Wacs Administration Guide

Fourth Edition

for WACS 0.8.6

B "Beaky" King

Publication date 30th July 2011

Wacs Administration Guide

by B "Beaky" King

for WACS 0.8.6

Publication date 30th July 2011

Abstract

WACS is a tool for building Adult Web Sites; it is equally suitable for managing a private collection or building a commercial web site. It has many best of breed features including dynamic filtering, model catalogs, automatic download and powerful search engine. It comes with a powerful API (application programming interface) implemented in both Perl and PHP5 languages to allow web developers to leverage it's facilities from their own programs.

This book describes the administrative tools used to manage, catalogue and update collections of digital media (images, videos, etc) maintained within a WACS web site. It provides both a tutorial and a reference document for the various administration applications within the WACS system. The intended audience is WACS web site managers and support staff tasked with maintaining a collection within a WACS system. Some familiarity with using WACS as a normal user is expected, carefully reading the User Guide should suffice.

The WACS source code and other documentation and support tools can all be found at the WACS website at Sourceforge [http://wacsip.sourceforge.net/] and at the WACS page on Launchpad [https://launchpad.net/wacs/]. A demonstration site using WACS is available at PinkMetallic.com [http://www.pinkmetallic.com] (site expected to go live January 2010). Commercial add-ons and support options can be purchased from Bevtec Communications Ltd, see their website at Bevtec Communications [http://www.bevteccom.co.uk/].

Table of Contents

List of Tables

List of Examples

Part I. WACS Administration Tutorial

This part of the WACS Administration Guide is designed to introduce you to managing collections of image sets and videos using the WACS tools. In order to work through the examples, you will almost certainly need access to a non-production WACS server and a reasonable collection of sample content. In the near future, the WACS developers hope to offer a set of sample content through our WACS demonstration site.

Chapter 1. Introduction

Overview

Welcome to WACS, Web-based Adult Content Server, a free software package for the management of material of an "Adult Nature" (or basically whatever euphermism for porn you prefer). It is web-based and can be used for the management of an existing collection, as a download manager, or as a back-end system for running a commercial adult web site. It is dramatically different from most other image gallery systems in that it understands photo sets and video clips as basic concepts, instead of single photographs. It also includes far more specialised tagging, source, relationship and attribute marking concepts than other more generalised systems. WACS' abilities in the areas of searching and dynamic filtering are really industry-leading in their power and flexibility.

WACS primarily consists of three components: the main WACS web-based user environment, the Application Programming Interface (API) and the collection management tools. The web-based user environment will probably be used directly when WACS is used as a private collection managment tool and as a back-office site maintenance tool when it's used commercially. Most commercial web sites will typically use the API to access the WACS infra-structure from their own custom web pages, although there is no reason that they couldn't offer the normal WACS user environment either as an alternative for power users or as their main environment. Whichever of these options you use, the administration tools create and manage the collection then offered via either of these content delivery methods.

About This Book

This electronic book, the WACS Administration Guide, describes the administrative tools used to manage, catalogue and update collections of digital media (images, videos, etc) maintained within a WACS web site. It provides both a tutorial and a reference document for the various administration applications within the WACS system. The intended audience is WACS web site managers and support staff tasked with maintaining a collection within a WACS system. Some familiarity with WACS at a user level would also be a distinct advantage, and we would strongly recommend working through the companion user guide first - who knows it might give you some ideas about neat extra features you can add to your own site. All documentation for WACS is available both within the distribution and from the WACS Web Site at Sourceforge.net [http://wacsip.sourceforge.net/].

It is important to stress that *ALL* of the collection management tools are implemented in Perl and the PHP interface is an optional addition to, not an alternative to, the core Wacs system which is perl based. Given the relative youth of the WACS system, php5 has been selected for the implementation to save future porting efforts as it is expected that php5 or later will be the minimum common standard by the time Wacs reaches 1.0. There is no intention to support older dialects of php at this point.

As the WACS software package is Open Source, we're always looking for contributions; if you create a site design (or prototype for one) which you don't end up using, maybe you would consider donating it to the repository of sample *WACS Skins*. We can always substitute our own artwork into already written web application code.

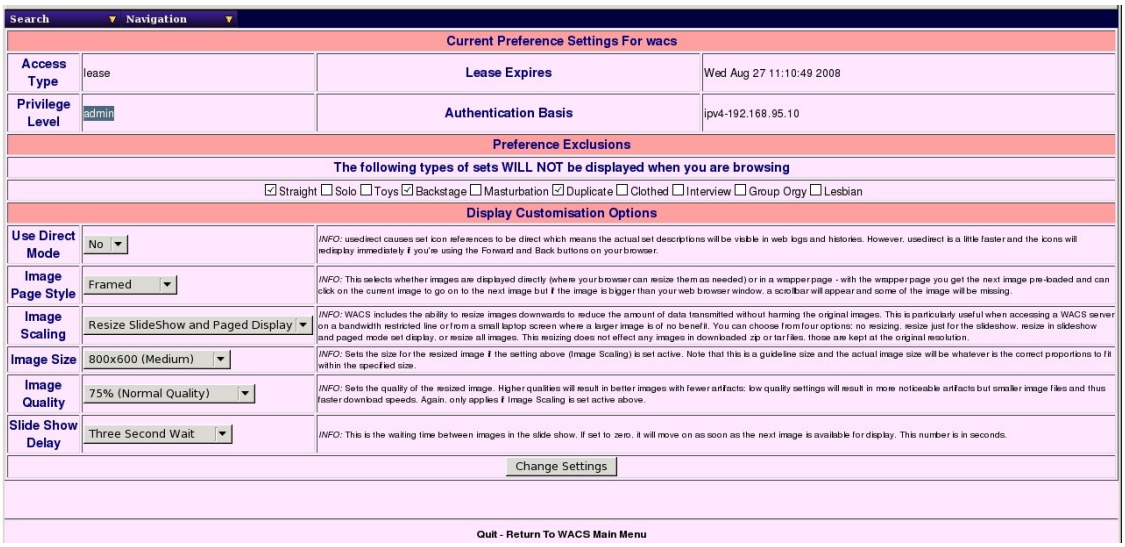
About The Examples

For copyright/licensing reasons, the example images feature sets from photoshoots by the main developer of WACS (Beaky) and a friend of his. These sets are now available for viewing and download on our demonstration site PinkMetallic.com [http://www.pinkmetallic.com/] which went live in January 2010. Initially this site will be free to use, but we may introduce a charge at a later date to help us fund the expansion of the content.

Chapter 2. First Steps

User Class

The WACS system has three basic classes of user: viewer, power, or admin. The viewer level of access is the default and provides all the normal functionality described in the WACS User Guide. However as a site manager or support team member, you need a little bit more access than that and the ability to actually make some changes. These are described as roles within the WACS system. You can check your current role by looking at the top part of the Preferences page (**wacspref**) - the screen shot below gives an example of this - the role field here is highlighted showing we're an administrator (admin).



The WACS Preferences Screen Showing Admin Role

So what exactly do these two enhanced roles allow you to do? Well, the role of power user allows you to view, modify and delete anyone's saved searches and to update the meta-data associated with any of the sets. It also shows you more of the mechanics underlying the WACS system - you can view download histories, download records and other aspects of how the system is behaving. It's not really about managing the collection so much as being able to offer help and advice to end users and knowing more about what is going on. To make significant changes, import and rename sets, change model details, etc requires full administrator rights - the majority of this guide will deal with those activities and the tools available to help you perform them.

If you've read the configuration guide, you will have seen that the main wacs configuration file (`wacs.cfg`) in it's security section includes two attributes related to these rights: adminusers and powerusers. This is a comma separated list of those user accounts that should be granted these rights on login; note that the rights are mutually exclusive and that an admin user automatically has all rights of a power user. Access rights to a given user account may be granted in one of two ways: by permanent access rights for the IP address, or by a lease based mechanism based upon logging in first. The Preferences screen above (**wacspref**) also shows you the basis upon which and the IP address to which your access is currently being granted.

Note

Permanent leases based on a fixed IP address and account combination will not be automatically updated. If you are using permanent leases, you will need to seperately modify

the role entry in the section for that IP address to read either `admin` or `power` to enable the admin modes

Changing Your Role (Leases)

If you logged in at the start of this WACS session, you will be using the lease-based security mechanism - this is the easiest type in which to change your account status to gain access to the administration tools. To make the change, you first need to log out of WACS; this can be done very simply by clicking on the logout link at the top right hand corner of the WACS main menu (aka front page). Once you've logged out you need to edit the main WACS configuration file, usually called `/etc/wacs.d/wacs.cfg` and find `adminusers`. This can be done with any text editor capable of working with XML or plain text files which you are familiar with. Once you've found the adminusers entry, simply add your account name to the list, separated from the other entries by a comma if necessary.

In this example we'll add johnd to an existing admin users list that includes the wacs account itself, and two other users called Karl D and Julie K (account names `karld` and `juliek` respectively).

Before:

```
<adminusers>wacs,karld,juliek</adminusers>
```

After:

```
<adminusers>wacs,karld,juliek,johnd</adminusers>
```

Note

In keeping with normal Unix practices we do not allow spaces in user names. Our accounts are of course often the same as the Unix account names; definitely so if we're using *Host Authentication* but still a good idea even if we're not.

Once the account name has been added to this file and the modified file saved, we simply log in again using the normal WACS login page and our user status will have changed. We can then proceed the next major section (the section called “Everything Looks Different”) on how to make use of our new found additional priviledges.

The actual current level of access is determined by the role field within the active leases file. If you want to grant extra priviledge for only the duration of the current lease, you can do this by directly editing the lease file. The default location of this file is `/var/run/wacs/leases.acl` and if you call this file up in an editor you just need to find the entry with the correct IP address and username and change the role in that entry from `viewer` to `admin`.

Changing Your Role (Permanent)

Here we have to deal with a slightly different concept. If you make the changes detailed above, whenever and from whereever you log in, your account will take on the elevated status it has been assigned. This is because whenever you login your right to elevated status is checked for in the configuration file and your access right record for that IP address in the leases file set accordingly. A permanent access right is based upon a particular permanent IP address and doesn't go through this assignment process. It is therefore completely possible for a given person's PC (so long as it has a fixed IP address) to be given administrator access but for that person not to have any elevated access when using their account from other system. In

some cases, this may be what you want, although it would preclude that person working from home, etc. Similarly you can also use this mechanism to offer automatic administrator priviledge to anyone logged in on the local system itself.

The permanent access rights are controlled by a file called `wacs.acl` which lives in the same directory as the main Wacs configuration file. This will typically be `/etc/wacs.d` , but might be elsewhere especially if multiple Wacs servers are running on the same web server. If the Wacs install has been done using the packaged versions (.rpm or .deb), a sample entry showing how to do this will be included but commented out in the installed default `wacs.acl` file. Once uncommented, this looks like this:

Example 2.1. Sample `wacs.acl` file giving admin rights to local users

```
<?xml version='1.0' standalone='yes'?>
<accesslist>
 <ipv4-127.0.0.1>
   <user>wacs</user>
   <type>permanent</type>
   <date></date>
   <role>admin</role>
   <prefexcl>D,B,C,F</prefexcl>
   <usedirect>yes</usedirect>
 </ipv4-127.0.0.1>
</accesslist>
```

Each section of the file is enclosed by a **<ipv4-*host_IP*>** and the entries within apply to that host address only. There is one magic entry, **<ipv4-all>** which can be added to disable the access control system entirely. Looking through the fields in each entry, the first one, *user*, gives the user name to attribute accesses from this host to. The second attribute, *type*, is either permanent or lease. If it is permanent, access will always be granted as the specified user to this IP address. If it is lease, access will be granted until the time/date given in this field (in unix seconds since epoch date format). The *role* field defines what level of access this user should be given to make modifications to the database.

In many cases where the site owner is not worried about security or who can access the Wacs system (providing they have access to the machine), uncommenting the entry for `ipv4-127.0.0.1` in the `wacs.cfg` is probably the easiest thing to do. Generally this lets the console user manage the system fully and requires everyone connecting in over the network to have to log in. However this mechanism is a little simple-minded and can be circumvented by someone with some knowledge and a genuine login account on the server.

You can also use the same mechanism to grant permanent access to any other IP address on the internet, including obviously certain machines on your local network. Shown below is a sample file that grants such access to two machines on the local network - in this example `johnd` is an administrator and `billw` is either telephone support or a salesman - he's only a power user!

Example 2.2. A Sample `wacs.acl` file for a local office network

```
<?xml version='1.0' standalone='yes'?>
<accesslist>
 <ipv4-192.168.1.15>
   <user>johnd</user>
   <type>permanent</type>
   <date></date>
   <role>admin</role>
   <prefexcl>C,F</prefexcl>
   <usedirect>yes</usedirect>
 </ipv4-192.168.1.15>

 <ipv4-192.168.1.17>
   <user>billw</user>
   <type>permanent</type>
   <date></date>
   <role>power</role>
   <prefexcl>D,B,C,F</prefexcl>
   <usedirect>no</usedirect>
 </ipv4-192.168.1.17>
</accesslist>
```

Warning

Of course if you're using the IP addresses for this purpose you do need to make sure that they are statically allocated and that each person's PC is always assigned the same address. You wouldn't want billw getting administrator access just because he was the first person to switch on his PC one morning, now would you?

