



## {Profile}

# Photo Size Optimization

## Specifications

V1.8

Version	Date	Redactor	Modification
1.0	08/05/2012	Manuel LAURENT	Creation
1.1	06/07/2012	Laura BG	Add details + specifications Sub1 (+ notes for Sub2)
1.2	11/07/2012	Laura BG	Change size for primary photos (format must be 3:4) = change display of the component + size saved
1.3	13/07/2012	Julien S.	Add OBO Impacts & specifications
1.4	03/08/2012	Laura BG	- Add special case for small photos: keep original size (3.4.4) - Infra. Impacts: Add volumetry analysis (3.8.3) - Change thumbnails generation mechanism for album photos (4.2.2)
1.5	09/08/2012	Laura BG	Actions to measure the impact on the platform (3.8.5)
1.6	20/08/2012	Laura BG	Change format of log + add infos & flag (3.8.5.1)
1.7	05/09/2012	Laura BG	Add analytics needs – insert for dimensions + upload time + new log when user meets an error (3.7)
1.8	17/09/2012	Laura BG	Specs Sub2: Display bigger photo

## Summary

---

1	Introduction .....	3
1.1	Presentation & context .....	3
1.2	Expected results .....	3
1.3	Perimeter .....	3
1.4	Subdivisions.....	3
1.4.1	Sub1: Upload bigger photos.....	3
1.4.2	Sub2: Display bigger photos.....	4
1.4.3	Sub3: Use the photo server.....	4
1.5	Glossary .....	4
2	Global overview .....	5
2.1	Flow overview .....	5
2.2	Upload photo flow .....	5
2.3	Thumbnails and moderation flow.....	6
3	[Sub1] Upload bigger photo.....	7
3.1	Perimeter .....	7
3.2	Configuration .....	7
3.3	Website: Signup .....	8
3.4	Website: My profile .....	10
3.4.1	Primary photos.....	10
3.4.2	Album photos.....	10
3.4.3	Configuration of the uploader .....	12
3.4.4	Specific case for small photos: keep original size .....	12
3.5	OBO .....	14
3.5.1	Photo moderation .....	14
3.5.2	Photo display.....	20
3.6	Mobile .....	22
3.6.1	Display.....	22
3.6.2	Store bigger photos.....	22
3.7	Analytics needs .....	23
3.8	System infrastructure impacts .....	24
3.8.1	File size comparison .....	24
3.8.2	Thumb / Full requests repartition on dating.....	24
3.8.3	Thumb / Full : volumetry repartition on dating .....	25
3.8.4	Photo server .....	26
3.8.5	Measure the impacts on the platform .....	27
4	[Sub2] Display bigger photo.....	28
4.1	Presentation & objective .....	28
4.2	Perimeter .....	28
4.3	Website .....	29
4.3.1	Mechanism.....	29
4.3.2	Configuration .....	30
4.3.3	Define a new type of photo - How To .....	31
4.4	OBO .....	33
4.4.1	Display in OBO.....	33
4.4.2	Change thumbnails generation mechanism for album photos .....	33

4.5	Mobile .....	36
4.6	Infrastructure impacts .....	36

[\* In this document all the text highlighted in yellow is non-validated elements or questions not answered yet.]

## 1 Introduction

---

### 1.1 Presentation & context

---

The product team wants to optimize the profile picture displayed on the website.

Currently the maximum size of photos is 300x350px – always portrait format for both main or album photos - and we want to increase it to a maximum value: **900x675px** = this is the maximum size we can display on a small screen. This optimization will allow a better visualization of profiles with a better quality.

### 1.2 Expected results

---

No specific figures are expected for now.

Analytics team will follow the number of uploaded pictures.

### 1.3 Perimeter

---

- Product: Dating.
- Countries: All countries – rollout to be scheduled step by step. Depending on the infrastructure impacts

### 1.4 Subdivisions

---

#### 1.4.1 Sub1: Upload bigger photos

- **Objective:**
  - Allow the member to upload bigger photos (max size= 900x675px) with the new photo uploader
  - Keep the display with the current format (300x350px)
- **System impacted:**
  - Photo uploader on the website (signup + my profile)
  - OBO moderation of the photo + display in the different cues.
  - Mobile - WAP
  - Photo server
  - DB
  - Hard drive
  - CPU
  - Front: php to upload the photo more used – take more time (increase the handler php?)
  - Bandwidth when upload the photo
- **Adherences:**
  - Modify photo server to force the size of the photo displayed
  - Wap: use the new photo server (new URL)
- **Rollout:**
  - Test on Meetic SP to measure the infrastructure impacts

### 1.4.2 Sub2: Display bigger photos

- **Objective:**
  - Display the bigger photos on my/his/her profile page by clicking on the layer (display photo full)
- **System Impacted:**
  - Website photo layer
  - FAQ, add a content to explain how to get a full picture
  - Akamai bandwidth
  - Traffic
  - Mobile – Touch, App, Wap
  - CRM display? → Resize?
- **Adherences:**
  - Phasing with the layer photo review
- **Rollout:**
  - Test on one country and follow analytics figures.

### 1.4.3 Sub3: Use the photo server

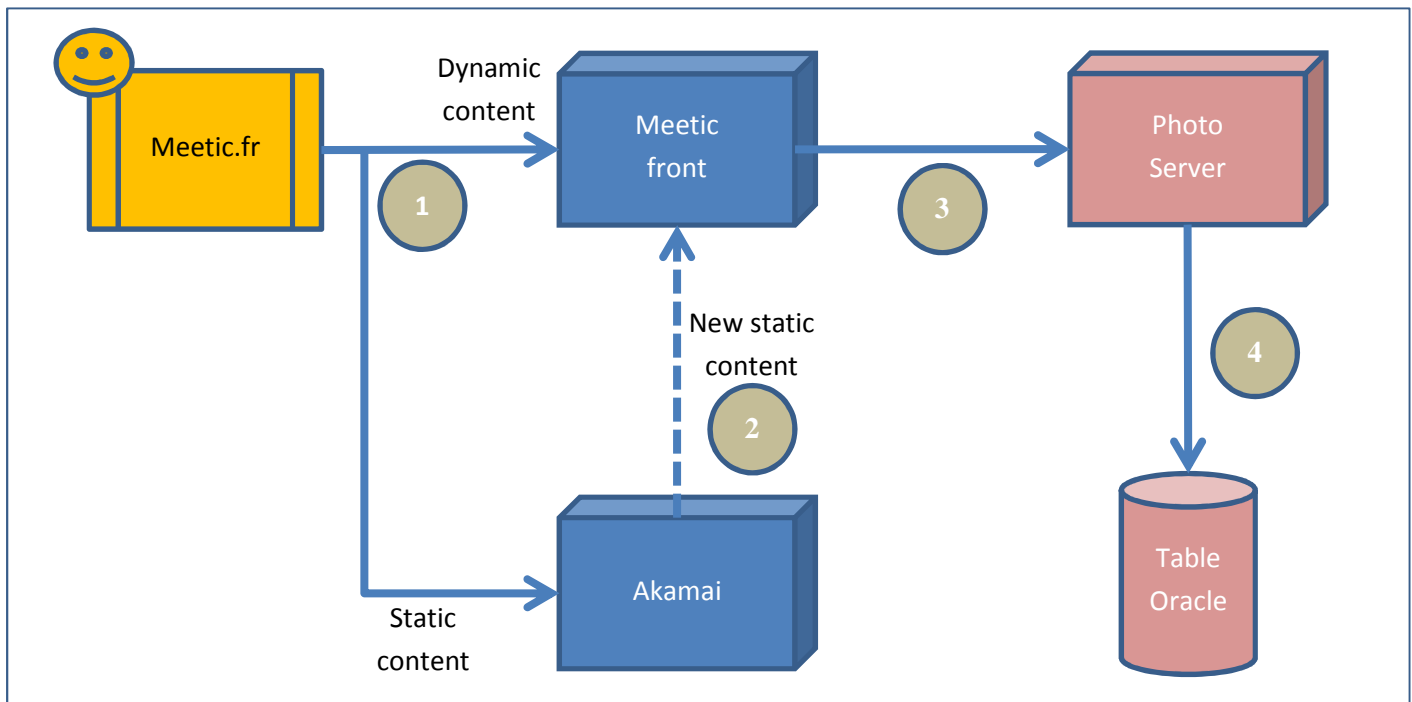
- **Objective:**
  - Use the photo server to generate and manage the thumbnails.
- **System Impacted:**
  - Website photo layer
  - OBO
- **Adherences:**
  - N/A
- **Deployment:**
  - Country / country due to infrastructures impacts.

## 1.5 Glossary

Word	Definition
Akamai	Meetic's supplier for static content, CSS, JS and pictures. (CDN : Content Delivery Network)

## 2 Global overview

### 2.1 Flow overview



1. Currently when a member visualizes a photo it is called from Akamai server. This solution decreases the load on Meetic front servers.
2. If it's a new photo (or all kinds of static content: CSS, JS, etc.), Akamai calls Meetic front to get this new content.
3. For photo content, Meetic front calls the photo server to get the picture.
4. Depending on some parameters (round borders, blur, size), the photo server gets the picture from the DB (photo are stored in a BLOB), modify the photo and send the data back to Meetic front.

### 2.2 Upload photo flow

The user can upload:

- 5 portrait photos max
- 20 album photos max

All the photos are currently saved on a portrait format: 300x350px (for both main & album photos).

On the portrait photos, the user should be clearly recognizable (the CC has many criterias to validate a portrait photo) whereas for the album photos it can be more personal: the user can upload his cat, his house or holidays photos... That's why we have 2 different moderation cues because the validation criterias are not the same for portrait or album photos.

For the moment we have 2 different components but functionally identical (it's the same from a user point of view):

- The new photo uploader: with JS + Flash.  
Currently available on Meetic SP + Lexa NL (*rollout on all countries scheduled for the week of the 24<sup>th</sup> of july*)
- The former uploader: with Flash only  
Available on all countries where the new component is not.

➔ For more details about the new photo uploader, please refer to the specifications:

<http://jira4.ilius.fr:8080/secure/attachment/15322/%5BProfile+Capture%5D+Specs+-+Photo+Uploader+NG-v1.0.pdf>

## 2.3 Thumbnails and moderation flow

When a user uploads a photo (which is cropped & resized to 300x350 thanks to the uploader component), it goes into the right moderation cue (portrait or album).

- For the portrait photos, the customer care generates manually the 2 thumbnails displayed on site – to center his face:
  - Thumbnail 94x94
  - Thumbnail 44x44

This step is necessary for the portrait photo in order to have a thumbnail centered on the face of the user – because on the lists (etc.) only the 1<sup>st</sup> validated portrait photo is displayed.

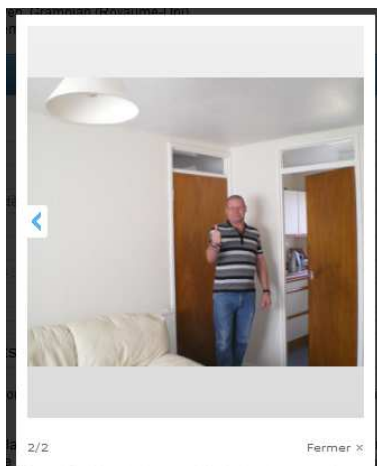
- For the album photos, the thumbnail 94x94 is automatically generated by OBO when photo is approved.

The thumbnails are generated from the photo 300x350px.

- The thumbnails are used on the lists, home log, daily matches and profile page.



- On the profile page, we use 3 kinds of photos:
  - 1:** Main portrait photo = photo full 300x350 (resized with the html / width hard coded)
  - 2:** Thumbnail of portrait photo = 94x94 (generated manually in OBO)
  - 3:** Thumbnail of album photo = 94x94 (generated automatically in OBO)
- The photo full size 300x350px is only displayed on the layer. When you click on zone **1**, a layer is displayed over the page to show the original photo (as it is stored on DB).



### 3 [Sub1] Upload bigger photo

---

The objective of this Sub1 is to enable the user to upload bigger photos (max size= 900x675px) with the new photo uploader BUT at the same time we don't want to impact the display. We also want to change the format of our photos to fit 4:3 (instead of 3:3,5 now).

