting settings

By default, the last used setting will be loaded when the tool is started, and the setting drop-down will *not* be shown. Launch the script with the ALT key pressed to get a floating palette which shows the setting drop-down:


QuickSave Tool (c) B.Schaffer
:\... Files\Gatan\DigitalMicrograph\
1
TF20
Numbered
Numbered 2
CM20
with text
from fields
➔
QuickSave Tool (c) B.Schaffer
:\... Files\Gatan\DigitalMicrograph\
1
TF20
Numbered
Numbered 2
CM20
with text
from fields
MyQuicksave

- Select the setting you like from the drop-down and the buttons will updated.
- Select "…add new setting…" to create additional settings. (see above)
- Select a setting with the ALT key pressed to delete it permanently. A dialog will be prompted first. Note, that in order to delete a setting, the tool needs to be re-launched. It thus closes automatically on deletion of a setting.

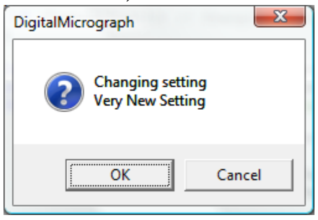


Really delete setting #2?

new setting

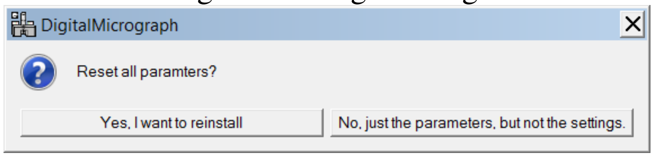
(Script needs restarting)

- Select a setting with the CONTROL key pressed to change it. A dialog will be prompted first. If not cancelled, a series of dialogs asks for changes in *file template* and *label* for all buttons.



Changing setting
Very New Setting

- Note: Deleting/Changing a setting will only work when the setting is *changed* to while the button is pressed. It will not work, if the setting is already chosen and just selected again. (Only changes in the drop-down menu are recognized by the script.)
- To change the template-string (put not the button label) of one button in a setting, one can click on the according save-icon (the disc) while holding down the SHIFT & CONTROL keys. A dialog will show the current template string and allow modifications, which are automatically stored. This is also a quick way of checking the current template of a button.
- The number of buttons is a parameter valid of *all* settings and can not be changed in individual settings. To produces a tool with a different number of buttons, re-initiate the tool completely by starting the script with the ALT and the SHIFT key both pressed and selecting "reinstall". This will *delete* all settings from the global tags.



Reset all paramters?

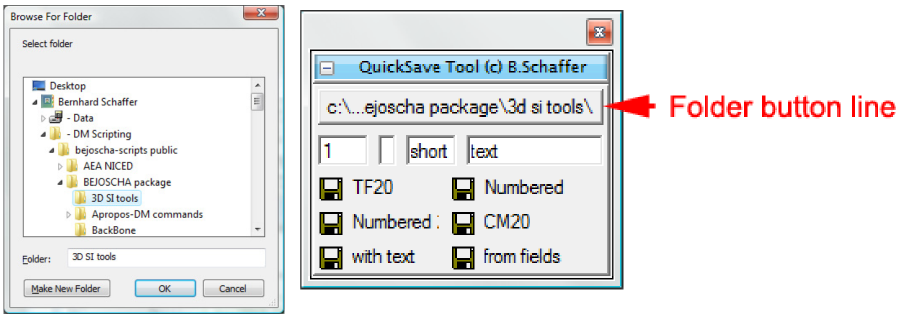
- To change some other general parameters (see help on install), start the script with the ALT and the SHIFT key both pressed, but select the "just the parameters" option.