Everything Looks Different

Once you've sorted out getting your account to have the role of administrator or power user, you can return to the normal wacs system and explore as normal. You will however notice that a few things have changed: the screen shot below shows the thumbs version of the wacs model page as viewed by an administrator.

wacsmpthumbs as seen by an administrator

If you look at the bottom left corner of each set detail tile, you will see a heading *Update:* and below that two links: one of these is to **wacssetmgr** (described in Chapter 9, *Wacs Set Manager*) and the other is to **wacsinfomgr** (described in Chapter 10, *The Info Manager*). These two commands provide a convenient web interface to updating most of the information about sets stored in the WACS system. The first of these, **wacssetmgr** (see also wacssetmgr) is primarily concerned with attributes, ratings and the nature of the content of the set and so the link to it is entitled *rating*. The second of these, **wacsinfomgr** (see also wacsinfomgr) relates more to information about the set itself - it's location, source, special icons and if present a text description of it's content. This link is entitled *info*.

In addition to the changes to the model page, many other parts of the WACS system will also add extra functionality when your account has administrator status. The front page gains a whole additional *Maintenance* menu, set pages get additional options on the first menu, and so on. We will cover each of these as we cover additional topics in future chapters of this tutorial section of the WACS Administration Manual.

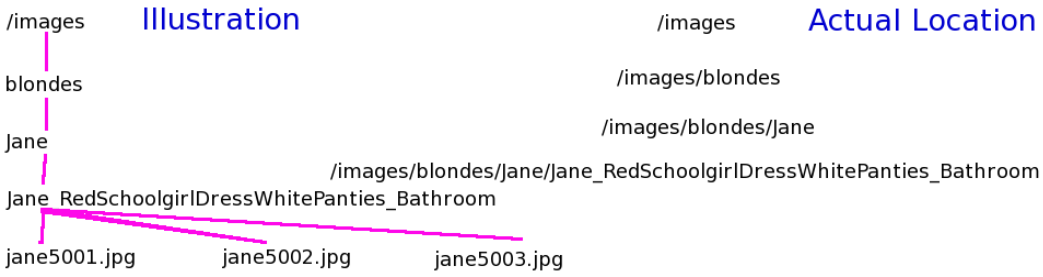
Chapter 3. Site Design

Overview: Laying Out Your WACS Site

As you know WACS is designed to allow for the management and presentation of a large collection of adult material, and as with any large collection there needs to be some level of organisation to how it is stored, if you are to be able to get the best results. Although there are many ways to find things within WACS, the actual underlying layout of the site is never obscured from the end user - it is part and parcel of the user experience. It is therefore important to pick a layout that works well for the type of content you intend to hold within the system.

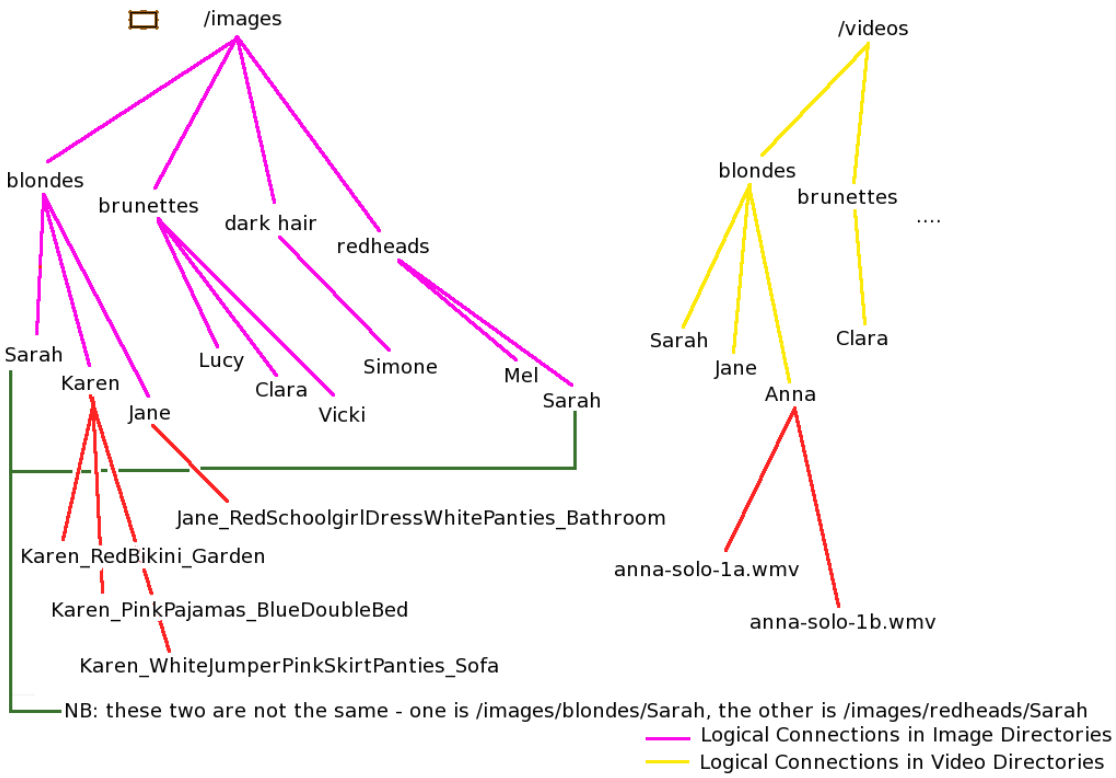
Within the WACS tools, you can pretty much use any layout you wish with directories nested up to arbitary depths - however WACS will naturally divide them into three sections: the toplevel, the middle bit and the final container. The toplevel directory, known as the *area* is stored in the field of the `sets` table of the database called **sarea** and so is usually known as `sarea`. The middle level directory or directories is known as the *category* and is again usually referred to by it's database field name of `scategory`. The final container is only actually a directory for image sets; for videos it is the actual file name of the video file - since WACS started as a purely image based system, this field is known as `sdirectory`. A number of the WACS tools include appropriate manipulations to split `scategory` into two distinct fields if used, but work fine if you simply leave the second half blank.

There is quite a lot of complexity to how the final component, the `sdirectory` is named which will be discussed in the next chapter, Chapter 4, *Naming Sets In WACS*. For now we're going to look at the various directory heirarchies that WACS can offer up as defaults and why you might wish to select one over another. You are free to use any other heirarchy you like, and we will discuss how that might be appropriate for sites with very specific themes.



In illustrating how the directory tree is laid out, we will use a tree diagramme in the style given above. In the above example, the toplevel tree of the Wacs document area is called `/images`, but may in fact be `/home/wacs/images` or any other similar pathname that makes sense on your system. Just make sure the correct path is used in the configuration file too. The next level directory is called `blondes`, thus making it `.../images/blondes` and so on down. Note that we will not normally show the actual image files in the subsequent examples that follows, but they should be assumed to be present at the lowest level. Note also that the absence of spaces in the filename, the presence of underscores and the exact capitalisation are both important and significant in WACS.

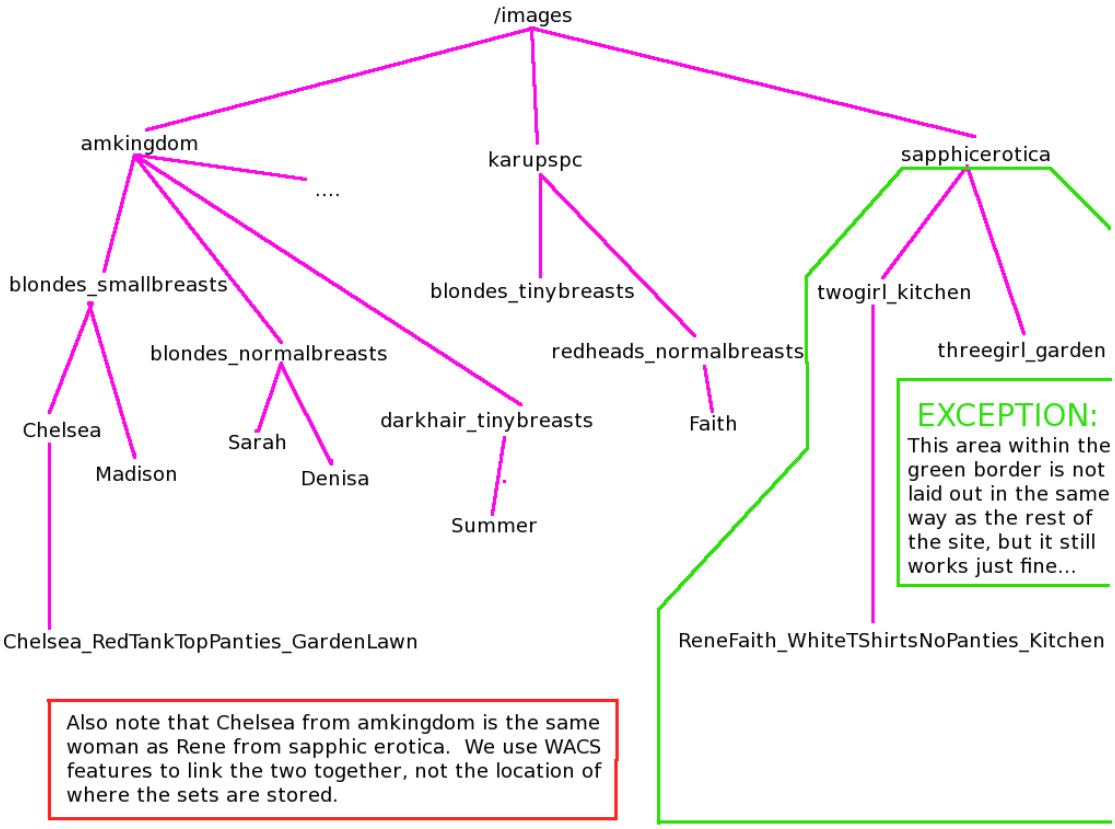
Simplistic Layout



One of the most simplistic layouts is to simply divide up the models by one key attribute and then by name, as shown in the diagram above. In this case we've choosen to use hair colour, which will probably work reasonably well for most sites featuring Caucasian (white) models and expecting no more than 30 to 50 models in all. In this case the `sarea` will be things like: blonde, brunette, redhead, and dark hair. You'll probably find that the section for blondes is significantly bigger than that for redheads, but that shouldn't be a surprise. Of course, if you're running a site with a particular niche like Oriental or Goth models, you may well find that you need to use a different criteria for the sub-division.

Despite being simple, we'd not particularly recommend laying it out in this way - putting all the sets by a model in a directory named for her isn't really the best approach as you can always find those sets simply by looking at her model page. You also need to consider what you'll do if there are multiple models with the same name - We've seen some of the big sites with over a hundred models called either Maria or Jana!

Vendor Mode



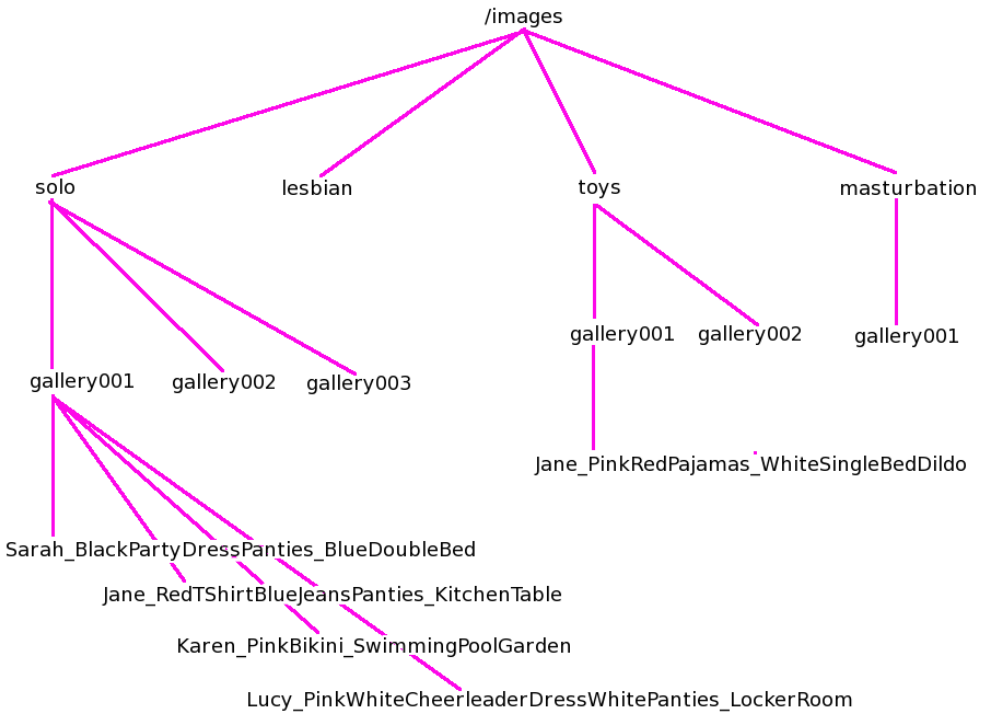
When Wacs was originally developed, it was being used to collect sets by selected models from a number of different subscription web sites and it seemed logical to organise the sets based upon the site from which they had been collected. This diagram represents that layout and this type of layout is still very much supported by setting the `layout->vendormode` attribute in the configuration file to `Y` and the `layout->style` attribute to `modelattr`. Additionally if using this layout, you may wish to use the modified menu configurations described in the AppNotes area.