So we will upload & store big photos on DB but we'll keep on displaying photos 300x350px on site (the display will be managed by the [photo server – cf 3.7.4](#))

#### 3.1 Perimeter

---

➔ No AB Test

➔ Meetic SP only for the 1<sup>st</sup> tests.

This project will be developed only on the new upload component ("Photo uploader NG").

The former photo applet won't be affected by this project.

Currently the new component is available on:

- Meetic SP – 100% signup
- Lexa NL – 100% signup
- Rollout on all countries scheduled for the 24<sup>th</sup> of July – 100% signup
- From the week of the 24<sup>th</sup> of July : Meetic SP + Lexa NL 25% on the My Profile page (Then rollout on all countries depending on the figures)

Considering that this project 'bigger photo' can have huge impacts on the platform (server, DB, disk, bandwidth...) we need to measure country by country the volumetrics, increase gradually the charge in order to prevent and anticipate the infrastructure impacts (add server, add disks etc.)

Therefore, as we'll have the new photo uploader on signup on all countries from July 24<sup>th</sup> we consider it's not reasonable to launch bigger photo on the new photo uploader directly without configuration because it will impact all countries at the same time; we need time to control the impacts on the platform (server, DB, disk...) and monitor the volumetrics.

➔ That's why we've chosen to launch bigger photo on Meetic SP only – on the new uploader (signup + profile page) - to measure the impacts. The other countries where 'Photo uploader NG' is activated should not be affected by the 'Bigger photo' project.

#### 3.2 Configuration

---

Considering the perimeter of the project, we should find a way to activate the 'Bigger photo' project in the config of the country.

So for Meetic SP, we should define a new variable in:

```
\libs\product\sites\es\meetic\www\config.inc
```

Example of variable to activate bigger photo:

```
$siteConfiguration['GLOBAL']['UPLOAD_BIGGER_PHOTO'] = TRUE;
```

By this way, we can activate the project country by country without impacting all the countries where 'Photo uploader NG' is already activated.

Reminder: The 'Bigger photo project' will be available only on the new photo uploader.

➔ If the variable is not defined in the config of a country, nothing changes and the photo uploader stays the same (on the signup or on the my profile page). The size of the photo stays 300x350px in DB.

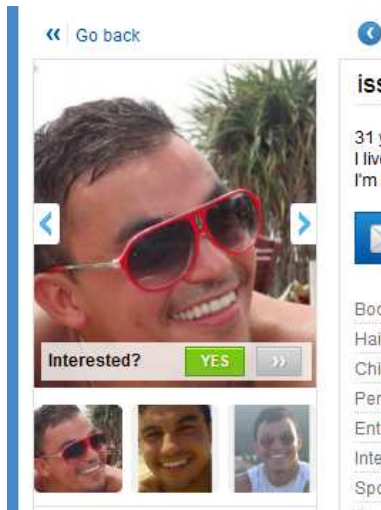
### 3.3 Website: Signup

On the signup the user can upload his main profile photo. This photo will be visible by the other members in the lists etc.

As the main profile photo is displayed on a portrait format on the profile page, we must keep this format (portrait) but we also want the photo **to fit 3:4** instead of 3:3,5. Indeed the current format doesn't match with any camera, this is just an historical format that we don't have any interest to keep.

Moreover we can't let the user upload a landscape photo without cropping it otherwise the photo displayed on the profile page will be resized and we'll see big white margins – this is not what we want.

*Portrait format*



*If we could upload landscape format*



So the objective for this project is to keep the portrait format for main photos but store it with a higher resolution and change the ratio to fit 3:4.

Today, no matter the original size & format of the photo, the uploader component crops it to a portrait format and resizes it to 300x350px max (format=3:3,5)

So for the signup and for all main photos (5 max) we want to keep the same mechanism with the crop to fit a portrait format but the resize should be done with the following max size: **506 x 675px** (width x height).

(> This max size has been defined by our designer to fit the main screen resolutions)

As we want to fit 3:4 on all primary uploaders we should change the display size in order to be proportional with 3:4 and save the photo 506x675px.

This ratio doesn't match with what we currently have on the uploader so we should change the display + the size of the photo saved.

For the photo uploader on the signup, we don't change the functional but only the size of the photo saved and the display of the uploader:

- **On the layer** (displayed by clicking on the link "Upload photo"): the size of the uploader component changes in order to **be proportional with 3:4** and when the user clicks on "Save & Continue" the photo is cropped according to what is displayed in the frame and sent to the server with that exact max size **506 x 675px**

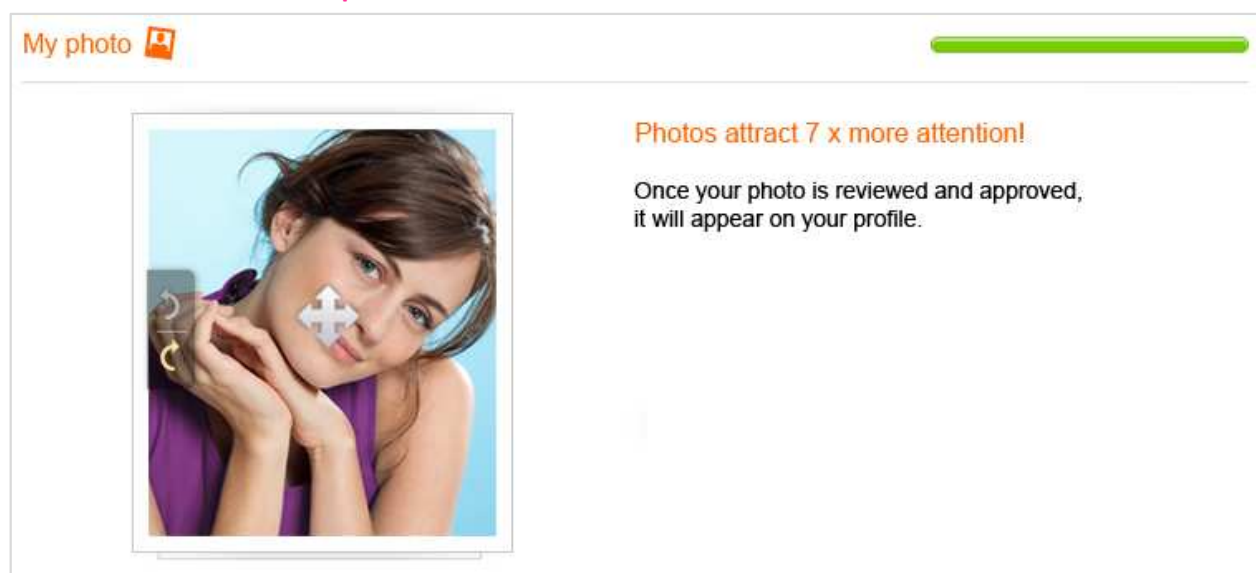




**Caution:** the photo block in the layer is smaller than what must be sent to moderation when the photo is uploaded = **506 x 675px** (width x height) - but it respects the proportions.

➔ This means that the photo is resized 'for real' to **506px** width max but the photo displayed to the user is smaller in the layer (cf mockup)

- **On the photo page:** same as for the layer = **change the size of the component to fit 3:4** + save the photo with this max size **506 x 675px**



➔ **Nota:** the sizes defined above are MAXIMUM sizes (saved & component size)

- Please refer to the special case when user uploads small photos – 3.4.4 -

➔ **Reminder:** This new max size must be saved only for country where `UPLOAD_BIGGER_PHOTO` is activated in the config. Same for the display in 3:4.

If the project is not activated in the config of a country, we keep uploading the photo 300x350px and format 3:3,5

## 3.4 Website: My profile

On the my profile page, the user can upload 2 kinds of photos:

- Primary photo (5 max) – the 1<sup>st</sup> one can be uploaded from the signup
- Album photos (20 max)

Unlike the signup where we can upload only primary photo (main photo) and we want to keep the portrait format, for the album photos we don't have any interest in keeping the portrait format – no constraint on the profile page display.

This means we'll have 2 different formats depending on the kind of photo we want to upload:

- Primary photo = portrait format mandatory with max size = **506 x 675px**
- Album photo = all formats accepted (portrait or landscape) with max size = **900 x 675px**

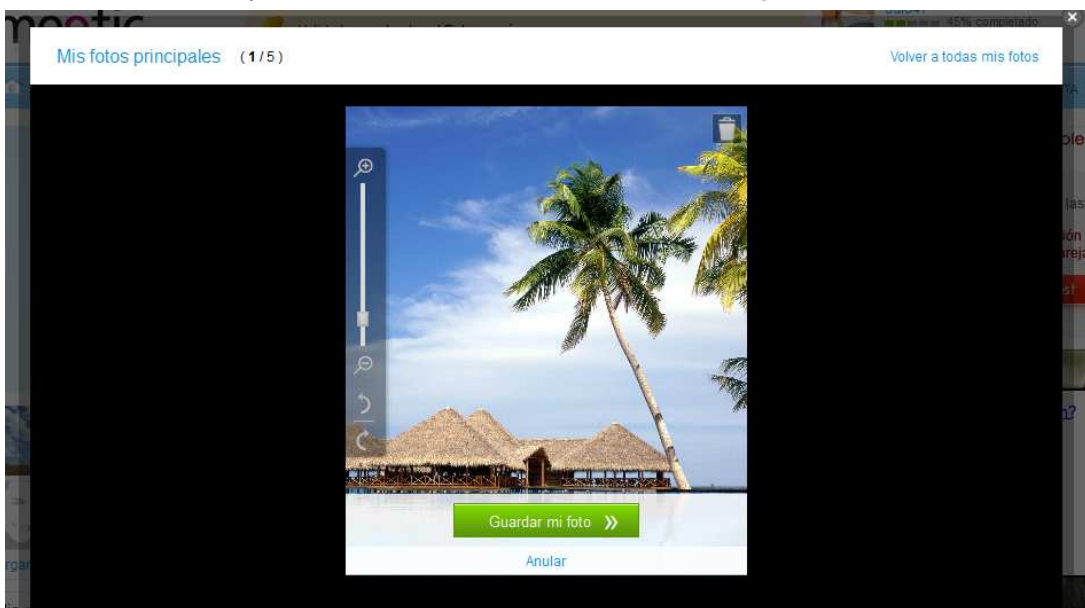
This has an impact on the photo uploader component.

Today the component is common for primary and album photos whereas for this project we must dissociate it depending on the kind of photo: primary or album.

### 3.4.1 Primary photos

For primary photos, it is exactly the same as on the signup:

- We keep the portrait format
- The size of the uploader changes in order to be proportional with 3:4
- The photo is saved with this max size **506 x 675px**



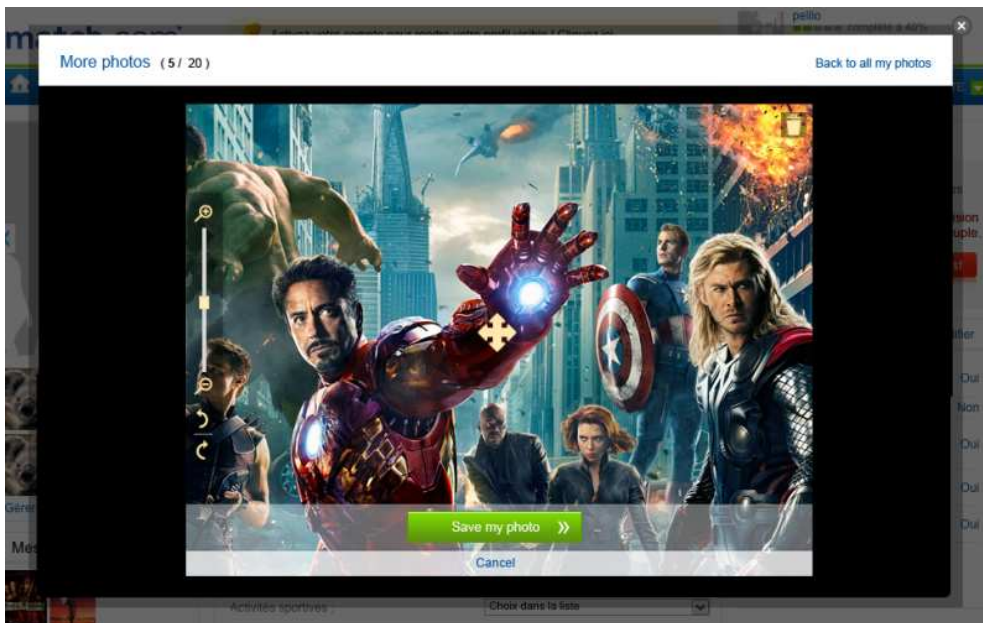
### 3.4.2 Album photos

For the album photos, we don't need to keep only the portrait format.