Using the QuickSave tool

Once the tool has been initialized and settings have been created (see above), the tool can be used as follows:

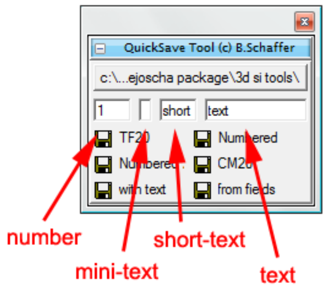
Selecting the directory

- The first button of the tool changes the folder into which the images are saved. Clicking on the button will open a directory-dialog. The button will be labelled with the (if needed abbreviated) folder path.



Using the variable fields

- The second line of the tool contains four fields for variable text. These fields may be used in the *file templates*, but they don't have to.
  - The first field is an integer-number field which will be used by the field-code {###} in the templates. Integer values can be entered, but this field is also auto-increased by one, if the save-buttons are used while the CONTROL key is pressed.
  - The second field is the *minitext* field which will be used by the field-code {minitext} in the templates. While any text can be entered in this field, it is meant to contain a single letter, i.e. "a". If the save-buttons are used with the SHIFT key pressed, a single-letter in the minitext field will advance in alphabetic order a→b→c …
  - The third field is the *short text* field which will be used by the field-code {shorttext}.
  - The fourth field is the *text* field which will be used by the field-code {text}.



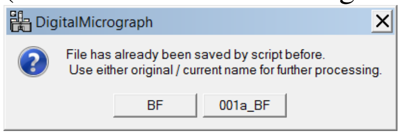
Saving – buttons

- The following lines contain the save-buttons with their respective labels as defined in the setting. Each of the save-buttons has a template which will be used to create the filename under which the image should be saved. Clicking on the disc-symbol will carry out the save-actions.

Clicking on the disc-symbol:

- The *front most displayed* image is saved in the pre-defined folder under the file name created from the *file template*.
  An example: Assume the front most images has the title "BF" and the file template of the button is:
  {###}{minitext}_{name}
  Pressing this save button will save this image as 001a_BF.dm3 in the currently selected folder. Depending on the basic settings, the image will also be brought to the "backmost" position, or minimized and the next (unsaved) image is automatically selected.
- The results window will state that the file has been saved

- If the image has already been saved with the QuickSave tool before, a dialog asks which of the names (the current or the original one) should be used for determination of the new filename.



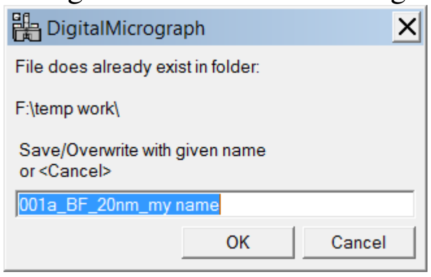
DigitalMicrograph

File has already been saved by script before.
Use either original / current name for further processing.

BF | 001a_BF

This is important, because the templates would possibly alter a filename in an unexpected way, if the same algorithm is applied more than once successively.
In the example from above, the image "BF" would first be saved as 001a_BF.dm3, but if saved with the same button again, it would be named 001a_001a_BF.dm3!

- Whenever the QuickSave tool encounters that a filename already exists, it will prompt a warning dialog with the choice of changing the filename, confirming or cancelling the saving operation.



DigitalMicrograph

File does already exist in folder:

F:\temp work\

Save/Overwrite with given name
or <Cancel>

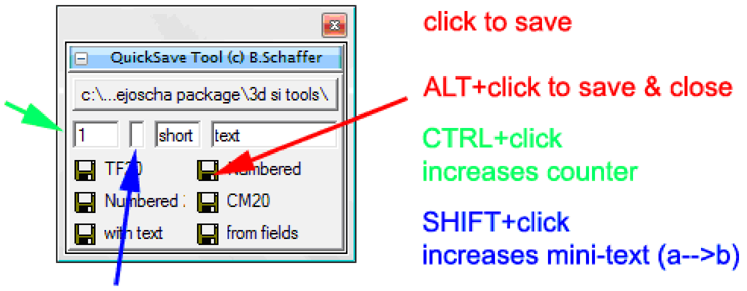
001a_BF_20nm_my name

OK | Cancel

Clicking on the disc-symbol with additional options:

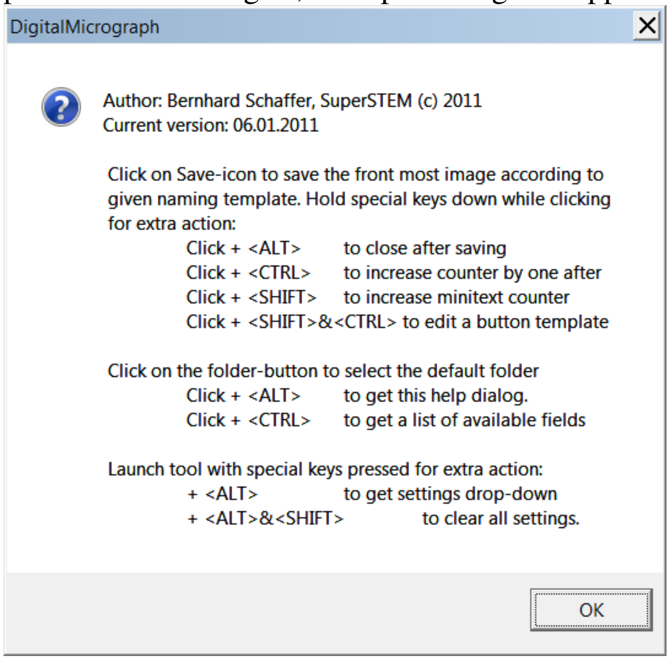
- Pressing the ALT key while clicking a save-button will save and close the front-most image.
- Pressing the CONTROL key while clicking a save-button will save and increase the integer-field by 1.

  In addition, if the mini-text field contains a single character, it will reset it to *a* or *A*.
- Pressing the SHIFT key while clicking a save-button will save and increase the mini-text field (a→b etc.)
- It is possible to hold the ALT & SHIFT
  or alternatively hold the ALT & CONTROL
  keys simultaneously while clicking the save button to increase counters and close the image after saving.
- Pressing both SHIFT & CONTROL keys simultaneously while clicking a save-button brings up a dialog to change the buttons template-string.


QuickSave Tool (c) B.Schaffer
c:\...ejoscha package\3d si tools\
1
short
text
TF20
Numbered
Numbered
CM20
with text
from fields
click to save
ALT+click to save & close
CTRL+click
increases counter
SHIFT+click
increases mini-text (a-->b)

Getting help

The tool has some simple help hardcoded to it. If you press the folder-selection button and keep the ALT key pressed while doing so, a simple dialog will appear.



DigitalMicrograph

Author: Bernhard Schaffer, SuperSTEM (c) 2011
Current version: 06.01.2011

Click on Save-icon to save the front most image according to given naming template. Hold special keys down while clicking for extra action:

```
Click + <ALT>      to close after saving
Click + <CTRL>     to increase counter by one after
Click + <SHIFT>    to increase minitext counter
Click + <SHIFT>&<CTRL> to edit a button template
```

Click on the folder-button to select the default folder

```
Click + <ALT>      to get this help dialog.
Click + <CTRL>     to get a list of available fields
```

Launch tool with special keys pressed for extra action:

```
+ <ALT>            to get settings drop-down
+ <ALT>&<SHIFT>          to clear all settings.
```

File templates & field codes

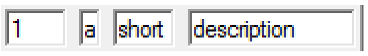
- File templates may contain any combination of characters, signs and field-codes.
- When a filename is created from a templates, the final name will be parsed for *forbidden* characters which will be replaced by a defined replacement character (see *first time use*).
- The following fields are currently available:

FIELD name	**effect**
`{name}`	replaced by the *title* of the image to be saved
`{name:`*se*`|`*re*`}`	replaced by the *title* of the image, but all occurrences of *se* are replaced by *re* first.
`{name:?}`	replaced by the *title* of the image, but a pop-up dialog will allow editing first.
`{####}`	replaced by the integer value of the integer-field. There may be any number of # characters in this field. Each # is replaced by a digit. Leading zeros are used.
`{minitext}`	replaced by the content of the *mini-text* field. Main purpose: Adding "sub-numbering" to the interfiled, i.e. 100a, 100b, 100c etc.
`{shorttext}`	replaced by the content of the *short-text* field
`{text}`	replaced by the content of the *text* field
`{date}`	replaced by the actual date in its 'short' form as returned by "GetDate(0)". Note, that possibly forbidden characters are replaced automatically.
`{DATE}`	replaced by the actual date in its 'long' form as returned by "GetDate(1)" Note, that possibly forbidden characters are replaced automatically.
`{time}`	replaced by the actual time excluding seconds. Note, that possibly forbidden characters are replaced automatically.
`{TIME}`	replaced by the actual time including seconds.

	Note, that possibly forbidden characters are replaced automatically.
{TAG:*tgPath*}	replaced by the value of the image-tag at given tag-path *tgPath*, if such a tag exist. e.g. {TAG:DataBar:Exposure Number} will be replaced by the value of the *Exposure Number* tag if such a tag exists in the image.
{GTAG:*tgPath*}	replaced by the value of the global-tag at given tag-path *tgPath*, if such a tag exist.
{S:*script*}	will not add to the file-name. Instead it will try to execute the *script* (all characters between the : and the }) as a DigitalMicrograph script prior to saving the image. This is a powerful field which allows to basically run an arbitrary script together with the quicksave. The best way to implement it, is to create a function and install it as a library and then use the field to call this function, but it is also possible to add limited script code directly.
{s:*script*}	Same as above but without reporting the script call in the results window.
{?}	If this field is found anywhere in the template, it will be removed from the filename, but before saving, a user dialog allows modification of the final filename.
{?:*A*|*B*}	Will prompt a dialog of text *A* with displayed default answer *B*. The final answer is used as the field.
{last}	replaced by the last filename saved to (using the tool)
{SUB:subPath}	will store in a subfolder of given name – creating it, if needed. *subPath* does not include the final \ but may in itself contain sub-branching, e.g. *{SUB:mysub\elemental}* would be valid.
{clean:A} {clean:A|B..} {clean}	If this field is found anywhere in the template, it will be removed from the filename, but before saving, all multiple-occurrences of A will be replaced by a single A, and end-standing or leading A's will be removed. Use this to clean up separator-characters at the end. It is possible to add more than one cleaning phrase in the field, i.e. {clean:A|B|CC} would clean all A's B's and CC's in the described way. A pure {clean} field will clean up the defined separator character.
|	A simple vertical line is replaced by the "separator" character defined for QuickSave during the setup. (Usually it is _ or a simple space)
{>>A}	If this field is found anywhere in the template, it will be removed from the filename, but before saving, all occurrences of A are replaced by the separator character. This action is done before any CLEAN action.
{>>A->B}	Same as {>>A} but replaces A by B.
{>>A->B|C->D}	Same as {>>A} but replaces A by B and (afterwards!) C by D.

Some file-template examples

The following examples all assume that – if not stated differently - all open image have the name <MyImage> and that the fields of the tool are set as follows:


1 a short description

The default separator-character is set to: _

automatic numbering

template: `{###}{minitext}|{name}`

purpose: Saves the front image under its name but with leading serial number. If the *minitext* field is not empty, sub-numbering by letters is possible. Using the SHIFT and CONTROL keys when saving allows comfortable incrementing of the numbers.

examples:

Saving the image will save as: 001a_MyImage.dm3.
However, saving it with the CONTROL key pressed will save it under the same name, but increase the counter. Thus saving the next image by clicking again will save the second image as 002a_MyImage.dm3 and the next as 003a_MyImage.dm3 etc.