The cornerstone of this layout structure is that the source site is the highest level grouping (the *sarea*). Underneath that we have combined the model's hair colour and breast size description into a single combined description. Generally this serves to divide each site's directory at this level into between 10 and 20 sub-directories, and even with a very large site (> 2000 models, 30,000 sets) this seems to remain reasonably manageable. Within each of these directories is a separate sub-directory for each model, and therein her set.

There are of course exceptions to this and over to the right hand side of the diagram, marked out by a green box, we have the sapphic erotica area. Since (almost) all their sets feature two or more models, focusing the organisation of material sourced from there on individual models makes much less sense, so in this case we use a completely different layout where we're grouping by number of girls in the set and location in which it happens. While the default system won't guess that much, it's absolutely OK to use a completely different organisation structure as illustrated and it'll all work fine. Notice also that whereas mostly in vendormode we're using two parts to *scategory*, namely the model attributes bit and then her name (as in `blondes_smallbreasts/Chelsea`), for the sapphic erotica section we're only using a single level sub-directory structure. So long as all of the image folders are at the same level within the sub-directory tree, this works absolutely fine.

In summary, vendormode is a good site architecture, particularly for collectors who are accumulating works featuring their favourite models in a private WACS server. It is not as appropriate for a commercial site or one that is based upon a less collection orientated activity, and that is where our third standard site design comes in....

Gallery Mode



The final layout that is supported in the defaults mechanisms within Wacs is a gallery layout of the type favoured by a significant number of the existing adult web sites on the internet. That is not to say that these are the only options, they are merely those which are supported by the defaults mechanism within the existing Wacs code. You can easily devise your own layout in most cases Wacs will support it providing a few basic premises are adhered to.

In the gallery layout, the toplevel organisation is by the type of content featured within the set; this means the toplevel will typically be things like: `toys`, `lesbian`, `straight`, `solo`, and `masturbation`. Essentially the top level *sarea* is the same as the standard set type category. The next level down simply consists of the word gallery followed by a three digit number within each top level category. As each new set is imported into the Wacs system, it is automatically placed in the gallery with the next available free slot within the appropriate top level category. This method gives the opportunity to view a number of similar sets from a range of models within the same container which seems to be a popular and accessible way of browsing collections. Of course all the sets can have links to the model pages of the models featured.

Of course all the latest sets tend to be in the highest number gallery for each category, and a number of the Wacs tools include configuration options to show the latest first and in some cases limit the number of gallerys shown for speed of display.

Summary

Wacs provides a lot of flexibility in it's support for site layouts but unless you have a particular need for a custom style of your own, you are probably best off using one of the standard styles outlined above. Consider carefully what you choose as it is definitely significant labour to change the layout at a later date, although tools that can do it are provided. Generally it is worth choosing either the vendor mode or gallery mode; the simplistic mode will only really work for sites where you are hosting solely your own content or have a very specific scope of interest which will result in only a very small number of models.

Please see the configuration manual chapter on layout to see how to make the selection of default values match your choosen layout. The configuration variables you will need to tune are `vendormode` and `style`. The next chapter will discuss how the actual container directories are named.

Chapter 4. Naming Sets In WACS

Goals In Naming Sets

Sets obviously have to be named or numbered in some way. If you want a system like Wacs to be able to search for things, it also needs to be given clues about what the set contains so it can add those details to the database so that there is some information there to be searched for. In Wacs, we use the set name as the way of kickstarting the process. In this chapter we will discuss how this is done and the constraints it places upon us in how we choose the names that we use. It is important to understand however that as with most other aspects of Wacs, this is just a recommended standard that will help streamline the experience. The Wacs system will use the hints provided in these naming conventions if they are followed, but will function OK if they are not. The only constraint is that the name should be a valid name for a directory on the server host file system and a strong recommendation to avoid the use of spaces in directory names - underscores (_) are much preferred and will be hidden from the end-user by way of being converted to spaces before display.

As you will have hopefully seen through reading the User Guide or through use of an existing Wacs system, it includes a significant number of search and selection features both through the *dynamic filtering* abilities of the model page and through the extensive *search* (aka tagging) system. Essentially you have two ways to provide this information to Wacs at present - the first is to enter it manually through the Chapter 9, *Wacs Set Manager* administrative interface, the second is to choose to use certain special words that will be picked up on when you name the set. The most effective results will usually be achieved through these two techniques being used in parallel. (There's actually a third way too, but that's really only for database Gurus...).

How It Works

Conventionally in Wacs we give sets pretty long names - generally all contemporary operating systems cope just fine with that idea - and those long names impart as much information about the set as we can possibly squeeze into them. What happens is that you describe the set, and that description is used for the directory name in which that set is placed. This is part of our policy to leave your content freely accessible and as usable as possible from outside Wacs as well as within; we *DO NOT* swallow your content. Our naming conventions are totally designed to make other methods of finding files benefit from the effort put into making them work under Wacs.

The standard Wacs format for set naming uses some basic conventions for how the names are assembled. There are considered to be three basic parts to a set name - these are:

1. The Model or Models Names
2. Her/Their Clothing
3. Location and Action Details

Each of these items are divided from the previous one by way of an underscore (_) character. Wacs will always replace the underscore with either a new line or a space; whatever best suits the way in which the set details are being displayed. Additionally Wacs uses a technique called Camel Text (so called because it has lots of humps in it) to further divide up the description to make it more readable. This is basically done by starting each distinct word with an upper case letter and having the rest of the word in lower case.

Camel-Style Text	***How It Will Be Displayed***
`OneTwoThreeFour`	`One Two Three Four`

Using these basic styles and constructs, we build up a name for the set that is simultaneously suitable for use as a directory name and as a description of it with a fair amount of detail. Putting it all together, we end up with a typical simple directory name looking something like:

```
Sarah_RedHalterDressStockingsNoPanties_GardenLawnTowel
```

Once we've selected to name a set in this way, a keyword matching process will be applied to that name. Looking at this particular example set name, we will hopefully find that we do know the model Sarah and so will be able import a number of details directly from her model record. Moving on the `NoPanties` phrase adds a set marking attribute called **nopanties**. The `Garden` keyword will cause two mark-ups to happen - a set marking attribute of **outdoors** will be added, as will a location attribute of **Garden**. The `Dress` keyword will cause the general clothing type to be marked as **Smart** as generally such dresses are. Of course if the keyword guesser gets it wrong, you can always correct it manually, but through slightly careful choice of the words we use, we've already determined no less than four things about the set.

Wacs will also add in the details about Sarah herself extracted from her model record, additional information about her such as whether she shaves her pubic hair, whether she has tattoos or piercings and whether she has unusual attributes like exceptionally small breasts or short hair will also be copied across to the set. Of course these change over time, and it is possible to correct these settings manually - we will discuss this later in Chapter 9, *Wacs Set Manager*. The net effect of all this is that we can now search for things like "Show me all the sets with girls with shaven pubic hair and no piercings in smart clothing but no panties in the garden."

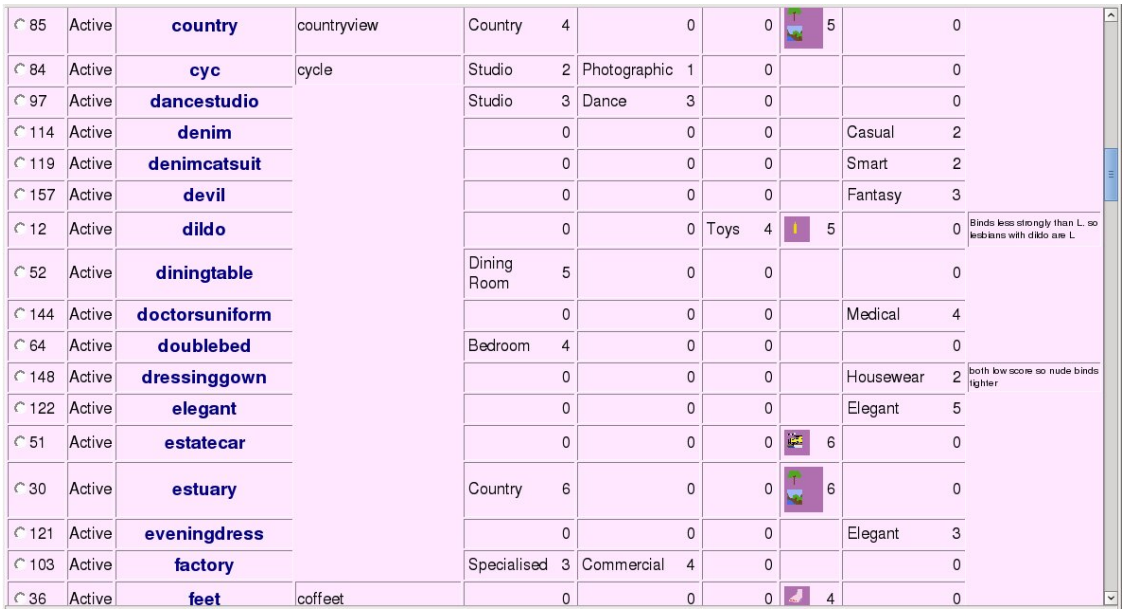
Hopefully you will appreciate how much information we've managed to extract from relatively little work on your part by way of carefully selecting the words you choose to use to describe the set. It will obviously take a little bit of time to get familiar with how the keywording system works and what the keywords are. Over the next chapter or so, we'll hope to provide an introduction to the most common keywords and the effects that they have. We'll also take a look at the keyword maintenance system itself and in particular how you can add any additional keywords that you like to use into the list of what the Wacs system makes use of.

Keyword Scoring

You're probably thinking that surely it can't be that easy and that a keyword scoring system like this is bound to make mistakes. Yes, it is and it does BUT in our experience such mistakes happen rather less than 5% of the time, which means that only maybe one in twenty sets needs manual attention to correct erroneous attributes applied to the set. We often add more, which is reasonably easy to do, but the fundamental point is that the keywording system works quickly and efficiently to build up a surprisingly comprehensive amount of structured information about the set. The level of accuracy is enhanced by a system of keyword scoring.

Each keyword the Wacs system knows about has a record in the **keywords** database, and for each of the possible types of attribute that the Wacs system might derive from it is both a value and a score. Thus if Wacs sees the keyword `Garden`, it notes a 7/10 score for a location of **Garden**, and a 5/10 score for a set attribute of **outdoors**. This means that it's really pretty sure the set is located in a garden, and thinks it's odds-on that it's an outdoor set. There's actually also an explicit exclusion of the phrase `IndoorGarden`, so that the normal attributes added to anything called `Garden` aren't added in the case of `IndoorGarden`.

At the other end of the spectrum, there are some words that have a very light "it just might be" level of binding. An example of this is the word `Denim` which gives a score of only 2/10 for it's assertion that this is a casual clothing style. Similarly the word armchair gives a very uncertain binding of just 1/10 that it might be in a lounge - any other location keyword like bed, dining table or kitchen will immediately override that lounge idea. The Wacs developers have actually done quite a lot of work on making the words bind with sensible strengths for the many sets in our testing database and generally the guesses are a lot more accurate than you might expect.



85	Active	country	countryview	Country	4	0	0		5	0	
84	Active	cyc	cycle	Studio	2	Photographic 1	0			0	
97	Active	dancestudio		Studio	3	Dance 3	0			0	
114	Active	denim			0	0	0			Casual 2	
119	Active	denimcatsuit			0	0	0			Smart 2	
157	Active	devil			0	0	0			Fantasy 3	
12	Active	dildo			0	0	Toys 4		5	0	Binds less strongly than L, so lesbians with dildo are L
52	Active	diningtable		Dining Room	5	0	0			0	
144	Active	doctorsuniform			0	0	0			Medical 4	
64	Active	doublebed		Bedroom	4	0	0			0	
148	Active	dressinggown			0	0	0			Housewear 2	both low score so nude binds tighter
122	Active	elegant			0	0	0			Elegant 5	
51	Active	estatecar			0	0	0		6	0	
30	Active	estuary		Country	6	0	0		6	0	
121	Active	eveningdress			0	0	0			Elegant 3	
103	Active	factory		Specialised	3	Commercial 4	0			0	
36	Active	feet	coffeet		0	0	0		4	0	

An example from Wacs Keyword Manager

If you take a carefull look at the above example which is a screenshot taken from the the section called “Keyword Manager”, you will see at the top of the list that the keyword `Country`, so long as it is not merely part of `CountryView`, will cause a score of 4/10 for a location of **Country** and a score of 5/10 for the attributes of **country** and **outdoors**. This is so that the phrase `CountryGarden` will not overrule the location of **Garden** but will also add the **country** attribute mark. The two resulting attributes, **country** and **outdoors** are represented here by their respective icons. You can check the exact attribute words used by editing the entry, but that is a topic for the section called “Keyword Manager” rather than what we have here. The main thing to consider at this point is to have some understanding of how the mechanism works and to be aware of the **wacskeywordmgr** and how to use it to browse the preloaded keywords and what they mean.

However before we leave this introductory chapter on keyword scoring we'll just take a quick look at some of the most common keywords that we are likely to want to use frequently.