This does not make sense because for viewing photos in large, landscape format is much more suitable.

Therefore, for album photos only we need to change the size of the uploader component to allow the upload of landscape photos – we remove the constraints of the portrait format.

The uploader component should be displayed with that size: **620x465px** and when the user clicks on “Save my photo” the photo is cropped according to what is displayed in the frame and then sent to the server with that exact max size **900 x 675px** → goes into the album photo moderation cue as before.



- If width > height: we take the width as a reference: the photo displayed on the component should be resized to width=900px
- If width < height: we take the height as a reference: the photo displayed on the component should be resized to height=675px

#### Examples:

- Original photo = 1920x1200px → After resize = 900x568px
- Original photo = 798x1200px → After resize = 448x675px

For some others specific formats we must crop a part of the photo if after resize it doesn't fit the expected format (size of the component).

#### Example:

- Original photo = 1200x1200px → After resize = 900x900px but we display in the component only 900x675px so when saving the photo we'll crop the equivalent of a 225px in the height

*Original photo (resized)*



*Photo saved after the crop to fit our max size:*



The photo is cropped according to what is displayed in the component => the user can drag the photo to center it if needed → same behavior as the current one.

→ **Nota:** the sizes defined above are MAXIMUM sizes (*saved & component size*)  
 - Please refer to the special case when [user uploads small photos – 3.4.4](#) -

### 3.4.3 Configuration of the uploader

In order to avoid having 2 different components for the uploader photo, the component should be as configurable as possible: set the size of the component and the format to be stored dynamically

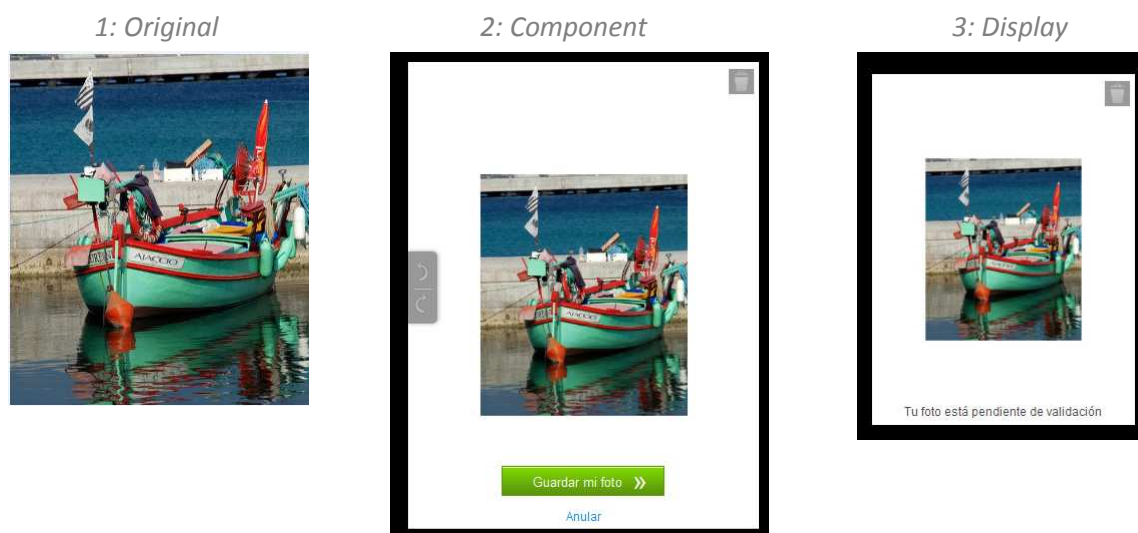
→ Values should be defined in a config file.

### 3.4.4 Specific case for small photos: keep original size

For users uploading photos smaller than the size of our component & the size we want to save in DB, we'll have a display issue → we'll lose the quality of the original photo and add big white margins which is not the objective!

Example:

1. The original photo selected by the users (=300x350px)
2. Display of the selected photo in the uploader component → we add margins to fit the format we want to save)
3. Display of the photo on site: size is forced by the photo server on Sub1 = 300x350px but with margins (in DB the photo saved =506x675px)







This means that even if the photo is smaller than the max size we want, we add margins to fit the format and we rescale it → for small photos it becomes unreadable.

1. Original photo = 200x175px
2. Photo as it will be saved in DB
3. Photo displayed in the uploader component = 137x120px
4. Photo displayed on site = 66x58px (in a frame of 300x225px to fit our 4:3 format)



So if the size of the photo selected by the user is < 900x675px for album OR 506x675px for primary photo (sizes defined in the config file) we should adapt the size of the uploader component and the size of the photo saved in DB in order to keep the original size of the photo and its quality.

Original photo	Size of the photo saved in DB	Size of the component (add margins but keep the original size)	Display on site (the original image remains readable)
			

Therefore, it means that the sizes (height + with) of the component + saved size defined in the configuration will be the MAXIMUM size.

So on long term (after sub2) we'll keep on having different sizes of photo to display, depending on the original photo size uploaded by our members → we'll only have a maximum size (900x675px for album OR 506x675px for primary photo)

➔ **Reminder:** This new max size must be saved only for country where `UPLOAD_BIGGER_PHOTO` is activated in the config. Same for the display in 3:4.  
If the project is not activated in the config of a country, we keep uploading the photo 300x350px and format 3:3,5

➤ **Action:** MR#1 – Tudor's Team

## 3.5 OBO

We will need to make some developments on the OBO back office in order to make sure it supports bigger photos. Those developments will mainly consist of arrange the way the photos will be displayed without breaking the page or decreasing the productivity of the customer service in charge of the photos moderation. We will also have to modify the photo approval step done in a popup.

### 3.5.1 Photo moderation

We have two types of pictures to moderate:

- Portrait
- Album

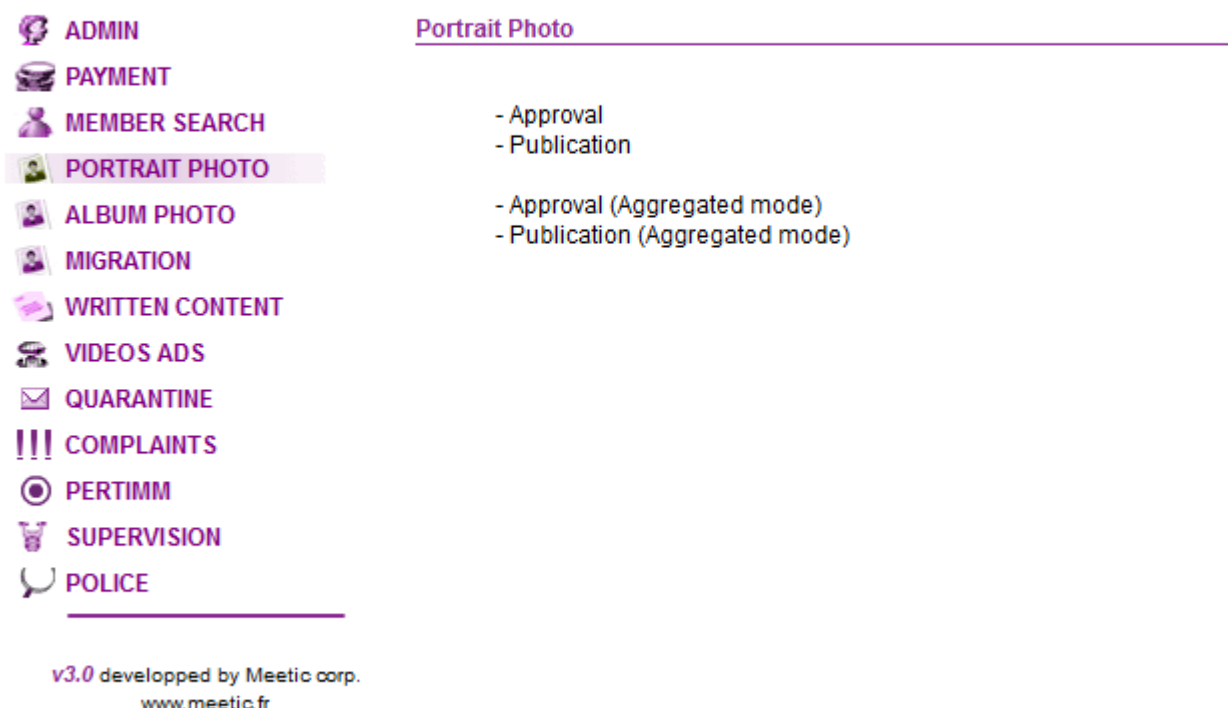
#### 3.5.1.1 Portrait moderation access

The moderation of Portrait photos is accessible by clicking of the “PORTRAIT PHOTO” link located in the menu. Here you’ll find four links:

- Approval
- Publication
- Approval (Aggregated mode)
- Publication (Aggregated mode)

The moderation of Portrait Photo is done in two steps:

- First step is to create the profile thumbnail and approve the photo (Approval links).
- Second one is a security check to make sure the photo is valid and publish it (Publication links).





### 3.5.1.2 Album moderation access

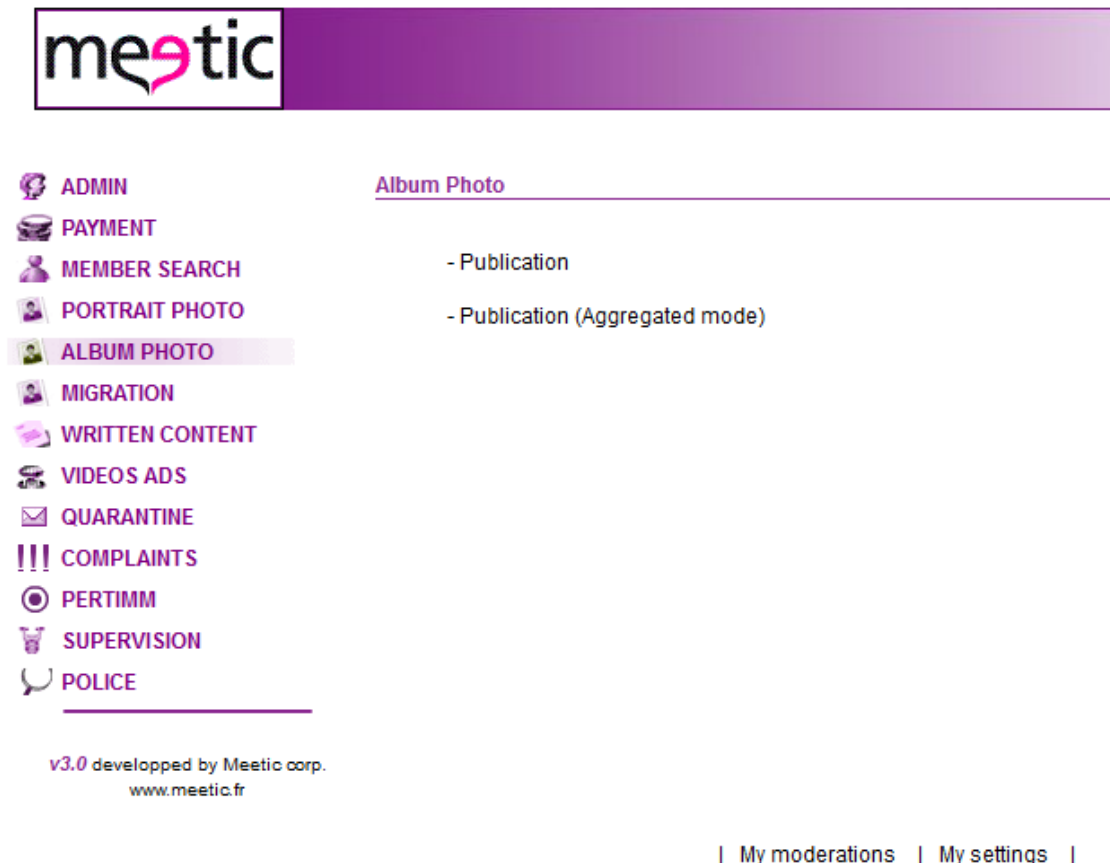
The moderation of Album photos is accessible by clicking of the “ALBUM PHOTO” link located in the menu.