Using the SHIFT key instead, will rather increase the lettering, resulting in a series of images saved as:
001a_MyImage.dm3
001b_MyImage.dm3
001c_MyImage.dm3

And using the SHIFT and OPTION key alternatively allows a series such as this:

001a_MyImage.dm3	(saved with SHIFT-click)
001b_MyImage.dm3	(saved with SHIFT-click)
001c_MyImage.dm3	(saved with CONTROL-click)
002a_MyImage.dm3	(saved with CONTROL-click)
003a_MyImage.dm3	(saved with SHIFT-click)
003b_MyImage.dm3	

If the *minitext* field would have been empty, there would simply be no sub-numbering with this template.

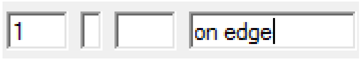
saving different images with same sample comment

template: `{##}|{name}|{text}`

purpose: If the image name (after recording) contains already some information (e.g. the name of the detector). This templates can be used to save *different* images with the same suffix, e.g. describing the sample position.

examples:

Assuming one acquires two simultaneous STEM images and the according images have the name “HAADF” and “BF”, then one could enter into the text field some information, e.g.:


1 on edge

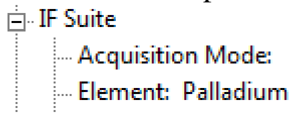
and save both files with two successive clicks as:
01_HAADF_on edge.dm3
01_BF_on edge.dm3

using some image tag info

template: `{##}|{TAG:IF Suite:Element}|{name}`
purpose: Sometimes information is stored in the tags which would be useful in a filename.

examples:

Some images have useful information in their tags. For example: EFTEM images acquired during an elemental map have the following tag:



```
IF Suite
    Acquisition Mode:
    Element: Palladium
```

The template above would add the element name to the filename on saving:
01_Palladium_MyImage.dm3
If the tag wouldn't exist, the file would be saved simply as
01__MyImage.dm3
If you don't like the double __ in this case, you should additionally use the {clean} field described later.

using some global tag info

template: `{##}{GTAG:mytag}{name}`
purpose: Sometimes useful information is stored in the global-tags of the acquisition PC but not in the images.

examples:

If some relevant information is stored in the global-tags of the PC, it can be accessed similar to the image-tags. This is particularly useful, if some scripts store their information in the global tags rather than in every image. In combination with the script-field this can be used to create complex saving templates.

modify & save

template: `{name}{?}`
purpose: This is very similar to using the standard 'save'. A dialog appears with the current image name as suggested filename. After modification, the image will be stored.

always save with same (modified) name

template: `{last}{?}`
purpose: A simple way of saving images with the same, manually adjusted filename

examples:

This template will – on save – always pop up a dialog asking for a filename. The last used filename will be suggested.

save numbered with adjustable title, version 1

template: `{###} {name}{?}`
purpose: A simple way of saving numbered images.
examples:

This template will – on save –pop up a dialog with the current image name plus the automatically given number. Before saving, the user may edit it.

save numbered with adjustable title, version 2

template: `{###} {name:?}`
purpose: A simple way of saving numbered images with the possibility to change parts of the title

examples:

This template will – on save –pop up a dialog with the current image name. After editing it, the image will be saved with the edited name plus the number-prefix. Note the small difference to version 1 which pops-up the final filename. This version will only ask to verify/modify the current image title. Other field-codes are added afterwards. This might be helpful if one wants to keep a strict naming convention.

save with the same name but in a sub-folder

template: `{SUB:worked_on}{name}`
purpose: Sometimes one wants to save modified versions of images as a copy in an separate folder, rather than overwriting the loaded image.

examples:

A (loaded) image with a specific name , e.g.
MyImage.dm3
would be stored (as a copy) in a subfolder:
worked_on\MyImage.dm3

adjusting standard filenames

template: `{name:BF|MAADF}`
purpose: Sometimes acquired images have the wrong 'default' name, e.g. if different STEM detectors are used. One then might want to save the image with a slightly modified default name.