Heavily Used Keywords

While we've just taken a look at the mechanism by which the keywords work and how we can examine the definitions and so on, that probably doesn't make you feel ready to use them right away. So we'll take a quick look at some of the most common keywords, what they mean and how they are represented so you can start making use of them as we progress to set unpacking and placement techniques.

Let's look initially at some of the clothing keywords that result in both clothing style and set attribute markups. Note in particular that we've decided that we want to sub-divide uniforms into multiple categories such as medical, hospitality, military and labourer rather than have a generic uniform clothing type. We could easily add a low-priority binding of uniforms to be of a simple clothing type of **Uniform** if all other uniform keywords fail to match. In fact, that sounds like a perfect task to do with the keyword manager when we're reviewing that in a later chapter (the section called “Keyword Manager”).

Table 4.1. Clothing Related Keywords

Keyword	*Clothing Type*	*Attribute Name*	*Attribute Icon*
`NoPanties`		**nopanties**	
`SeeThru`		**seethru**	
`Uniform`		**uniform**	
`Schoolgirl`	Schoolwear	**schoolgirl**	
`Cheerleader`	Schoolwear	**cheerleader**	

Even marking up using just these five icons will actually make quite a difference to what you can and can't search for within your Wacs managed collection. Of course these icons are only a very small proportion of the story; there are at least two other major categories which only affect the location and action section of our naming convention. These are those that show an action and those that show a location.

Starting with an action, we have a number of keywords that help define the type of the set: the keyword `FUCK` indicates a straight sex set, as does the keyword `Blowjob`, while other words like `Dildo` typically indicate a toys set. Do note however that once again the scoring system comes into play and a set that is marked as lesbian and includes the word `dildo` will remain marked as a lesbian set because of the higher score although it'll gain a the **dildo** attribute as well. Other commonly used keywords for the actions include `Anal` for anal sex, `Fisting` and `Piss` for those respective actions.

There are additional attributes for that third part of the name which describe locations rather than actions, these include `Garden` which implies **outdoors**; `Beach`, `River`, `Lake` and `Sea` which imply **outdoors** and **country**.

Hopefully these examples give you somewhere to start when naming your set. Do remember that use of none of this is mandatory - calling something simply "set23" or "Sarah_Set23" will work just fine. However we feel it would be a shame not to use this feature to help you get started with the mark-up process. Do also remember that you can easily add more attributes not covered by the keyword system using the rating facility aka Chapter 9, *Wacs Set Manager*. We will return to this topic in later chapters.

Chapter 5. Preparing To Create A Model Record

Introduction

One of the really major aspects of Wacs is it's ability to access sets via a range of methods, one of the most important of which is to be able to look at the work of a specific model. When we're adding new sets, we want to be able to add that set to a model's portfolio and in order to do this, the model has to already exist within the Wacs system. This chapter will outline how we prepare the necessary bits to create a model for the first time. If you want to look at an existing model record for reference, there are currently two model pages provided as XML files within the samples directory of the Wacs distribution. These can be imported into the Wacs system using either the model manager itself (new feature in Wacs 0.8.3) or using the command line tool **wacsimport** command which is described in Chapter 12, *Migration Tools*.

Headshot Image Preparation

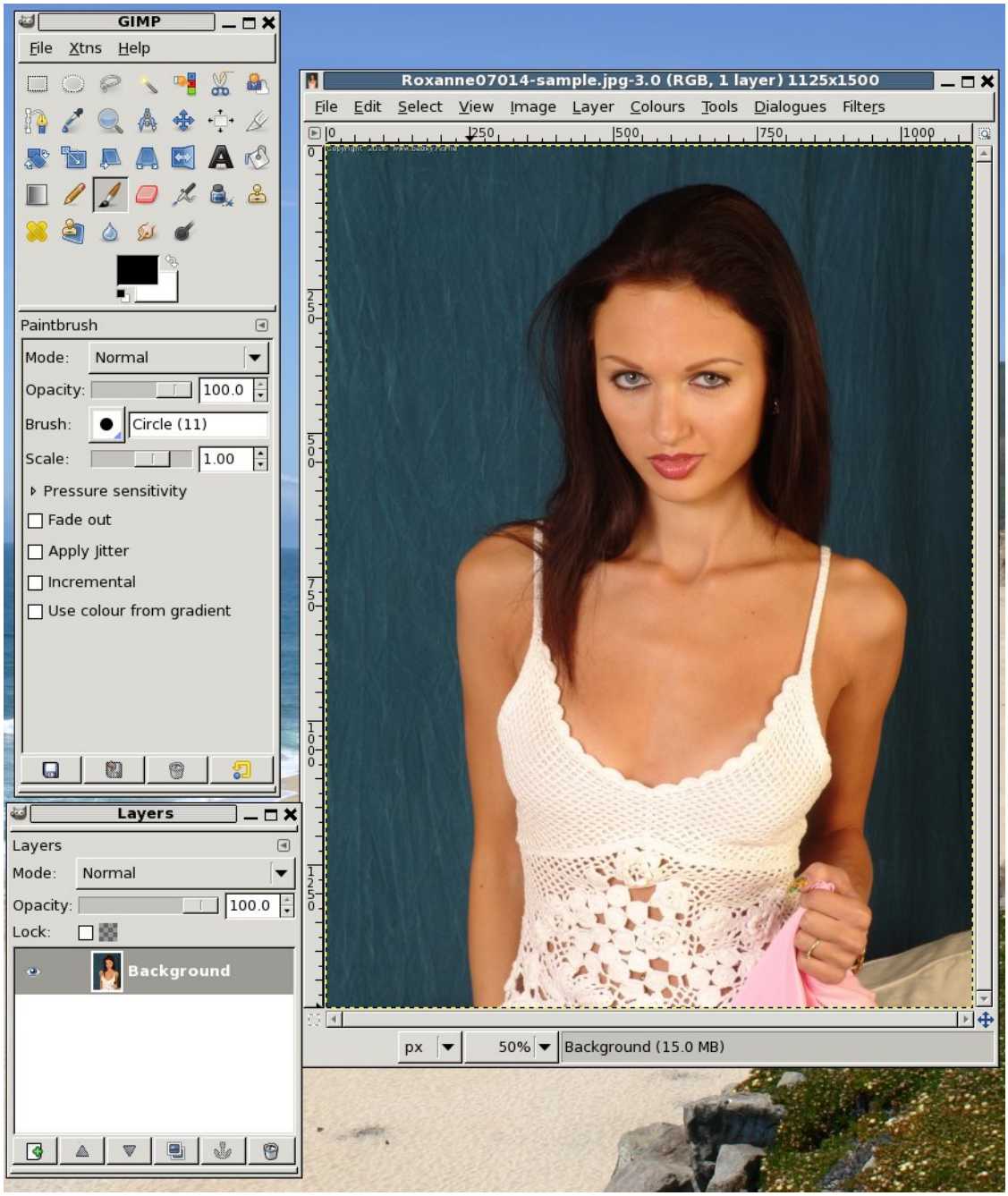
As you will have hopefully seen from either using an existing Wacs system, or by reading the User Manual, we do make very heavy use of the model headshot icons throughout the Wacs system, so it is definitely worth spending some time and effort to get them right. We try to have two headshots per model; a large one at approximately 260 by 340 pixels, and a smaller one at *exactly* 120 by 156 pixels. In Wacs terminology the large on is known as the `bigicon` and the smaller one as merely the `icon` or `modicon`.

Note

We've found it is really quite important that at least the smaller size headshot is of a uniform size and shape for all models so that the indexes format nicely and aren't visually ragged. It doesn't have to be this specific dimension, but it's worked well for us and I'd commend to you not to change it without good reason. Similarly you might wish to make the larger one similarly uniform if you think you might use it in your own custom PHP pages or perl applications. Generally we don't use the larger icon except where it's the only one presented, and so that is less of a problem if it varies a little.

The exact location at which the icons are stored within the web server filesystem is specified by the `modbigicons` and `modicons` configuration entries in the `fsloc` section of the Wacs configuration file (`wacs.cfg`). It is assumed that these will be accessible via the web server without additional authentication, and will be in a location such that adding the value of `siteurl`, either `bigicons` or `icons` and then the values specified in the appropriate boxes in the Model Manager application will result in the full pathname to the image files for the icons. It should be perfectly possible to use any reasonable type of encoding for these headshots - JPEG, PNG, TIFF and GIF should certainly all be useable.

The best practice is usually to select a good, well-exposed head and shoulders or half-torso shot of the model looking towards the camera, from which to crop and scale the appropriate icons. As model headshots are often featured in the welcome page to a site and are generally not explicit, they are not covered by the access control system within Wacs. You may choose to let this influence you to select primarily clothed shots of the models for use in the headshots. In the examples below, we'll be looking at doing the scaling and editing using the GIMP image manipulation program, but you can of course use any other appropriate tool that you prefer. We're illustrating just one approach to doing it.



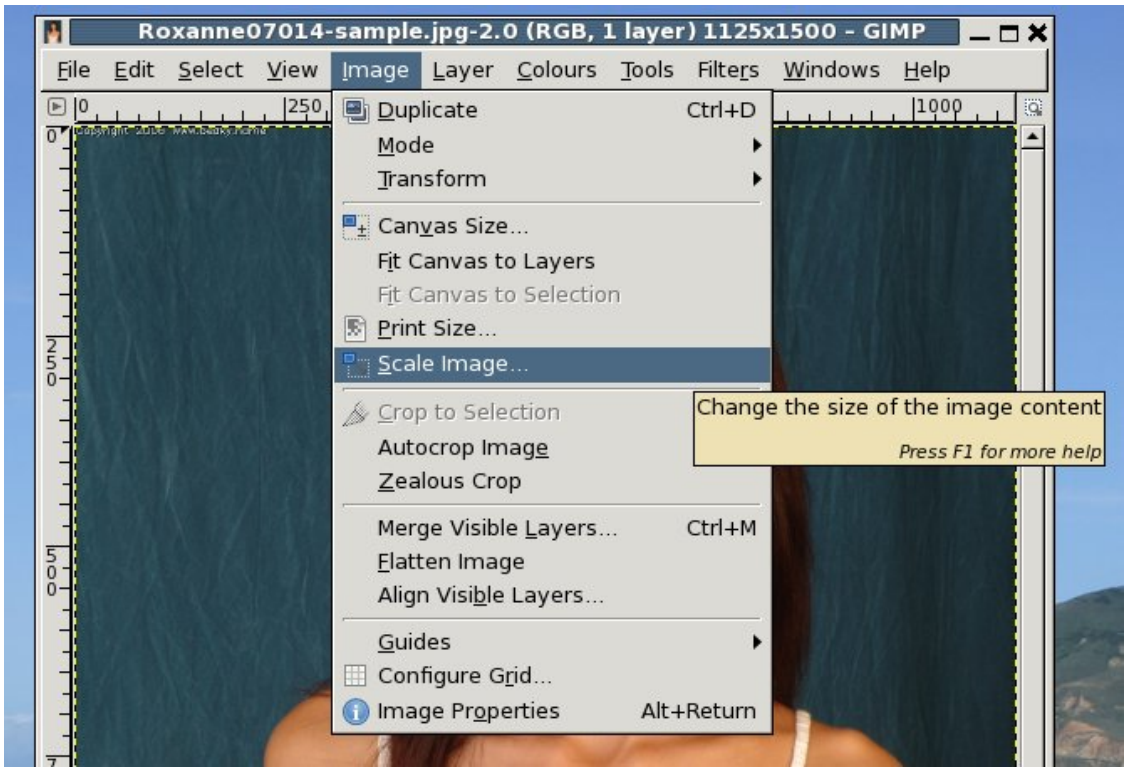
Tip

You can do the image editing and resizing anywhere - it doesn't have to be on the server itself - you can use your desktop PC, laptop - whatever you're most comfortable with. You just need to be able to upload the result to the correct directory on the WACS server system whether it's through a networked shared drive, a secure-ftp program or even a USB memory stick. On a Wacs system installed from either RPM or DEB package releases, the default locations are: `/usr/share/wacs/html/bigicons` for the big icons and `/usr/share/wacs/html/icons` for the smaller ones.