Here you’ll find 2 links:

- Publication
- Publication (Aggregated mode)

The moderation of Album Photo is done in one step:

- Simple security-check to make sure the photo is valid and publish it.



### 3.5.1.3 Current Portrait photo approval step

Here is the popup you'll see while approving a portrait photo:



When loading this popup, a temporary image will be created and scaled to **400x350px**. This picture is the one displayed in the popup and every action done such as crop and rotation are processed on this image. The overlay box on the picture is a JS Script that we use to crop the picture and generate the thumbnail. Once you choose the area to crop by dragging the overlay box, you have to click on the **“Display”** button to have a preview of the generated thumbnail. When you're good to Approve, just click on the **“Approve”** button to finalize this step.



### 3.5.1.4 Compatibility of the approval step with bigger photo

The main idea is to duplicate this template and rework it with jQuery so we can handle the picture in its original size (but still displayed to fit the popup to avoid impacts for the customer care work) and make the cropping and rotation on client side.

*This idea needs to be validated by the customer care service as it will certainly change the UI of the popup.*

→ SOLUTION TOO COMPLEX – not approved

If this idea is too complicated to implement or not validated by the customer care service, we could have the same behavior as now except that the temporary picture will be a copy of the original photo and that it has to fit the popup by adding the width and height attributes in the html code.

This picture will be reworked each time we crop or rotate it with PHP and MagickWand (the current php extension used to work with images) and then reload in the popup.

To do so, we will have to transcribe the action coordinates of the scale picture from the popup into the temporary original one when dealing with bigger photos.

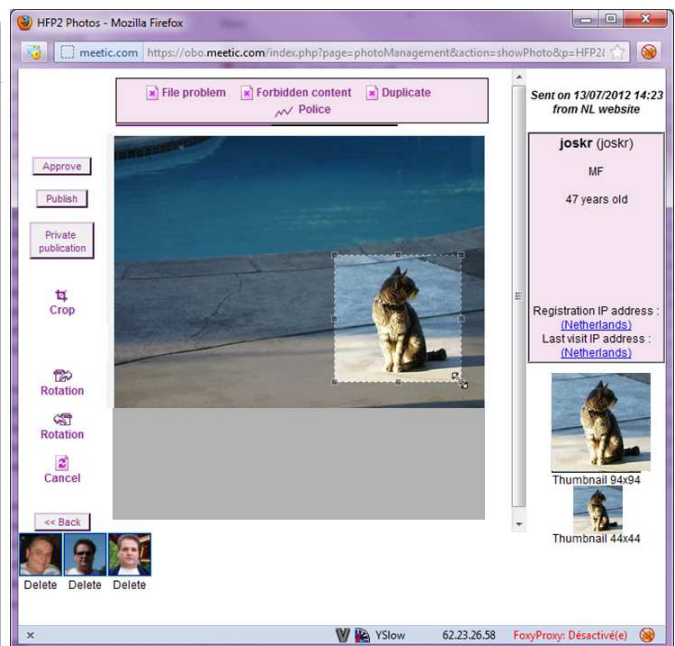
// Estimate the LOE for each solution (jQuery or not) → depending on the LOE of each solution I'll talk to the CC = if use jQuery is easier and will impact the display in the popup, I'll have to "sell it" to the CC.

→ SOLUTION TOO COMPLEX – not approved

*Current display in the popup*



*Proposal of a display using Jcrop (for example)*



**Display Button** => will reload the preview frame on the right and by calling photos/photoPreview.php with the coordinates. This page will generate the link to the temporary image and call photos/makeThumbnail.php to generate them.

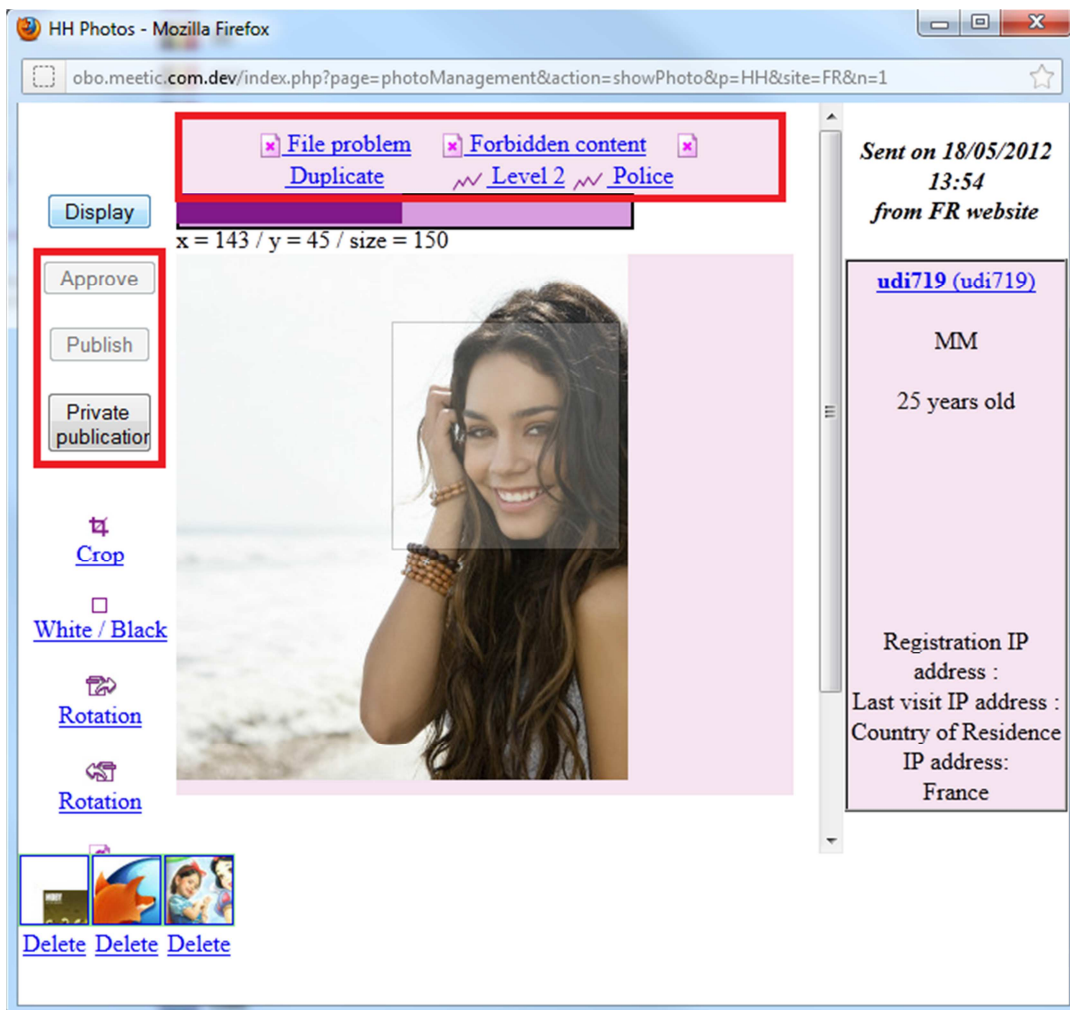
**Crop action** => This will reload the popup by calling photo/photoSizing.php with the coordinates of the overlay box.

**Rotation Action** => This will reload the popup by calling photo/photoSizing.php with the coordinates of the overlay box (not sure they are used though) and the direction of the rotation (droite => right, gauche => left).

Once the picture is approved or sent to moderation, we will rescale the temporary picture to 400x350px in order to save the hard drive space (=useless to store the big photo on the local OBO Server).

The picture will stay in the temporary folder for 3 days and might be used by the customer care service to check if a problem appeared during the approval process.

➔ **CAUTION:** Do not remove this behavior = temporary photos must be kept on the local server.



**Approve button** => Calls photos/photoManagement.php with a parameter "action" set to "validate" and also provides the overlay box coordinates to that will be used to generate the final thumbnail. There is also a parameter "sp\_mode" set to 1 that means that the photo needs to be published

**Publish button** => It's the same action as the approve button except that the parameter "sp\_mode" isn't provided meaning that the photo won't need a publish action.

**Private Publication button** => Calls photos/photoManagement.php with a parameter "action" set to "private\_validate". This option transfers the photo into the album photo moderation queue when the picture can't be used as a portrait one.

**File problem** => Calls photos/photoManagement.php with a parameter "action" set to "delete" and also provides the reason of the refusal which is "invalidFormat" in this case.

**Duplicate** => Calls photos/photoManagement.php with a parameter "action" set to "delete" and also provides the reason of the refusal which is "noMail" in this case.

**Forbidden content** => Calls photos/photoManagement.php with a parameter “action” set to “delete” and also provides the reason of the refusal which is “CGU” in this case.

**Level 2** => Calls photos/photoManagement.php with a parameter “action” set to “escalate”.

**Police** => Calls photos/photoManagement.php with a parameter “action” set to “niv3”.

3.5.1.5 Portrait and album photo publishing

Once you’re done with the approval of a photo, you’ll need to publish it. This step is also used as a last security check to make sure the photo is valid.

Just like the approval step, a temporary file scaled (400x350px) is generated for each picture so we have nothing to do here

➔ The bigger photo project shouldn’t have any impact on the publication lists (portrait & album)

- index.php?page=validPrivatePhotos&action=showPhotos
- index.php?page=validPhotos&action=showPhotos

Photos Publication (Aggregated mode)

For the langage : NL - PHOTOS PUBLICATION MF sub (2)



cascade\_ni - MF cascade\_ni



 cascade\_ni - MF - 25 years old - Date : 04/07/2012 12:19:32

☒ Public photo OK

☐ Private photo OK

☐ File problem

☐ Forbidden contents

☐ Duplicate

☐ Supervisor

☐ Police

Registration IP address :  
Last visit IP address :  
Country of Residence IP address:  
France

Comments list

	ToU	PUB	M-A	SPAM	OTHER
	-/-	-/-	-/-	-/-	-/-
	-/-	-/-	-/-	-/-	-/-
	-/-	-/-	-/-	-/-	-/-



{Profile} Photo size optimization v1.8 - Dating

Page 19 / 36

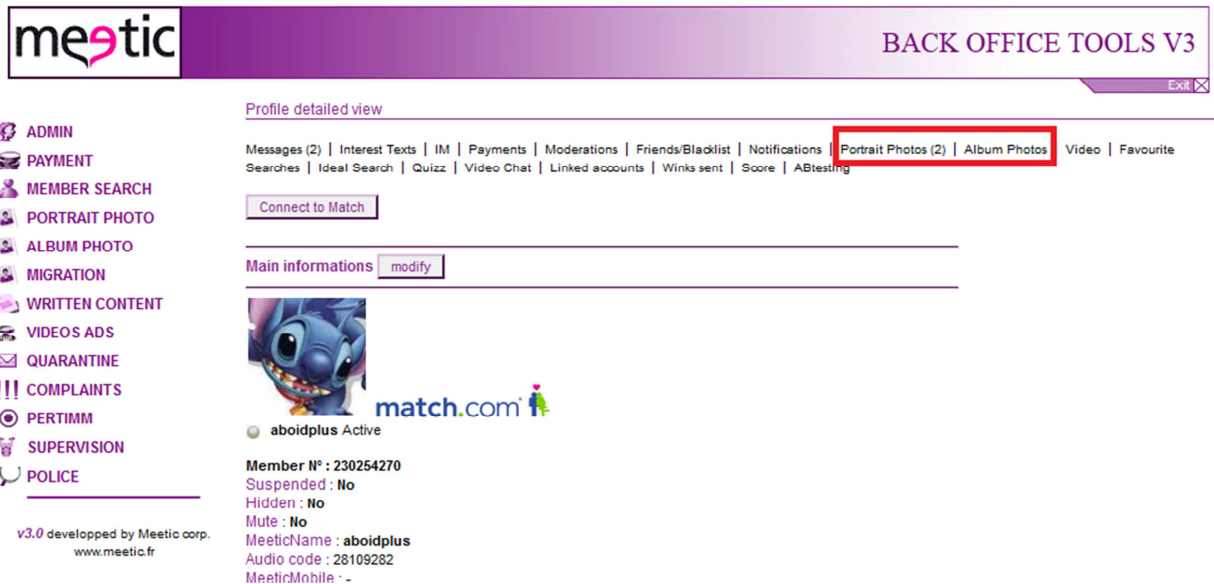
### 3.5.2 Photo display

The photos are displayed in 3 places in OBO:

- In the popup available through two links in the member profile
- During the approval step moderation
- During the publish step moderation

#### 3.5.2.1 Display of portrait/album photo from a member profile

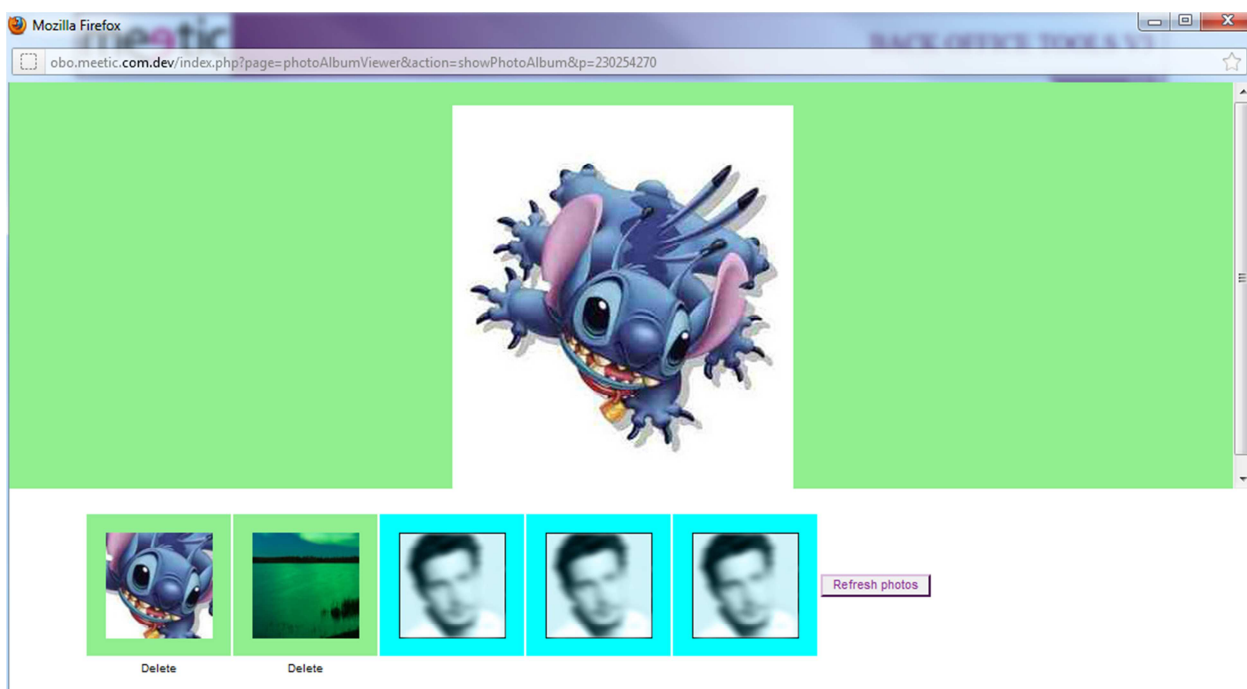
On the member's profile, we have two links that opens a popup to display the portrait or album photos:



When opening the portrait photo popup, the image is taken from the database as it is and is not rescaled.

It's a bit different for the album photo as the PHP script includes a scaling process on fly to make sure the picture will fit the popup.

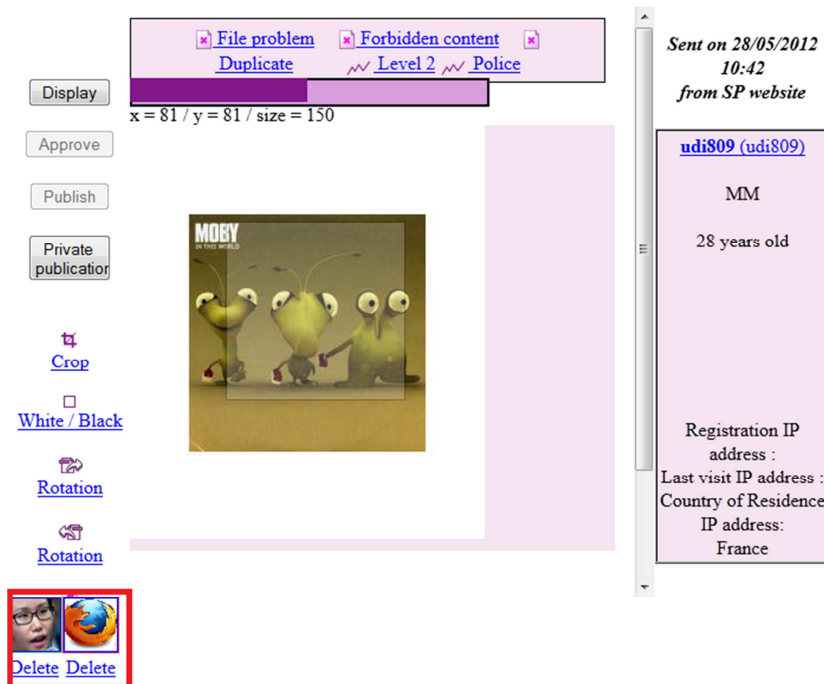
➔ As we don't want to impact the display, we could use the same scaling script in the portrait photo popup to deal with bigger photo or we could also use the max-width max-height CSS attributes with the img tag in the template to avoid image processing.



### 3.5.2.2 Popup during the approval step

If the user being moderated already has other validated photo(s), it will be displayed at the bottom left of the popup. When you click on one of the thumbnail, a new popup will open.

➔ The script being used to display the picture in this popup is the same as the one in the member profile so there won't be another modification to do.



### 3.5.2.3 Popup during the publishing step

Just like during the approval step, if the user being moderated already has other validated photo(s), it will be displayed at the bottom left of the box.

When you click on one of the thumbnail, a new popup will open.

The script being used to display the picture in this popup is the same as the one in the member profile so there won't be another modification to make.





## 3.6 Mobile

### 3.6.1 Display

For the mobile we have different environments:

- Touch
- Mobile app
- Wap

Today the Touch & Mobile App use the new photo server to get the URL of the photo.

So for this Sub1, as the photo server will force the size of the photo to 300x350px, this doesn't have any impacts on the Touch & the App – for the display.

Unlike the Touch & App, the Wap still gets the photo from the former photo server which uses a different algo & encrypted datas for the resize (or other operations).

For the wap we have many different sizes of photo because each photo is displayed depending on the size of the mobile screen. So we can have for a same photo, a display from 10px to 300px.

This means that **for the wap, the mobile team should migrate to the new photo server** using the new URL where we can add different operations (scale, resize etc.).

This way and because the server will force the display to 300x350px, we won't impact the display on the wap.

- Url with the former photo server (*encrypted data – structure to be analyzed*):  
<http://photomobile.meetic.com/mobile/eX53XFRYWH9QXlhczQICXgJFR0EIBAZFAkIR/photo.jpg>
- Url with the new photo server (*encrypted data = different structure*):  
<http://picda.ilius.net.recette/photo/e3VwWFFWVXpQXI9UaQILUQZKRzR.ZXA-/>

➔ For Sub1 the main objective is to don't impact the mobile (touch, app, wap) even if we upload bigger photo from the site.

Mobile's team has checked that the display with bigger photos is not broken on the 3 environments and everything is ok. This is because the photos are already rescaled today to fit the width of each mobiles screen.

### 3.6.2 Store bigger photos

As the upload mechanism is completely different on mobile, the upload won't be affected on this Sub1.

The site will save bigger photos but we'll keep the same current size from mobile (300x350px).

The component will be changed by the mobile team in a second step:

- Amelie Billoud would write the functional specifications
- Sylvain Garrigues's Team will develop it

➔ The storage of bigger photos from the mobile is a different project and must be added to the mobile roadmap to be coherent with the site. Otherwise we'll continue uploading photos with the wrong format from the mobile and we'll always have a difference in DB. Upload photo from mobile should also distinct the 2 formats we now have :

- 3/4 for portrait photos
- 4/3 for album photos.

NB: As from the WAP + Iphone the photo is uploaded directly by email, an analysis should be done to define the impacts & the risks.

➔ This won't be delivered with Sub1

➤ **In charge:** Sylvain Garrigues & his team (mobile)

## 3.7 Analytics needs

### 3.7.1.1 Add dimensions on DB

For Analytics purpose, we want to save in DB the width & height of a photo when bigger photo is activated.

Today we already log some infos in a log (.csv) but this file is quite difficult to work on for Analytics and they want to be able to cross different datas.

On long term the idea is also to determine for exemple if we have 25% of our photo < 500px / 10% > 600 / 15% > 800 etc... ==> to exploit more precisely the datas

So Analytics needs to know the dimensions of the photos.

Therefore, we'll add 2 new columns in `MEETIC.PHOTO_ALBUM` (for portrait photos) and

`MEETIC.PHOTO_ALBUM_PRIVATE` (for album photo) :

- **PHO\_WIDTH**
- **PHO\_HEIGHT**

So insert the width & height in these columns when saving a photo ==> only when bigger photo is activated.

### 3.7.1.2 Add upload time on DB

Moreover, we would like to log the upload time in DB - so we'll create a new column (type FLOAT):

- **PHO\_UPLOADTIME** (in `MEETIC.PHOTO_ALBUM` & `MEETIC.PHOTO_ALBUM_PRIVATE`)

We want to log the time from the moment the user clicks on the button "Save" till the photo is really saved on the server // from a functional point of view= the time the user has the loader.



➔ The data should be saved in seconds with 3 decimals please.

We'll have to do that 1st of all on the current new component in Flash+JS.

### 3.7.1.3 Add log when user meets an error

We need to know if the constraints we've fixed today for the upload photo are realistic or not.

What's the % of our users that meets an error during the upload ? etc...

Today we display an error message to the user if he uploads a photo:

- smaller than 105x105px
- bigger that 5MB

Therefore, we need to add a new log where we'll increment a counter each time a user meets one of these errors - on the 2 versions of the uploader: new uploader component and bigger photo activated ( // Not the old component in flash)

Format could be:

- Photo < 105x105px: xxxxxx
- Photo > 5MB: yyyy

We don't need the abo\_id, pho\_id or else in this log. A counter is enough.