examples:

An acquired STEM image from DigiScan might be named:
BF scan image
by default, regardless of the actual detector. One could use the above template to save such images as:
MAADF scan image.dm3
This could of course be combined with other template options.

using scripts within the templates

template: `{S:addMyScaleMarker(){name}`

purpose: Sometimes it can be useful to execute simple scripts prior to saving an image. f.e. one might want to invoke a script which add the ScaleMarker or some labelling.

examples:

The best way to add little scripts is to write a script-function which acts on the front-image and then install this function as a library. This is an example script:

```
 void addMyScaleMarker()
      {
      getfrontimage().imagegetimageDisplay(0).ApplyDataBar()
      }
```

With this script installed as a library, the template above would first add the scale-marker to the image, and then save it under its current name.

Note: If the filename-template contains *only* script-fields, it will always return a null-string as filename. As the script does not save files if the filename is a null-string, this can be "abused" to have buttons which act on front images but do not save. There are easier ways for this purpose though…

using scripts and tags for powerful automated naming

template: `{S:DoMytag()}{TAG:dummytg}{name}`

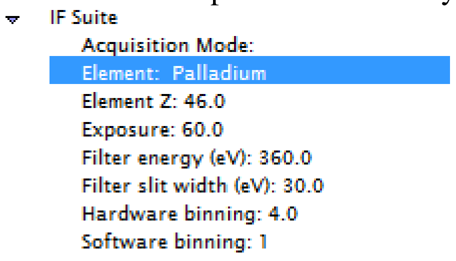
purpose: The combination of scripts and tags allows for powerful combinations. A script can be used to set a tag which is then read by the template. This works, because fields are evaluated sequentially on the fly.

examples:

Say one wants to 'built' a filename based on information stored in the image tags, f.e. the element in an EFTEM map, one could make a script which stores this information in a tag of the image:

```
void DoMytag()
     {
     string element = getfrontimage().GetStringNote("IF Suite:Element")
     getfrontimage().SetStringNote("dummytg","_"+mid(element,1,2)+"_")
     }
```

With such a script loaded as library, the template above would act on an image with the tags



```
▾ IF Suite
    Acquisition Mode:
    Element:  Palladium
    Element Z: 46.0
    Exposure: 60.0
    Filter energy (eV): 360.0
    Filter slit width (eV): 30.0
    Hardware binning: 4.0
    Software binning: 1
```

as follows: _Pa_MyImage.dm3
and with images without that tag as: MyImage.dm3

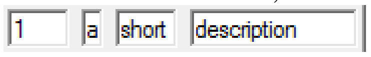
using the "cleaning" field

template: `{##}{minitext}|{name}|short|{text}{clean}`

purpose: This templates can be used to save *different* images with the same suffix additional information, e.g. describing the sample position and magnification. However, it takes care of dispensable separator characters.

examples:

Assuming one acquires two simultaneous STEM images and the according images have the name "HAADF" and "BF", then one could enter into the text field some information, e.g.:



1 a short description

and save both files with two successive clicks as:

01a_HAADF_short_description.dm3
01a_BF_short_description.dm3

If, however, the same template is used without the description and the short description in the text fields, those images get now saved as:

01a_HAADF.dm3
01a_BF.dm3

Note the difference: Without the {clean} field, it would instead be:

01a_HAADF__.dm3
01a_BF__.dm3

using the "replace" field

template: `{name}{>> }`

purpose: This templates can be used to replace all occurrences of a *white space* character by the default separation character (e.g. the underscore). The difference to the `{name:A|B}` field is, that the `{>>A->B}` field replaces the strings from the final name after all templates have been evaluated, whereas the `{name:A|B}` field only acts on the original image name.

examples:

Assuming that the default separation character is the underscore _ :

BF (DS) is saved as BF_(DS).dm3