Making The Big Icon

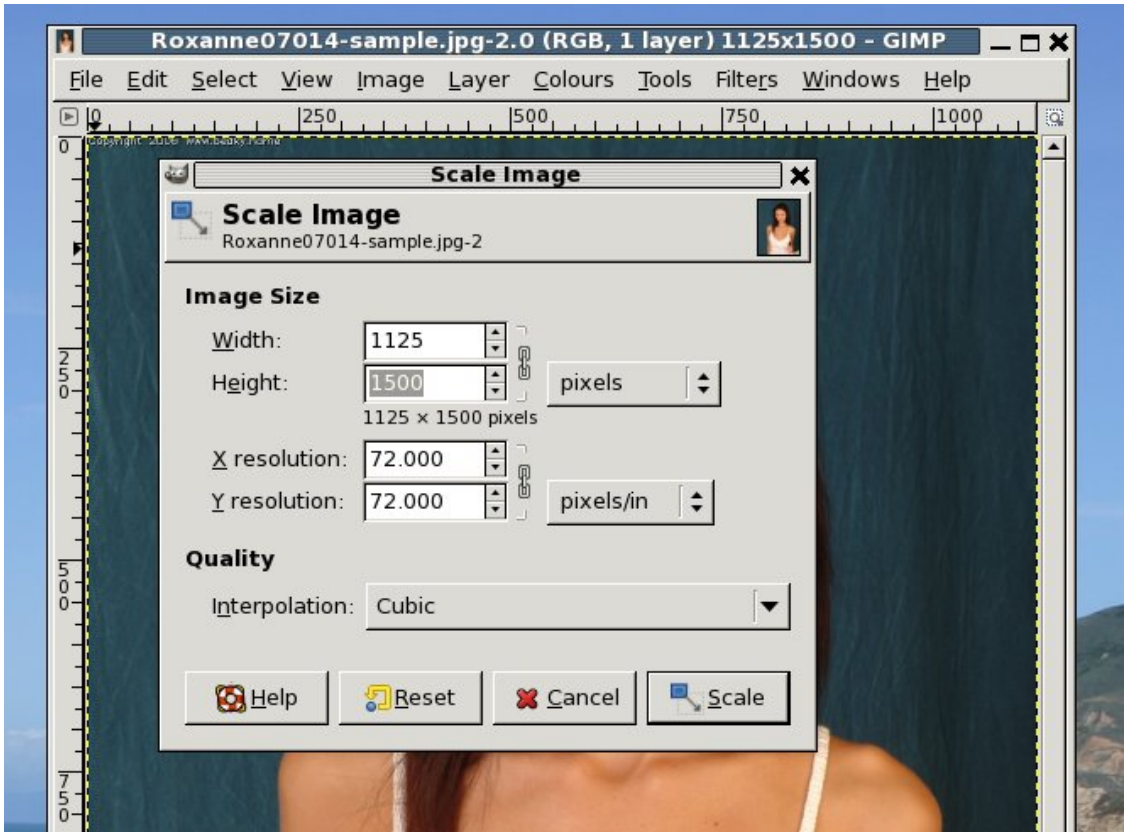
The first step of course is to start up the GIMP, either from the command line or via the Applications->Graphics menu, and then open the image we wish to work on. The screenshot above illustrates how the GIMP should look once you've openned the image; in this case we're using a shot of Roxanne.

Since the image is currently 1125 x 1500 pixels (that information is given in the title bar BTW), we're first going to reduce the size of it so that we have a reasonable chance of getting more than just her eyes and nose in the picture. To do this we call up the scale image tool from the Image menu as shown above.



The first headshot icon we're going to produce is the larger of the two, the one at 260 x 340 pixels. We will then produce the second smaller one by scaling down and/or cropping further that image. Anyway, the key point to bear in mind is that we're aiming for a portrait shaped image of 260 pixels across and 340 pixels tall.

Once you've called up the image scaling tool, the following dialog box opens up and you get the chance to reduce the size somewhat. By exactly how much depends on both the image your working with and the size and style you want for your site. Some sites may prefer a full torso shot either clothed or nude, while others may prefer a simpler head and shoulders style.

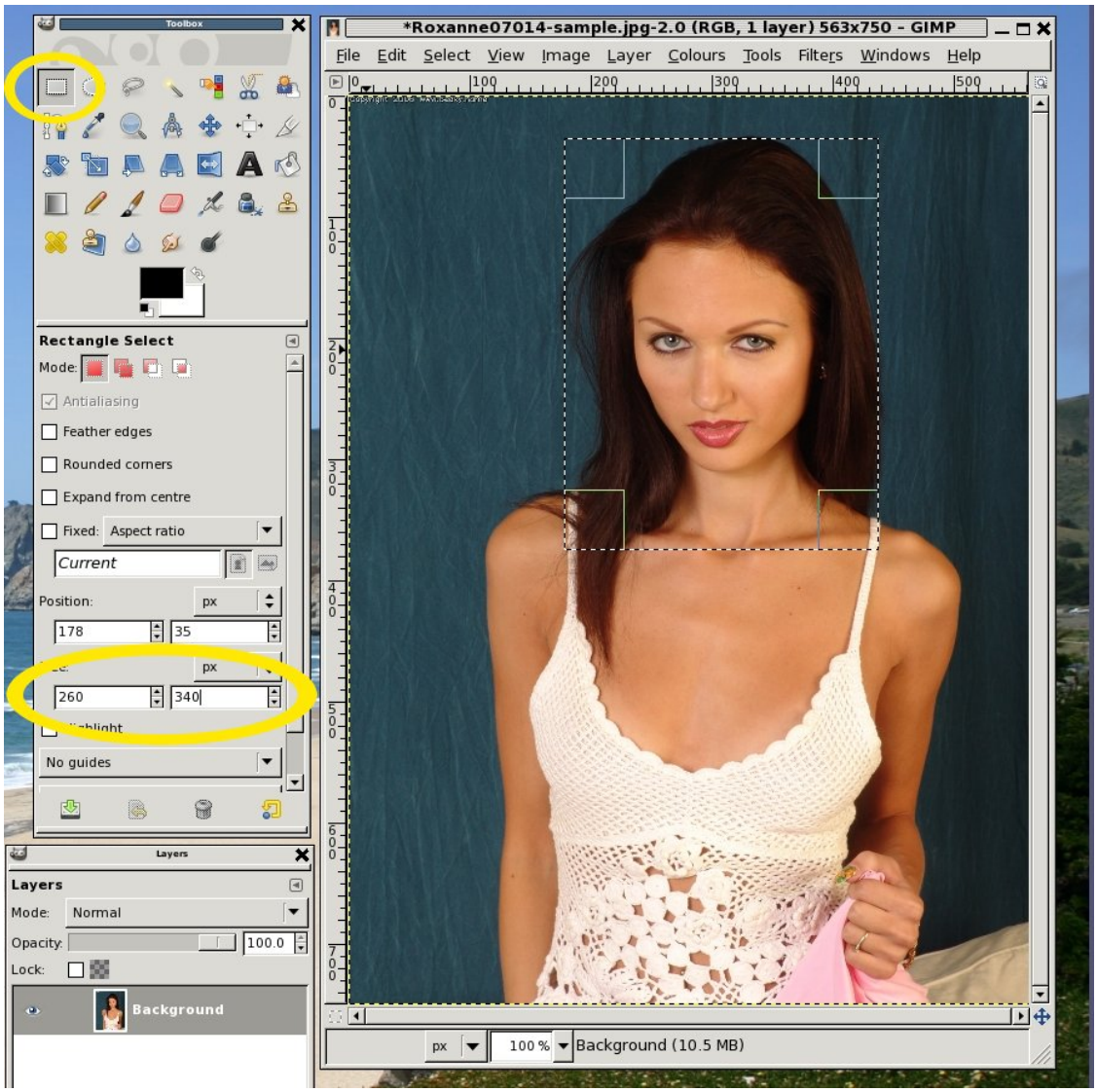


You'll notice that the two image size figures (here 1125 and 1500) are linked together with a chain icon. This means that they will change in union in order to preserve the current shape (aspect ratio) of the image. This is perfect for what we want here, so if we just overtype the `1500` with half that, ie `750` we will shrink the image to half it's current size. At that point we should in a position where a 260 by 340 selection from it will make a reasonable headshot icon.

Tip

If the photo we are working with is already more of a headshot than a full torso shot as this one is, we might choose to use a value of `500` to replace the `1500` instead of the `750`. This will mean that our 260 x 340 selection from it will comprise a larger proportion of the original image. Bottom line, you may have to try several different values until you find a value that you are comfortable with.

Once you've clicked on the **Scale** button, you will probably find that the image is now smaller than the window - going to View -> Zoom and choosing 1:1 (100%) should make it fill the window again. Simply pressing the `1` key should also have the same effect. The next step is to put Gimp into rectangular selection mode and select the appropriate area of the image.

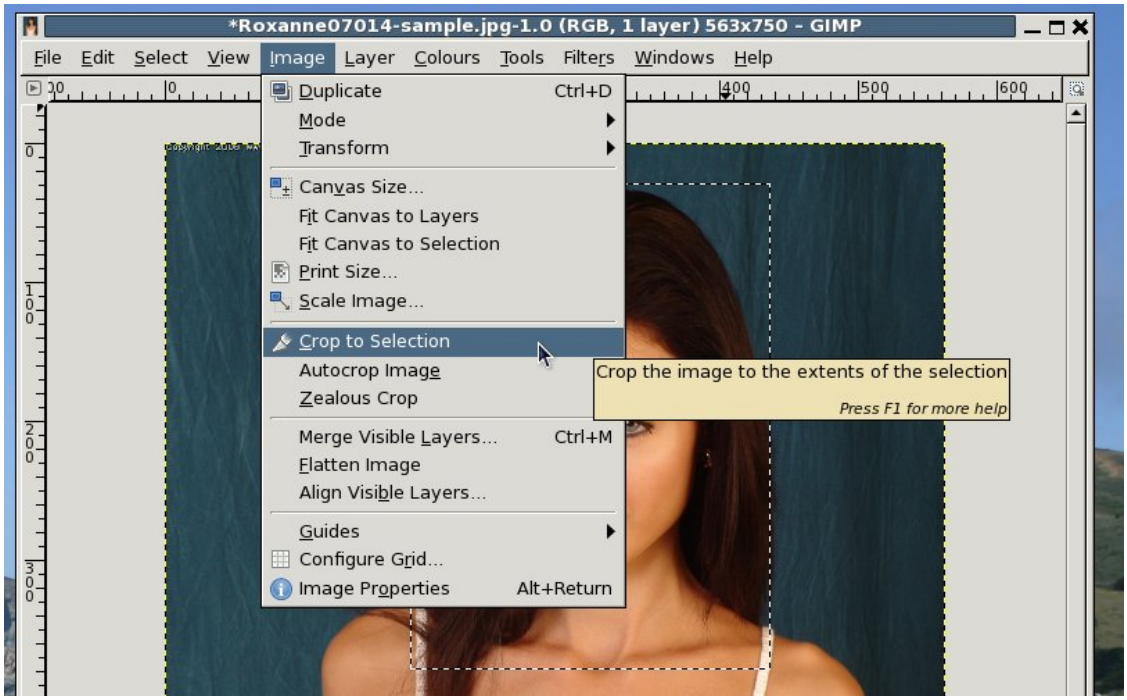


In the top left corner of the screenshot, we've highlighted where you need to click on the rectangular area selection tool. Once you've done this you left-click-and-hold on the top corner of where you want the icon to start and then stretch it to the bottom right hand corner. If you watch the boxes on the lower left side that are also highlighted, these will show you the size of the current selection. The pair of numbers above are the current position and are not relevant to this procedure. Once you have these numbers showing as `x = 260` and `y = 340` , you can let go of the left mouse button. A further left-click-and-hold within the selected area will allow you to move it around until the best cropping position has been found.

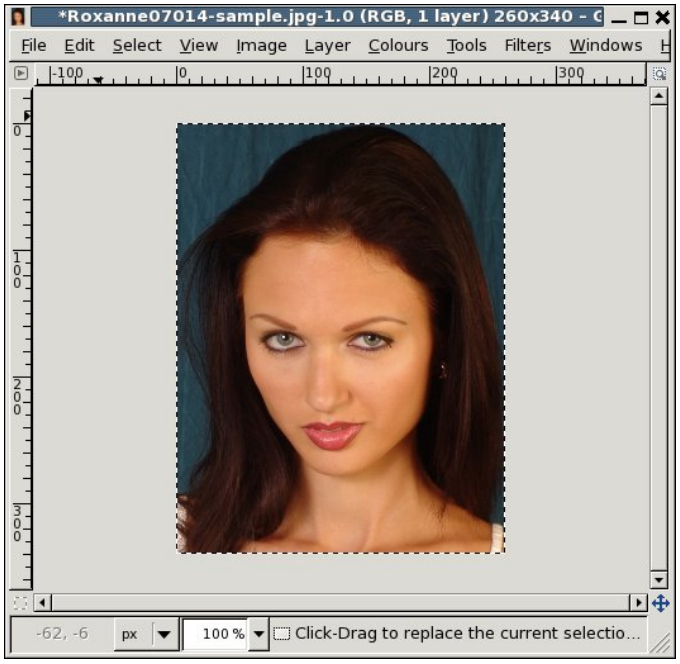
Tip

You don't have to struggle with the mouse to get the precise sizing right - you can just click in each of the two dimension boxes in turn, type in the desired value and then press return. This will make the wireframe crop box into the desired size and you can then just use the left-click-and-hold to allow you to maneuver it over the image until it's giving you the right sort of headshot.