➔ The name of the log should be **photo\_size\_limit.log** available in `mcorp/data/log/meetic`

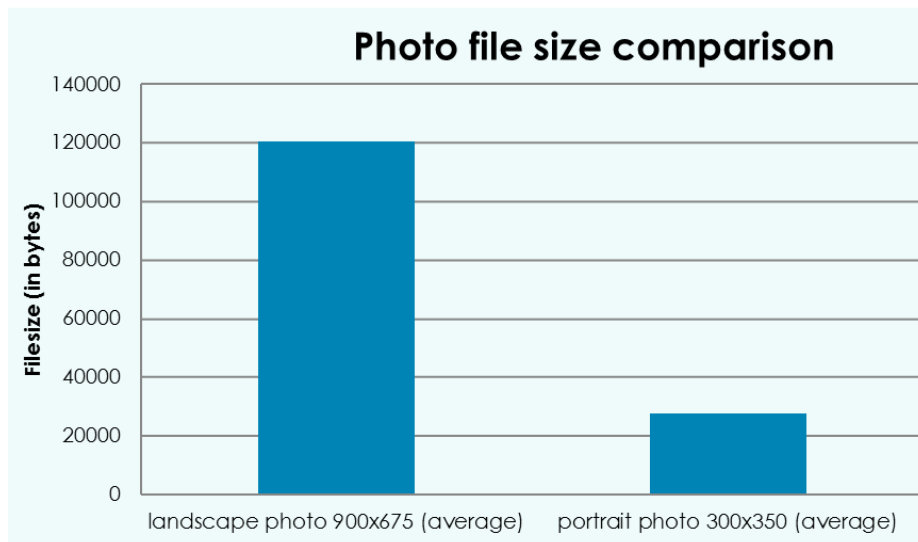
➤ **Action:** MR#1 – Tudor's Team

## 3.8 System infrastructure impacts

### 3.8.1 File size comparison

Core Team (*François Pezier*) has studied the impacts on uploading bigger photos, here are the results of the analysis:

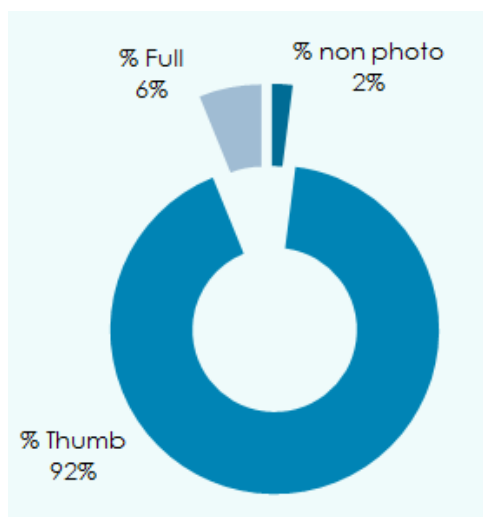
- Today the weight of a 300x350px photo is **27kb** in average
- With the new full size 900x675px, the weight is **120kb** in average = **444% bigger** than the current format.
- Between 900x675 and 300x350: **size is 4 times bigger = 6 times more pixels**



### 3.8.2 Thumb / Full requests repartition on dating

From an analysis done on picda.ilius.net, june 9<sup>th</sup> 2012 between 18h and 24h – we can observe that 92% of the Akamai traffic is on the thumbnails (94x94) whereas only 6% of the full size photos (300x350).

This is because we load 20 thumbnails on only 1 page (ex: the ons page) whereas we have that users mainly browse the lists and don't load the profile page a lot to zoom on the photos.



NB : 2% non photo requests is crossdomain.xml

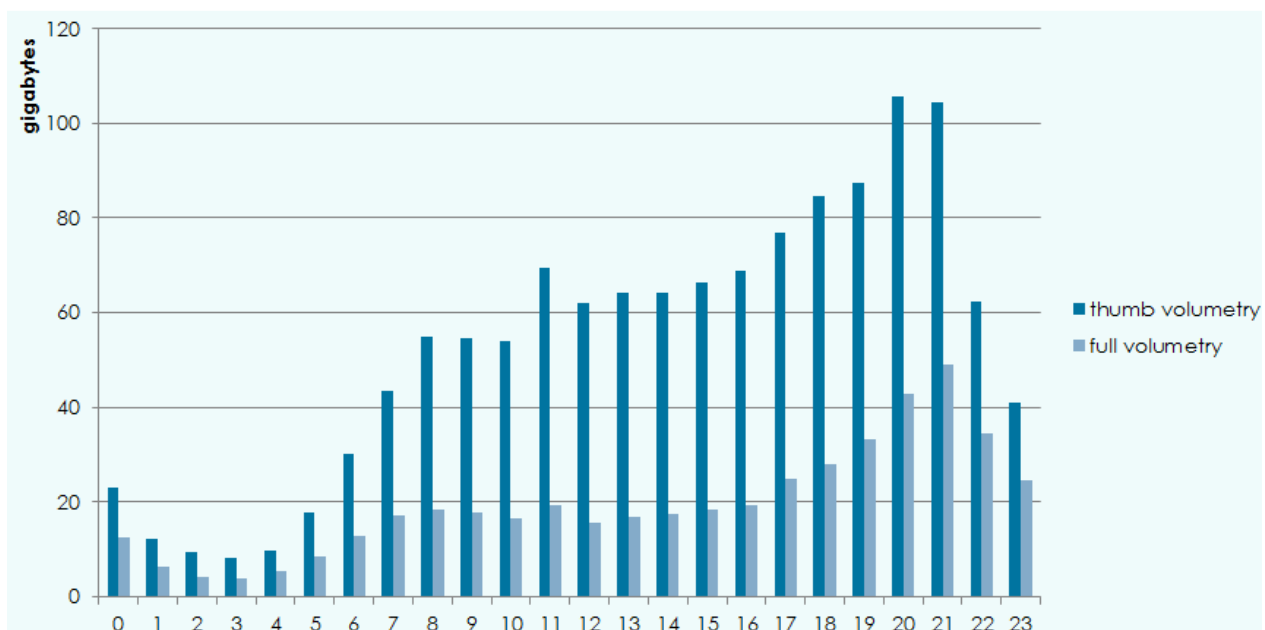
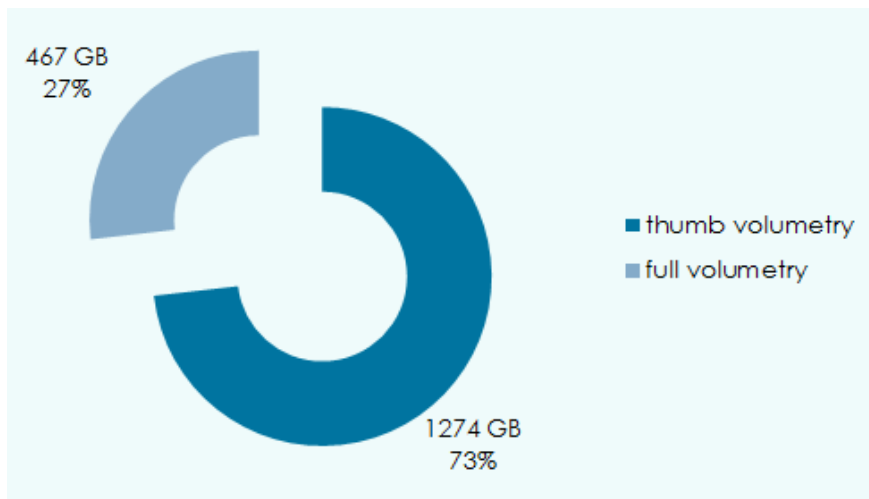


### 3.8.3 Thumb / Full : volumetry repartition on dating

On picda.iliius.net, Monday the 11<sup>th</sup> June 2012 between 0h and 23h59

On one day :

- Thumbs represent 1274 GB
- Full photos represent 467 GB



➤ In charge of the analysis : François PEZIER

### 3.8.4 Photo server

For this Sub1 we are going to enable the upload of bigger photos but we don't want to impact the display on site. So we will store big photos on DB but we'll keep on displaying photos 300x350px on site.

As the photo displayed on site are provided by the photo server, we will add an "operation" on the photo server to force the size of the photo displayed.

This "operation" is not a resize but a scaling of the photo.

- For photos > 300x350 → the server will scale the photo to reach 300x350px max (keeping the ratio)
- For photos < 300x350 → the server does nothing – the photo is displayed as it is on stored DB.

It will impact only the big photos uploaded, not the former ones.

We consider an 'operation' on the photo server as a 'live modification' of the photo stored in DB.

It changes directly the display but the photo on DB keeps the same specifications.

For example the photo server can:

- Blur
- Add round corners
- Resize
- Scale
- Crop
- ...

The uploaded photo saved in DB is not replaced by the one 'live modified' by the server – only the URL of the photo changes on site.

For info, the url of a photo is composed of:

- Host + prefix + Encrypted data
- And encrypted data = Picture ID #Source #Operations (scale, round...)

So we'll just change the url of the photo for the display – adding the operation Scale.

⇒ Please refer to the doc photo system for more details [\[DOC\]Photo system.pptx](#)

But adding operations on a photo has an impact on the CPU consumption because the query will be a bit longer. Therefore, to measure the impact Raphaël Moreau has added the operation SCALE on only 1 photo server first (we have 2 photo servers).

Hopefully we have observed that the impact is minimal so Raphaël has added the scale on our 2 servers on production (+ all our environments dev/pp/r7).

⇒ Today (03/08/2012) all photos above 300x350px are rescaled automatically by our photo server.

Scaling the photo from the server is the only way to enable the upload of bigger photos without impacting the display on site (keeping 300x350px on site).

➤ **In charge :** Raphaël MOREAU for all actions on the photo server

### 3.8.5 Measure the impacts on the platform

Today Spain's traffic is around 10% of our whole traffic so we guess this won't have a significant impact on the different graphs we follow.

(For info we have the stats for each server on <http://stats.ilius.net./munin/photostc/photostc1/index.html> - here it's Photo server #1 // we have 2 servers).

But we need to know exactly how much Bigger photo costs to define the number of new servers / disk etc... we should buy.

This 1<sup>st</sup> test will determine our strategy regarding the prod platform, and consequently the rollout of Bigger photo. Moreover, for Analytics purpose we'll need to know how many users add photos bigger than the current format (300x350px). Depending on the result we'll plan the launch of the display - because we can display bigger photos only if we have enough datas on DB.

#### 3.8.5.1 Add logs

Since we don't flag photos in DB as a "big one" we are not able to know the weight of bigger photo on the disks. So to know exactly this weight we need to add logs – and we'll check with the DBA the weight of bigger photo every day.

- Add logs in a CSV format – 1 file per day with this format:  
`/mcorp/data/logs/log_biggerphoto_YYYYMMDD.csv` (ex: `log_biggerphoto_20120820.csv`)
- Format : `[DATE - Hour]; PHO_ID; ABO_ID; {{SITE}}; {{FLAG}}; {{TYPE_PHOTO}}`
  - **SITE** : You can get this info from the config.inc of the site. We have 2 variables depending on the site so:
    - Take the `_PAYS_ID_SITE_REF` by default
    - If `_PAYS_ID_SITE_REF` doesn't exist, put the `_PAYS_ID_SITE_DEFAULT`
  - **FLAG** :
    - Flag = **0** if photo size < or = 300x350px (*=current format*)
    - Flag = **1** if photo size > 300x350px (*=Big photo*)
  - **TYPE\_PHOTO** :
    - **P** = if the photo uploaded is a photo portrait
    - **A** = if the photo uploaded is a photo album

➔ Add a new line in the log only when Bigger Photo is activated in the config.– we shouldn't add a line in the log when using old uploader component or new component but with size = 300x350.

➔ This will give us an idea of how many disks we'll need to add – considering that SP = 10% of our volume.

#### 3.8.5.2 Upload bigger photo in a dedicated handler

Today we have 4 or 5 handler php to isolate some features and don't impact the whole site if we have an issue on 1 feature. Example:

- We have dedicated handler for Search, cultural modules, Push profiles etc....
- So if search is KO it doesn't impact the whole site, only the searches.

To distinct the handlers we use different extensions = `.ph3`, `.phtml`, `.asp` etc...

- Today to upload a photo we use the common handler = `.php`

Therefore with bigger photo we know that it will take more time to upload a photo so if we have a big issue uploading a big photo => it will slow down all the others PHP and the entire site.

That's why we want to isolate the upload of bigger photo in a dedicated handler

➔ We've chosen the extension `.ASP`.

#### How to ?

- Create a new file `.ASP` (with the same name of the file that currently does the upload) => `myprofile\photo_manager.asp` (instead of `.php`)
- Add an include of the `.php` in this `.asp` (as an example you have : `search/myblacklist.phtml`)
- Define to use the `.php` by default and use the `.asp` only if bigger photo is activated.
  - We should continue to use the `.php` for the former upload component (in flash) + the new one without bigger photo
  - We should use the `.asp` only if we upload "Big" photo (so when bigger photo is activated)

## 4 [Sub2] Display bigger photo

---

### 4.1 Presentation & objective

---

On Sub1, the objective was to store in DB bigger photos without automatically resize it to 300x350px.