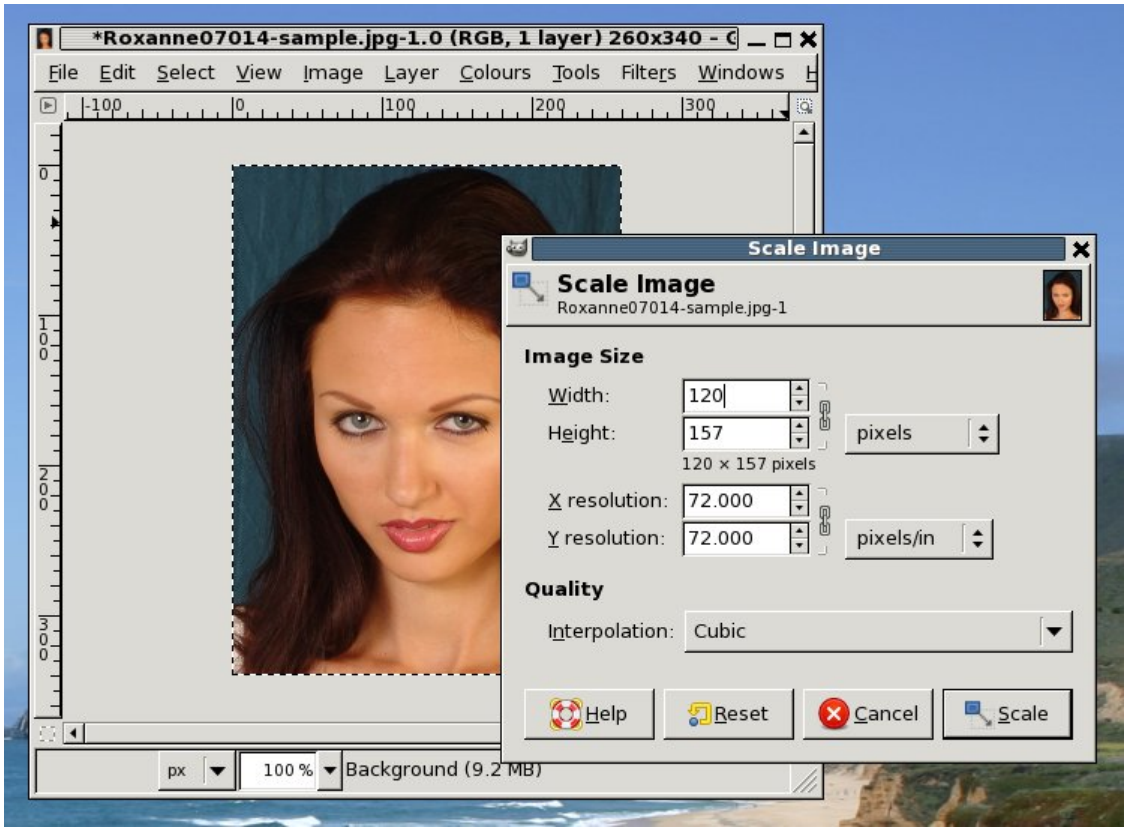
Once you're completely happy with the headshot cropping and the size of the area you're working on, you then need to actually perform the cropping itself. You do this with the **Crop to Selection** option on the Image Menu.



Once this is done, we should be left with our final version of our large size headshot. The file step is then to go to the **File** Menu, choose **Save As...** and give it a suitable name - in this case we've called it `Roxanne-big.jpg`. The **Save As...** will pop up a dialog box asking for the new name - you should make sure it includes the filename extension as this determines the type of file created. It defaults to the same file type as the image it was derived from.

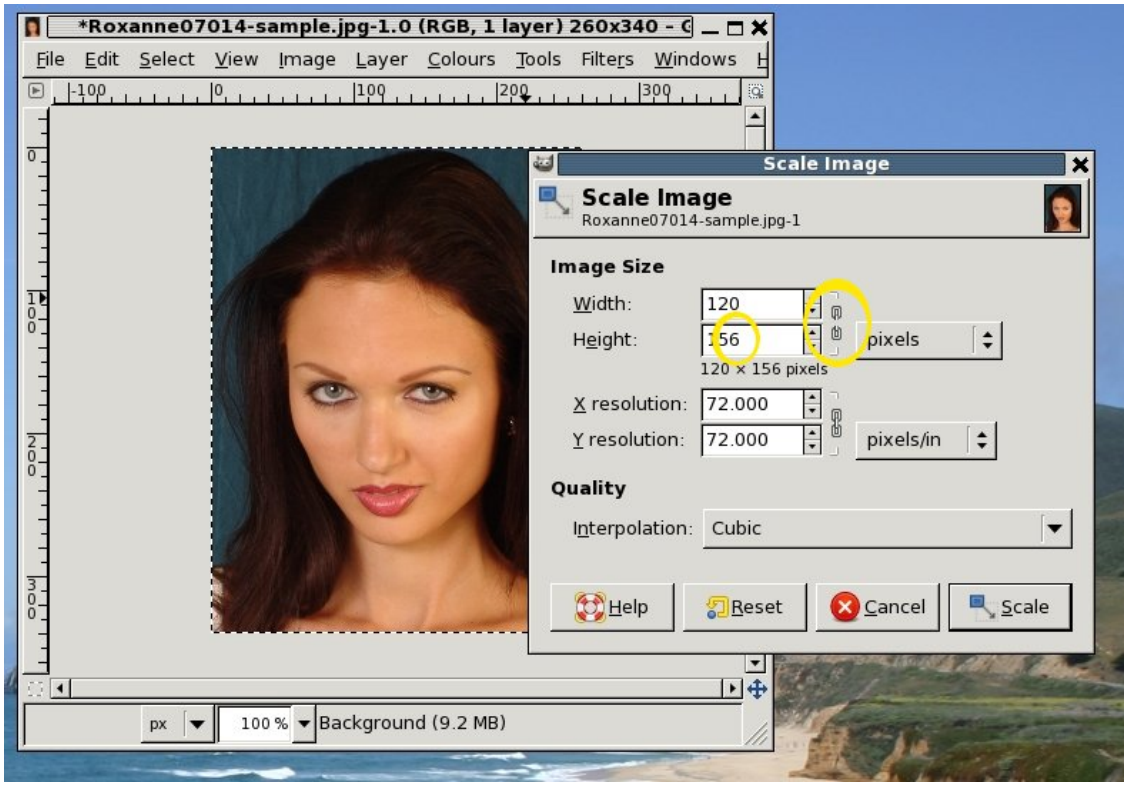


Once you've given it a name and clicked on the **Save** button, you will normally get a second dialog box related to options for the particular type of file you're trying to save. The only one we usually change is the quality, which we would recommend setting to 85% percent for JPEG images. For PNG images, we usually recommend compression level 6. Of course it doesn't really matter what you use, but you want to strike a balance between quality and the size of the file - these headshots will be downloaded a lot as people browse your Wacs site.



Making The Small Icon

The next step in the process is to produce the small icon at 120 by 156 pixels. The first step in this process is to reduce the icon further in one of the dimensions by using the **Scale Image** option from the **Image** menu. The simplest way to do this is to type one of the desired sizes into the image scaling dialog box as shown above. As you can see, our 260 x 340 icon reduces to 120 x 157 when you specify an x dimension of `120`. That's basically within a rounding error of the 120 x 156 value we were looking for - ie just one pixel off. While we want to be careful about distorting the image, changing 157 to 156 is a 0.75% error which is pretty negligable - it's not about to change the whole shape of her face or anything like that.

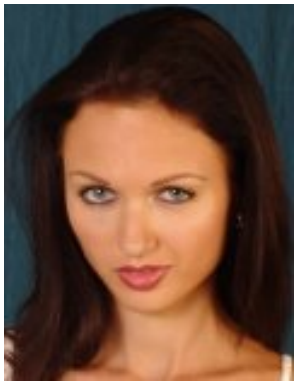


To make this change from 157 to 156 without affecting the 120 value (which is correct), you need to unlink the chain symbol between the two numbers. Then you simply overtype the 7 with a 6 and press the **Scale** button on the dialog box. The image is now scaled suitably and you can proceed to do a **Save As...** to save it, calling it something like `Roxanne-small.jpg`. You can now quit the gimp - we're basically done with it.

Icon Placement

The absolute final step is to copy the finished icons into the right place to make use of them. This is fairly dependant on how the relevant Wacs site has been configured, and as mentioned earlier, the definitive answer is the values of `modicons` and `modbigicons` in the **fslos** section of the Wacs configuration file, `wacs.cfg` for the site in question. The convention is to have the files named the same but in different directories, so in the default locations for the packaged versions of Wacs, this would be `/usr/share/wacs/html/icons/Roxanne-1.jpg` for the smaller icon and `/usr/share/wacs/html/bigicons/Roxanne-1.jpg` for the larger icon.

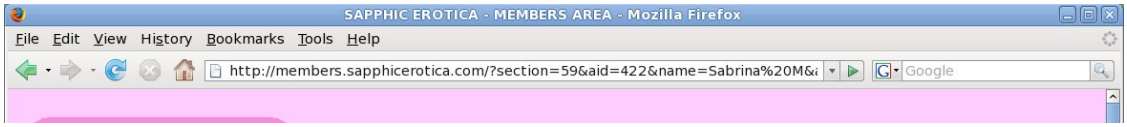
Table 5.1. Final Example Headshot Icons

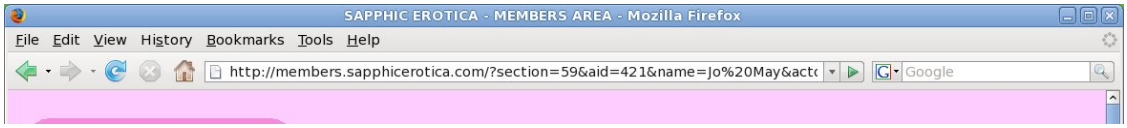
Big (260x340) Icon	Small (120x156) Icon
	

Determining The Site Id

The other significant step in preparing to create a new model record is determining the ID for that model on a relevant site. While not absolutely essential, it really does help if we have a publicly recogniseable *Handle* on who this model actually is. This is perhaps less important if you're running your own site with your own content, but there's still lots to be gained by offering cross links to other sites especially if they offer commissions for referals. A lot of the WACS tools, particularly those related to migration (see Chapter 12, *Migration Tools*) will find things a lot easier and drop the ball less often if they have these references.

Consider for a moment these two URLs screen shots which are taken from a web browser exploring the Sapphic Erotica site:

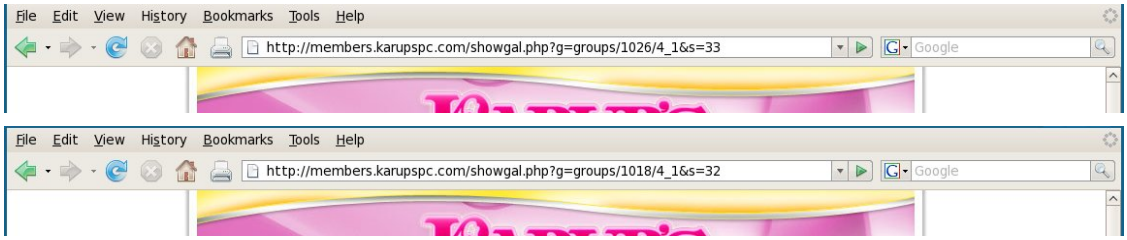




What you'll see is that the URL is largely the same, changing only in two places with each different model you select from the model directory on the site. The things that change are the `aid=` and the `name=` values. In other sites, it's quite common for the name to not even be mentioned at all and there simply be a model id number that determines which model should be shown. Often the what you're looking for varies,

it might be `md_id=` on one site, `modelid=` on another and so on but if you look at several of the URLs it should become apparent what is actually changing when you look at different models. Sometimes this can even be a simplified version of her name as it used to be for instance on the Karup's PC site.

Karup's PC has now changed to using purely numeric model numbers and here are the examples for Sabrina and Roxanne from the new KPC site:



What you'll see is that the URL is essentially the same, the bit that is changing is the number between `showgal.php?g=groups/` and the `/4_1&s=32`. We can only really guess at what those two pieces mean, but looking at the number between them as we explore the model index shows that it is only that bit that is changing. This therefore tells us that Sabrina's number is `1026` and Roxanne's number is `1018`. This is the bit you want!

From the examples above, we now know that to refer to Sabrina M from Sapphic Erotica, we know her reference there is *422*; we also know that her number on Karup's PC is *1026*. Similarly if we're referring to Jo May, we know her reference there is *421*. All of this information goes into the Identity Map (idmap) table in the database. Each model can have many different IDs across a vast range of sites - usually all we actually need is one solid one in order to be unambiguous as to who we're referring to. Here's the information we actually need to make a site record: site code, site reference number and name on this site.

Table 5.2. Typical Site Identity Information

	Sabrina		**Roxanne**
Site	SE	KPC	KPC
Key	422	1026	1018
Name	Sabrina M	Sabrina	Roxanne

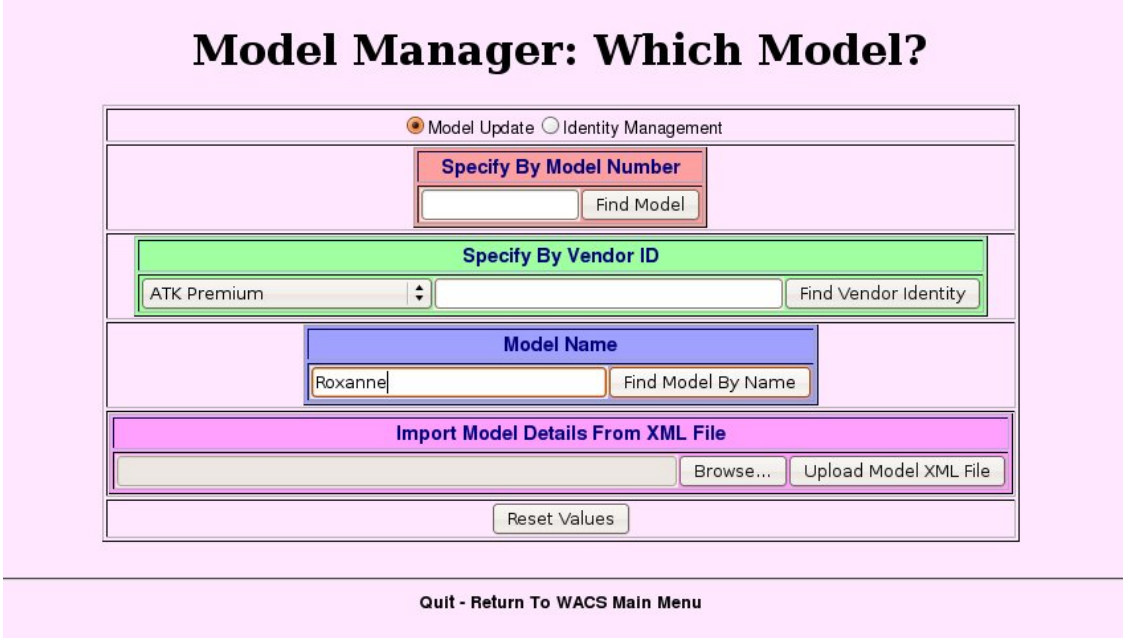
With this information on our model gathered and a nice pair of headshot icons prepared, we're now in a position to move forward and actually look at using the Wacs collection administration software. There have been a lot of concepts to understand in the last three chapters but hopefully it'll give you a clearer idea of what we're trying to achieve going forward. In the next chapter, we look at what you can actually do with model records...

Chapter 6. wacsmodelmgr - The Wacs Model Manager

Introduction

One of the really major aspects of Wacs is it's ability to access sets via a range of methods, one of the most important of which is to be able to look at the work of a specific model. When we're adding new sets, we want to be able to add that set to a model's portfolio and in order to do this, the model has to already exist within the Wacs system. This chapter will look at the model manager application, outline how we create a model for the first time and show you how to update a model record. If you want to look at an example model record for reference, there are currently two model pages provided as XML files within the samples directory of the Wacs distribution. These can be imported into the Wacs system using either the model manager itself (a new feature in Wacs 0.8.3) or using the command line tool **wacsimport** which is described in Chapter 12, *Migration Tools*.

With a suitable account running in administrator mode, the model manager is the top link (and also the default) on the rightmost menu, **Maintenance**, on the Wacs front page. Note that if you're not offered this menu, but instead see the **Preferences** menu, your account is not currently running in administrator role. Please go back to Chapter 2, *First Steps* for details on roles and how to change them.

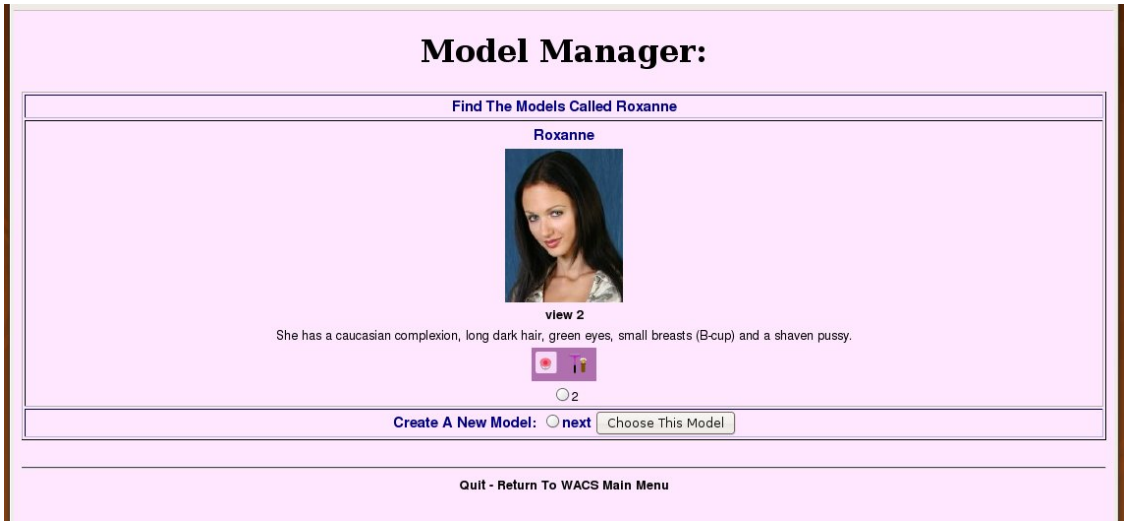


As you can see, this screen offers quite a few different ways to locate the model you're looking for. The first of these is by simply specifying an existing model already defined on this Wacs installation by model number and then hitting the **Find Model** button. The second allows you to select by a model's identity on a known site - if you remember from the last chapter (the section called “Determining The Site Id”) that Sabrina has a key (aka reference number) of 422 on Sapphic Erotica. You would pull down the list of sites, select **Sapphic Erotica**, type in `422` and then click on the **Find Vendor Identity** button. The third option does a name based search and then offers a list of matching models (if any) and is what we're about to use in the next few examples. The fourth option is use in conjunction with the migration tools and will be descibed in a later chapter (Chapter 12, *Migration Tools*).

Before we look at how to actually create a model, we're going to just quickly examine an existing record - in this case Roxanne's - so you are familiar with the overall layout of the model manager. We start off by entering the name of the model we wish to examine in the box labelled **Model Name**, and then click on the button **Find Model By Name**. When you click on this button, The Wacs Model Manager will attempt to show you the headshots and descriptions of every model known by that name on the current Wacs site. This includes aliases and non-standard names specific to only one site.

Note

With this particular name, *Roxanne*, it would have probably been more sensible to have searched for the shorter alternative spelling of *Roxan* to just cover the possibility that she is known to this Wacs site but under the shorter version of the name or as Roxanna. We do get the chance to edit the name later.

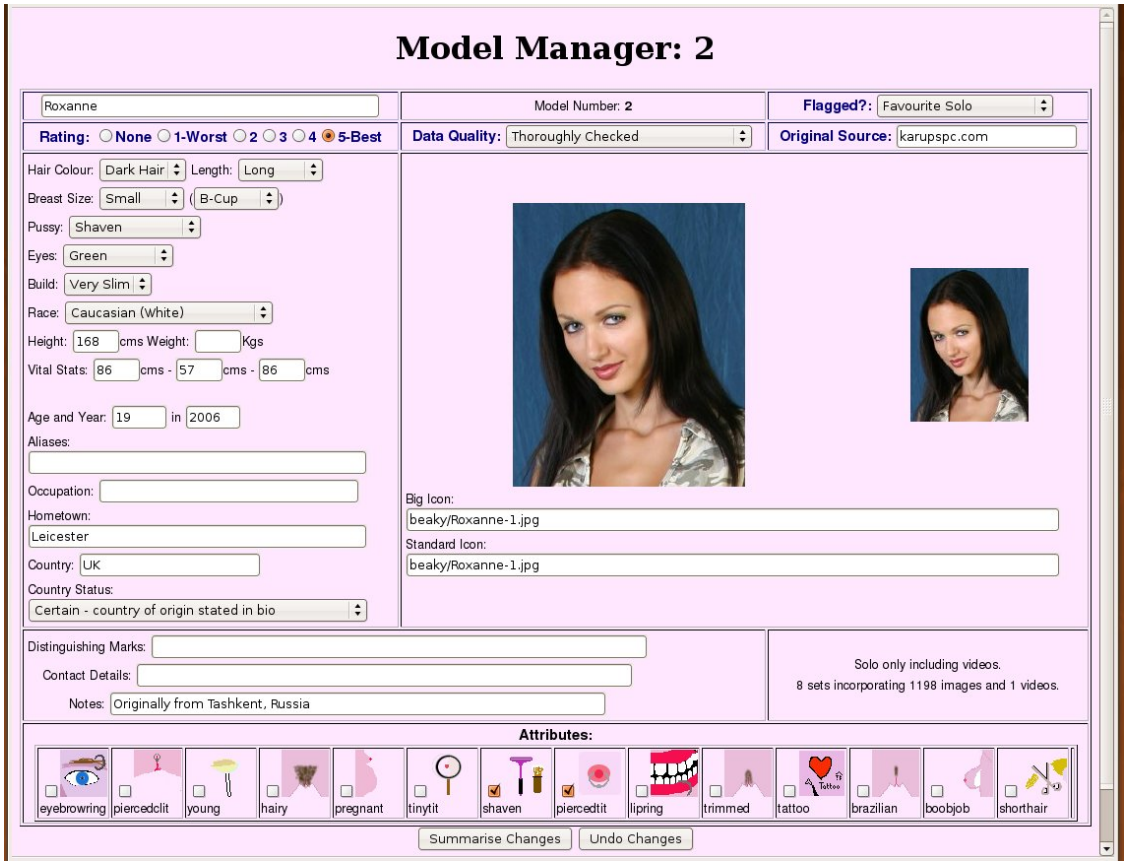


In this example we've been given two options - one is to choose existing model number **2** (Roxanne) or **next** - for now we're going to select our existing model number **2**. If we wanted to create a new model, also called Roxanne, we would simply select **next** which allows us to create a new model. If there had been other models with this name, we would have been shown each of their headshots and basic details as well as the option to create a new model.

Tip

The tick box you need to select is just below the attribute icons - with just a single digit next to it, it may be a little difficult to spot on first viewing. The **view 2** link just below her headshot allows you to view her model page to confirm any details you wish to confirm before selecting her. After visiting it, you can merely push the back button on your web browser to return to this screen.

It's important to understand that it's at this point that we're deciding between updating an existing model and creating a new one. The point here is that before we create a model we want to have confirmed that there is not an existing entry for this model. If we select one of the existing models, Wacs allows us to edit the model record for that model. If we select **next**, we start on the process of creating a new model record.



This screen contains most of the data that we store for any given model, grouped in hopefully reasonably logical fashion. Along the top we have the basic data; name, rating, source, any special flagging and the data quality indication. On the left we have all of the descriptive data, while to the right we have the two headshot icons and the respective paths to those files. Moving down the page we have descriptive information about the model, and at the bottom a row of icons that can be applied and tick boxes to make those marks. At the very bottom we have the web form buttons and quit link.

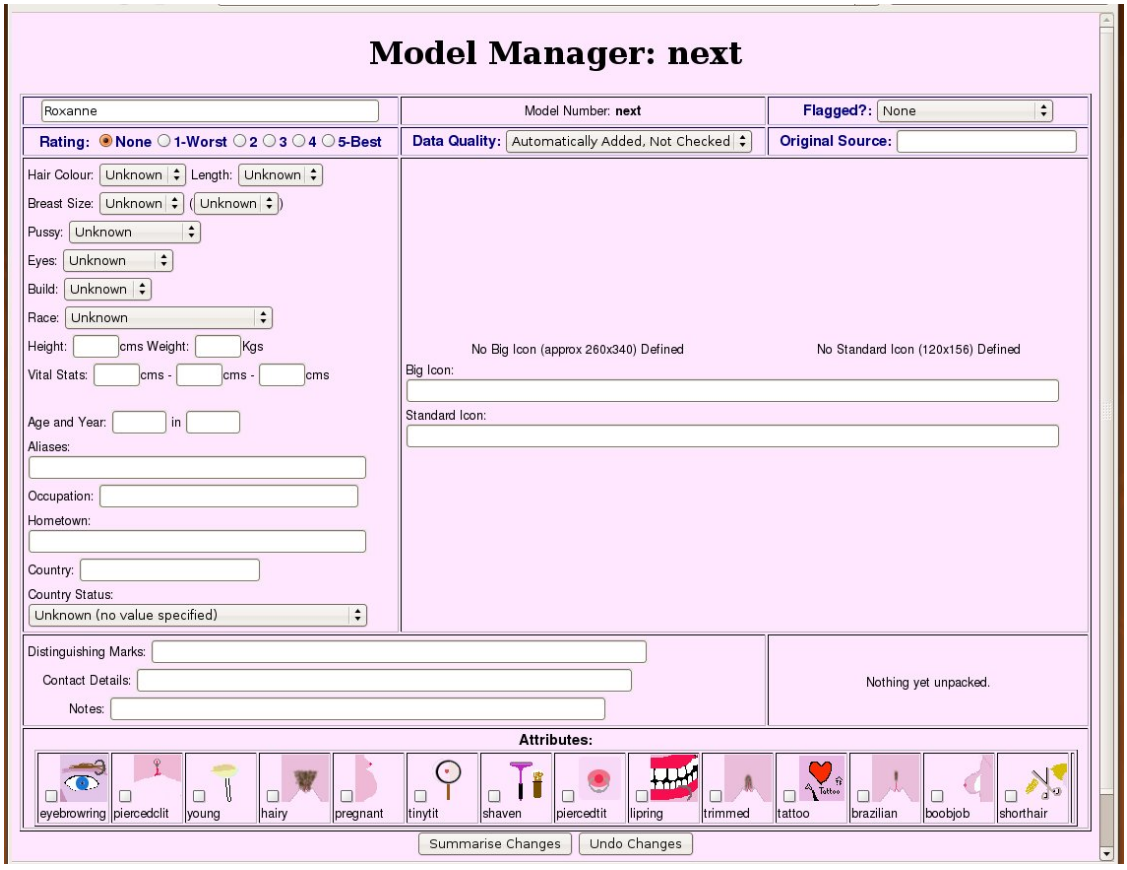
Now you have a basic idea of how the model manager works, we're going to move on to how to actually do various tasks using it. The first one being to create a new model record.