We now have these max sizes:

- Main photos= 506x675px
- Album photos= 900x675px

This step (sub1) was transparent for the user because we kept on displaying the photo in 300x350px max (resize by the photo server).

This step is necessary to control how our platform reacts and validate we don't have any issues.

So now for Step2 : the objective is to display photos as it is stored on DB → do not force the resize by the photo server anymore when click on the zoom photo.

### 4.2 Perimeter

---

The launch of Sub2 will depend on the nb of bigger photos in DB for each country → we want to have an important volume (To be defined by Product + Analytics) before launching this sub2 on production.

Today we don't know the % of click to display the layer zoom photo.

⇒ *This has been added to the SuperTracking project in order to define if this feature is often used or not, but we're still waiting for the figures.*

As we'll use a new URL to display bigger photo in the layer zoom, this could have a huge impact on Akamai servers = huge budget because we pay according to the volume / nb of requests.

Because changing the size of the photo displayed means changing the url of the photo and regenerate all the caches (we'll use a new type to display the bigger photo).

Therefore, while we don't have the figures from SuperTracking, we can't define the exact perimeter (*SP only? group of countries? all countries... ?*).

→ Nevertheless, in the meanwhile we would considerate that this Sub2 should be tested 1<sup>st</sup> on **MEETIC SP** (*no AB Test*)

## 4.3 Website

Profile page => layer photo when zoom on the photo → add loader before displaying the photo.

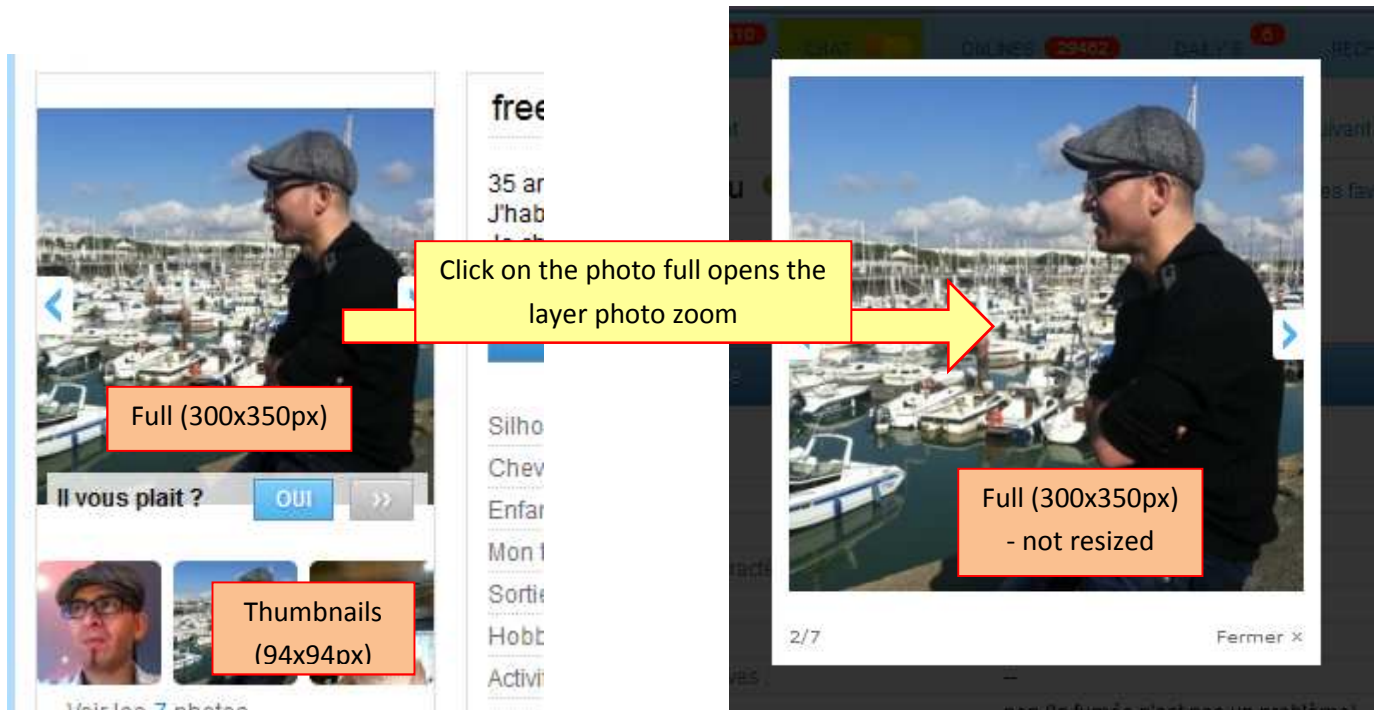
### 4.3.1 Mechanism

#### 4.3.1.1 Current behavior

Today on the profile page, 2 kinds of photos are loaded – but displayed smaller thanks to a html resize:

- **Full** (=300x350px)
- **Thumbnail** (=94x94px)

→ Due to Bigger photo Sub1, since we didn't want to impact the display, **the photo type Full is now always resized to 300x350px** max thanks to the photo server (even if it's a bigger photo that has been uploaded).



- When the user clicks on a Thumbnail, it displays the photo Full on the dedicated bloc – this means each time a user browse the thumbnail, we load the photo Full.
- When the user clicks on the photo Full → it opens a new layer “photo zoom” with the photo Full not resized in html this time.

#### 4.3.1.2 What we want

We need to force the size of the photo Full to 300x350px even if it's a bigger photo, otherwise we'll have to load the bigger sizes (*or as it is stored on DB*) each time a user clicks on the thumbnail.

→ And this is not what we want, for 2 reasons:

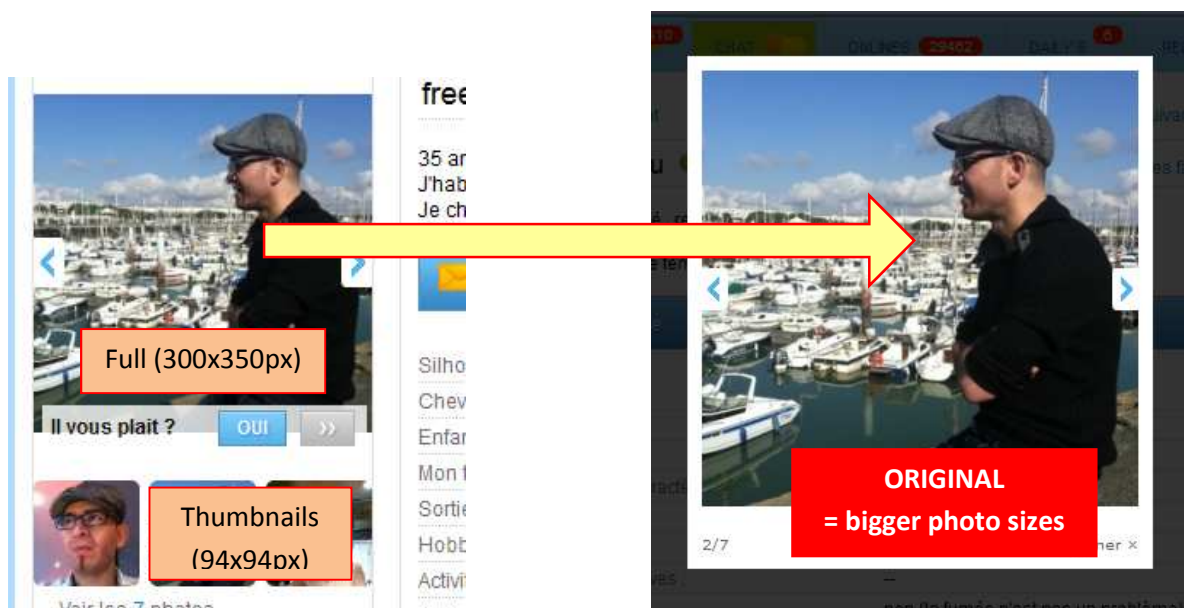
1. Display the bigger photo would mean change the url = huge impact Akamai = cost!
2. We don't want to load the biggest size & resize it in html => we should limit the loading on the page

→ We don't want to change the URL of the thumbnail & the photo full

Therefore we have to define a new type of photo - here's the new nomenclature:

- **Original** = original photo saved on DB – no resize = old size (300x350px) or bigger sizes (506x675px or 900x675px max)
- **Full** = photo original resized to 300x350px by the photo serve
- **Thumbnail** = 94x94px

So what we want on this Sub 2 is to keep loading the photo full + thumbnail on the profile page and ONLY when click on the photo full to zoom => display the "ORIGINAL" photo on this layer photo zoom (photo as it is stored on DB)



#### 4.3.2 Configuration

Since for the moment we don't know if the layer photo zoom is often clicked (volume ?) we would have to launch step by step the display of bigger photo, otherwise this would have an impact on Akamai server to invalidate the cache = impact on the budget because we pay according to the volume / nb of requests.

Therefore, we should find a way to activate the display of Bigger photo in the config of the country.

So for Meetic SP, we should define a new variable in:

```
\libs\product\sites\es\meetic\www\config.inc
```

Example of variable to activate bigger photo:

```
$siteConfiguration['GLOBAL']['DISPLAY_BIGGER_PHOTO'] = TRUE;
```

This means 2 behaviors:

- **Default:** for the countries where bigger photo is not activated (*nothing defined in the config*) → use the photo type **FULL** in the layer photo zoom
- **Display bigger photo active:** use the photo type **ORIGINAL** in the layer photo zoom (*and full + thumbnail in the page*)

### 4.3.3 Define a new type of photo - How To

#### 4.3.3.1 URL Photo

Here is an example of a photo URL:

<http://picda.ilius.net.dev/photo/eXJzW1BaWXVQXlhczAYFUwZHRzR.ZXtQdDMaSwZIQEFAQI9geQoVRnt1aA--/>

This URL contains an encrypted part which once decrypted is formatted like this:

eXJzW1BaWXVQXlhczAYFUwZHRzR.ZXtQdDMaSwZIQEFAQI9geQoVRnt1aA--/

152045891341477534#DFPM#CR(351,233,532,532)

Data are separated with a #.

- **152045891341477534** is the Photo ID
- **DFPM** stands for **D**ating **F**ull **P**ortrait **M**ale (tells the photo server that it's a dating photo full size (400x300) from the Portrait Album of a Male).
- **CR(351,233,532,532)** is an operation that will be processed by the photo server to crop the picture with the coordinates provided (optional).

Here is the list of the main operations available:

Type	Description
SZ	Resize
SC	Scale
RC	Round Corner
BL	Blur
CR	Crop
RO	Rotate

#### 4.3.3.2 How to generate a photo URL

First of all, you'll have to add new constants for the new size by adding two entries in this file:

config/photos.conf.inc

Here is a sample of the file:

➔ Portrait photos / public photos

```
Define('PHOTO_TYPE_FULL',0); // Taille originale
Define('PHOTO_TYPE_LITTLE',3); // Taille 44
Define('PHOTO_TYPE_BIG',4); // Taille 94
// New constant for public pictures here... Should be something like
Define('PHOTO_TYPE_ORIGINAL', 5);
```

➔ Album photos / private photos

```
Define('PHOTO_PRIVATE_TYPE_FULL',100); // Taille originale
Define('PHOTO_PRIVATE_TYPE_LITTLE',103); // Taille 44
Define('PHOTO_PRIVATE_TYPE_BIG',104); // Taille 94
// New constant for private pictures here... Should be something like
Define('PHOTO_PRIVATE_TYPE_ORIGINAL', 105);
```

The dating websites uses the function `photo_getURL()` located in `/mcorp/www/meetic/libs/product/photos.inc` to generate the URL of a picture.

This function needs 3 parameters:

- `AbolD`
- `PhotoID`
- `Type` // one of the constant defined in the `photos.conf.inc`

In this function you will find this array used to specified the data to generate the picture URL :

```
$infos = array(
    PHOTO_TYPE_FULL => array('#DFP%s' , 'FULL'),
    PHOTO_PRIVATE_TYPE_FULL => array('#DFA%s' , 'FULL'),
    PHOTO_TYPE_ORIGINAL => array('#DOP%s' , 'FULL'),
    // Here we need to add two new entries so we can use the constants previously defined
    // in the photos.conf.inc file
    PHOTO_PRIVATE_TYPE_ORIGINAL => array('#DOA%s' , 'FULL'),

    PHOTO_TYPE_NOTVALIDATED_FULL => array('#DFP%s#NV' , 'FULL'),
    PHOTO_PRIVATE_TYPE_NOTVALIDATED_FULL => array('#DFA%s#NV' , 'FULL'),

    PHOTO_PRIVATE_TYPE_LITTLE =>
array('#DTA%s#SC(94,94,F2F2F2)#RC(8)' , 'THUMB'),
    PHOTO_PRIVATE_TYPE_BIG =>
array('#DTA%s#SC(94,94,F2F2F2)#RC(8)' , 'THUMB'),
    PHOTO_TYPE_THUMB_BLURRED =>
array('#DTP%s#SZ(94,94)#BL(0,9)#RC(8)' , 'THUMB'),
    PHOTO_TYPE_LITTLE =>
array('#DTP%s#SZ(94,94)#RC(8)' , 'THUMB'),
    PHOTO_TYPE_BIG =>
array('#DTP%s#SZ(94,94)#RC(8)' , 'THUMB'),
    PHOTO_TYPE_SPOT_MSN_60 =>
array('#DTP%s#SZ(60,60)#RC(8)' , 'THUMB'),
    PHOTO_TYPE_SPOT_MSN_75 =>
array('#DTP%s#SZ(75,75)#RC(8)' , 'THUMB'),
);
```

The members/index uses the function `member_infos_displayPhotos()` located in `libs/display/member/infos.inc` to get all the pictures.

In this function, we first get all the photos ID for the portrait pictures and loop through the results to get the photo URL by calling the function `photo_get()`. The same thing is done for the album pictures.

For both album and portrait photos we will have to add a new data into the arrays that will be the URL for the Original photo.

Here is a modification sample for portrait pictures :

```
$yatsVars['member_main_photo_src_big'][$j] = photo_get(0, PHOTO_TYPE_BIG);
$yatsVars['member_photo_full'][$j] = photo_get(0, PHOTO_TYPE_FULL);
$yatsVars['member_photo_original'][$j] = photo_get(0, PHOTO_TYPE_ORIGINAL);
// code to add
```

➔ This new data 'member\_photo\_original' will only be needed when zooming on a picture from the members/index page.



## 4.4 OBO

### 4.4.1 Display in OBO

We've already managed on Sub1 – [OBO – Photo display \(3.5.2\)](#) the display of the photos in OBO as we didn't want to change the tool and impact the CC.

Therefore, all the photos (even the big ones) are resized to fit the moderation pages & popup. This way, it doesn't affect CC's work.

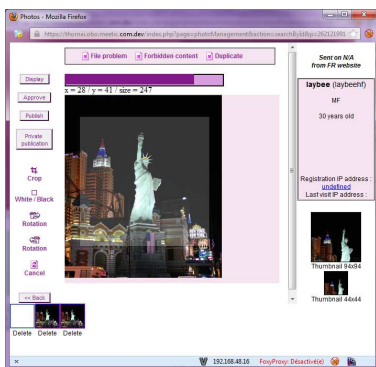
➔ So we don't have anything to change regarding the display of big photos in OBO for this Sub2.

### 4.4.2 Change thumbnails generation mechanism for album photos

As defined on the general overview ([2.3](#)) the thumbnails are generated on 2 different ways:

- **Portrait photo:** CC generates manually the thumbnail displayed on site – to center the user's face.

This step is done photo by photo.



- **Album photos:** the thumbnail 94x94 is automatically generated by a OBO script when photo is approved.

These photos are moderated in a list – 10 photos at the same time.



For the portrait photos the CC generates a nice thumbnail centered in the middle of the photo, a square without margins (a much as possible) = 94x94px. Whereas as we can see in the example below, for the album photo, the thumbnail generated automatically is just a rescale of the original photo, adding white margins (=94x94px).

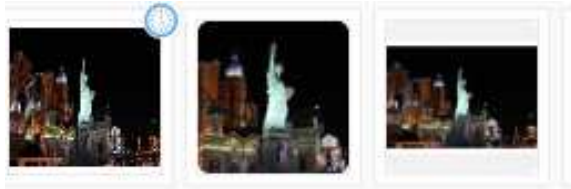
Therefore, now that we'll have landscape photos for the album, most of the album thumbnails will be generated with margins and won't even fit the whole square:

Original photo uploaded	Thumbnail generated in the album cue	Thumbnail generated in the portrait cue

We can see what it looks like in the photo manager layer- this is not nice:

1. While photo is not validated => the photo saved in DB is displayed (full size) & stretched to fit the square *(we consider this is not an issue since photos are moderated very very quickly, this case happens only during a few minutes)*
2. Thumbnail has been generated manually - in the moderation popup => this is nice.
3. Thumbnail has been generated automatically => it doesn't fit the square (add margins), this is not nice.

#### → Album photo



So as we can see there, with the same original photo and depending on how / when it's moderated, the thumbnail is completely different and we want to avoid this display.

Indeed, with the "Bigger photo" project all the album photos will be in landscape format so all the thumbnails generated will have white margins & won't fit the whole square if we don't change anything.



[[ We don't have any issue for portrait photo since all portraits are moderated manually, so is the thumbnail => we will always have nice "full" square thumbnail

1. Thumbnail is generated manually so that's ok – it looks nice.
2. While photo is not validated => the photo full is displayed & stretched to fit the square *(not an issue)*

#### → Portrait photo



→ The objective then is to change the way we generate automatic thumbnail for album photos and make it cleverer.

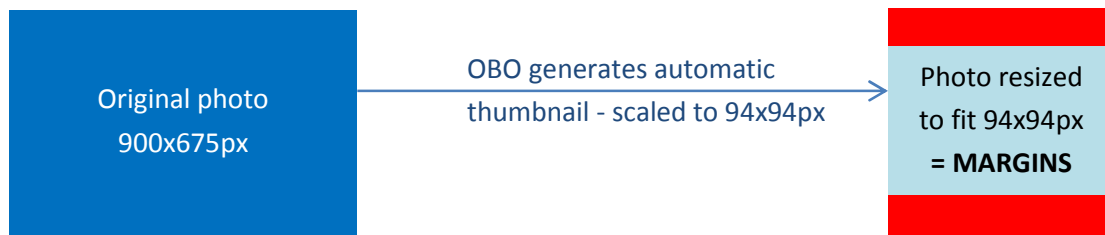
So for the moderation page where we validate the album photos 10 by 10 and where we generate automatically the thumbnail 94x94 we want to improve the mechanism in order to avoid margins and to have a photo that fills the entire square as defined below:

// Moderation page where automatic generation mechanism should be changed

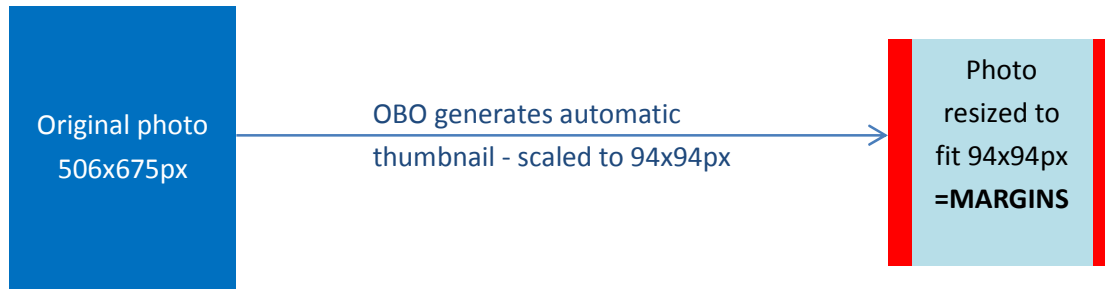


## Current behavior:

### Landscape format:

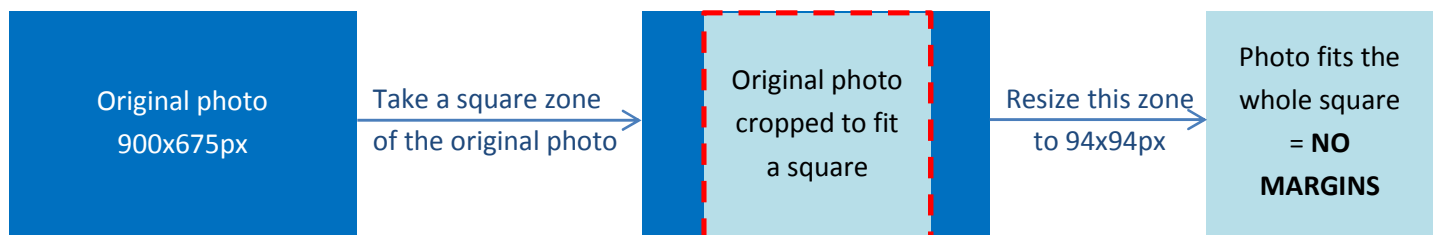


### Portrait format *(this case is more acceptable than the landscape format because fits the height - nicer)*

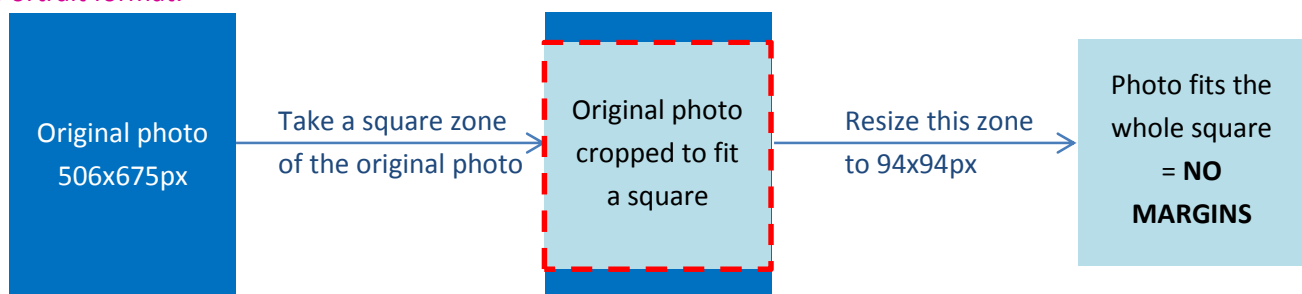


## What we would like:

### Landscape format:



### Portrait format:



➔ This way, all the generated album thumbnails will be nice and visible *(except for cases where original format has very specific size like panorama etc...)* and we'll avoid that kind of display when we visit a profile:

- 1st line = portrait photo (= portrait format + validated manually)
- 2<sup>nd</sup> line = album photos (= landscape format + generated automatically)



➔ Action: MR#1 – Tudor's Team // **ALREADY DELIVERED WITH SUB1**

## 4.5 Mobile

---

Since Sub1 the photos are already rescaled to fit the width of each mobile screen.

Mobile's team has checked that the display with bigger photos is not broken on the 3 environments (wap, app, touch) and everything is ok.

➔ So nothing special has to be done for this Sub2.

If we want to display bigger photos on mobile (*fullscreen for example*) this is a different project that must be added to the mobile roadmap.

- Amelie Billoud would write the functional specifications
- Sylvain Guarrigues's Team will develop it

➔ None specific mobile development will be delivered with Sub2.

## 4.6 Infrastructure impacts

---

The main impact of this Sub2 on the infrastructure would be on the Akamai servers.

As explained before we should anticipate the invalidation of the Akamai caches before we display bigger photos.

Because changing the size of the photo displayed means changing the url of the photo and regenerate all the caches

➔ because we'll use a new type to display the bigger photo

On the profile page for the moment we load directly the photo full ➔ so we don't need to change the URL of the photo displayed on the profile page (on the left column) ➔ we would generate new URL only for the "Original" photo (=900x675px)

- Today we have 2 formats:
  - Thumbnail
  - Full
- With this Sub2 we'll have:
  - Thumbnail
  - Full
  - Original = New Type to be delivered by the photo server

As specified before, we would schedule a launch step by step to control the impact on the Akamai servers each time. This is the only risk we have for this Sub 2.