Creating A New Model

So now you've seen a quick overview of the model manager, let's actually do something useful with it. The first thing we're going to do is to actually create a new model record. For this we're going to use Roxanne as an example, making use of the two icons we created in the previous chapter (Chapter 5, *Preparing To Create A Model Record*). If you already have Roxanne defined, there's nothing to stop you using any other model's photo you happen to have lying around to work through this example.

We start off the process as before by going to the main menu and selecting **Model Manager** from the **Maintenance** menu. Once again we enter Roxanne into the model name box and click the **Find Model By Name** button. We get the same choice as before but this time we select **next** as shown:

If you now click on the **Choose This Model** we move on to the next screen which will be a blank template for you to fill in. Initially only two things are filled in -- the name of the model (ie Roxanne) and the pseudo model number of `next`.



Tip

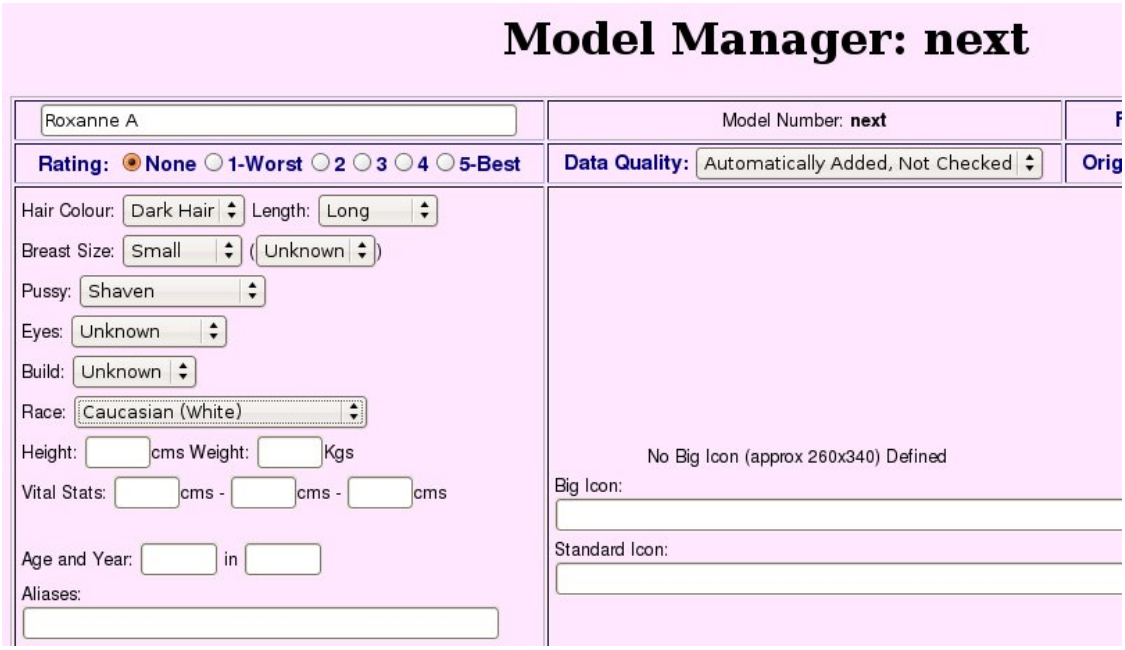
If you already loaded the sample Roxanne record onto your Wacs test system, you may wish to rename this new record to *Roxanne A* - you can do this easily by just adding the *A* bit on the name field at the top left corner of the model manager screen.

Note

What's this about next? What happens is that when we've finished working with creating this model record and actually save it to the database, `next` will be replaced with the next available model number at that time. This is to try and ensure that on really big sites where two people might be working at once on creating new model records, there is as little a chance as possible of them both competing for the same model number.

Basic Attributes

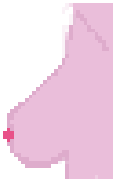
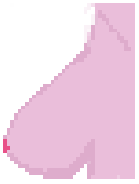
Let's start with the basic model attributes - the colour and length of her hair, the size of her breasts and the way she keeps her pubic hair. These are to be found in a group in the upper left portion of the main model manager screen.



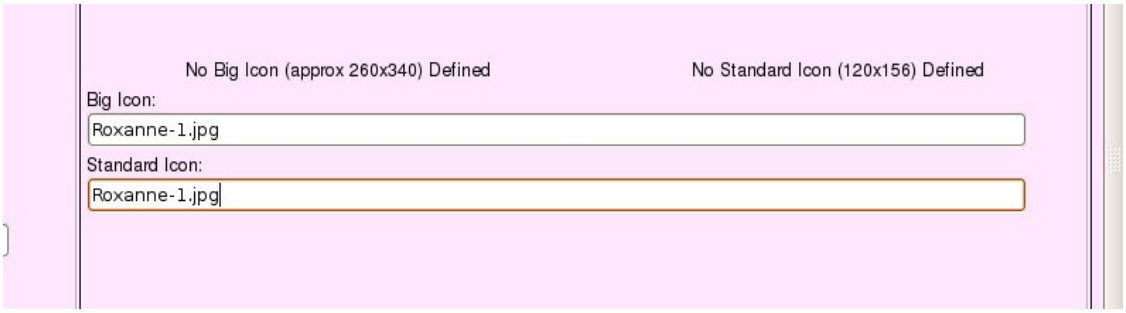
Hopefully the choices for hair colour and length are reasonably self- explanatory - the key thing to remember here is *Usual*. If she usually has long hair, and then suddenly appears on a couple of sets with a short bob cut, the length should remain long (although you might wish to manually flag those sets with the **shorthair** attribute - how to do this will be covered in Chapter 9, *Wacs Set Manager*). The pubic hair value should only be shaven if her pussy is *completely* shaven; a Brazilian style shave leaves a small tuft of pubic hair above her clit but nothing on either side. Note that this is such a decider for some people that we actually mark up sets whereever we can with an attribute describing the pubic hair style.

The breast size value is much more subjective but our take on it is based upon the probably apocryphal description of how to decide when a young woman needs a bra - simply put “Will a pencil put underneath her breast stay there?”. The table below gives our rules of thumb for these values:

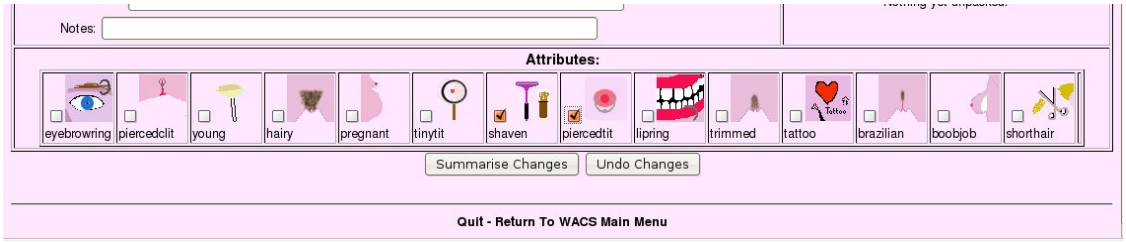
Table 6.1. Breast Size: Our Take

Value	Icon	Description
Tiny		Rolling the pencil up her chest, the first bump you notice is her nipple... if at all. This should also be marked with the *tinytit* attribute.
Small		Rolling the pencil up her chest, it stops when it reaches the underside of her breast but probably wouldn't stay there.
Normal		The pencil will stay there, no problem.
Large		Hey Lady, Where's My Pencil?

One final aspect we'd just comment on in this area is that of race - most of these should hopefully be self explanatory except that we have divided up Asian into two using the traditional British descriptions - the word Asian is used for complexions native to the Indian sub-continent while Oriental is used for those from South-East Asia and the Chinese sub-continent. Hopefully this distinction is reasonably obvious.



The next section we're going to look at is the fields for entering the names of the previously prepared headshot icons. As discussed earlier the files need to be copied into the directories specified for `modicons` and `modbigicons` in the `wacs.cfg` file. You do need to include the file type extension on the end of these filenames. It's perfectly OK to place icons into sub-directories of the model icon directories, and to give the relative pathname including directory path components here.

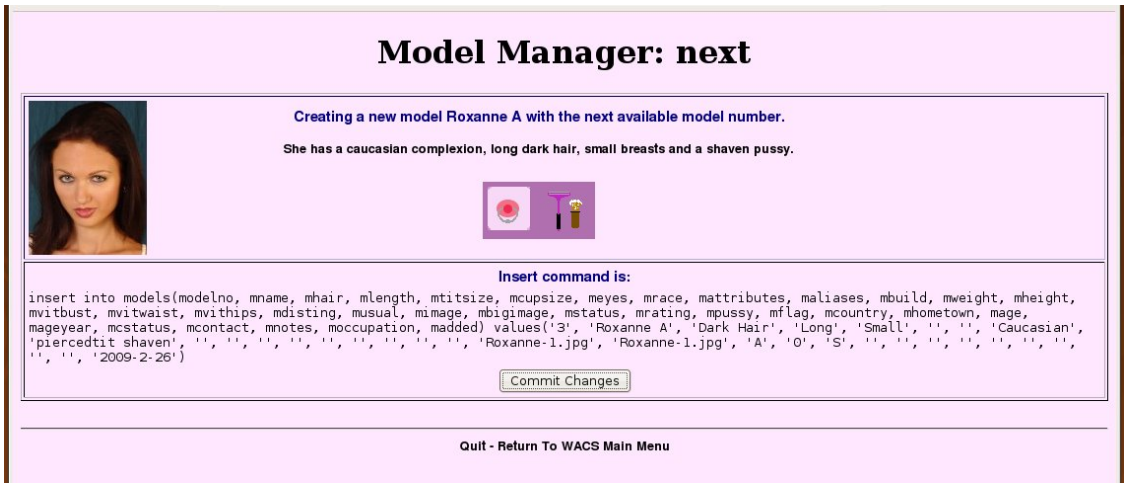


Finally down the bottom of the model manager main screen, we have the mark-up attributes. These are attributes that apply directly to the model herself and cover such things as pubic hair style, and various types of body modification: tattoos, piercing, breast augmentation, etc. Normally what you specify here will be automatically copied into the attribute marking of all sets she's involved in. That's not to say it isn't possible to remove specific details on a per set basis - it is. Thus if a model gets a tattoo or boobjob later in her modelling career, these can be added to just those later sets. Generally it's a value call as to what is the most common state across the collection you have of her as a whole.

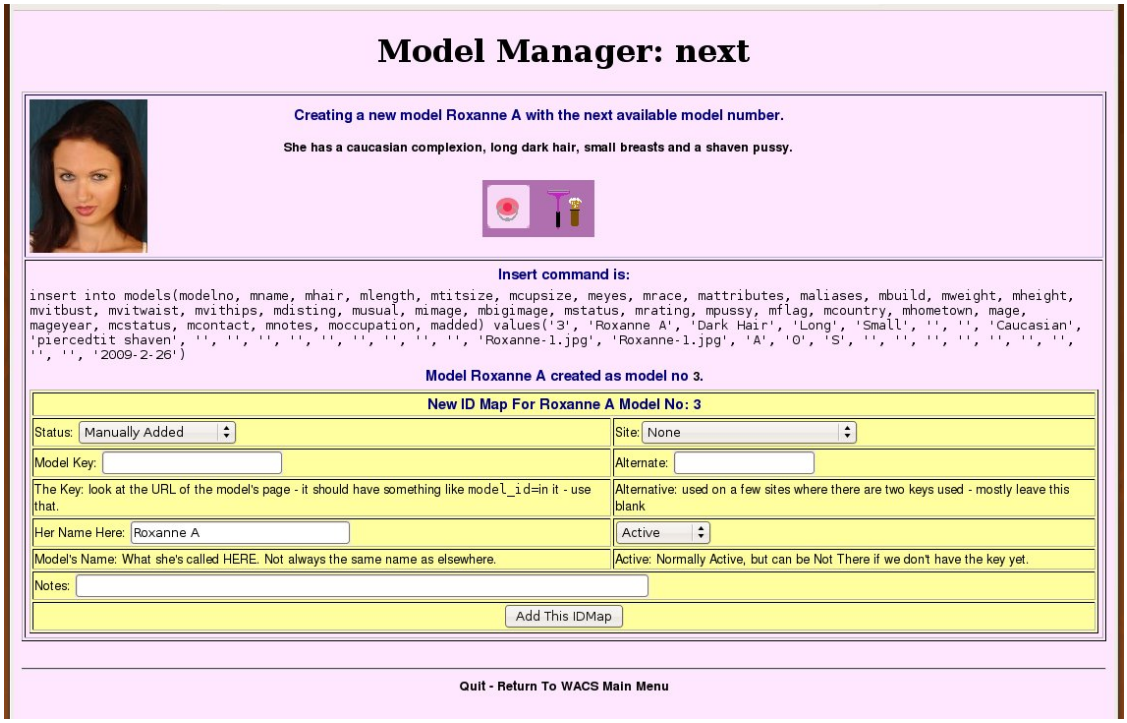
For Roxanne we set just two model attributes - **shaven** because she is in our sets and **piercedtit** because she has a stud through her left breast's nipple. Once we've selected these, we can click on the **Summarise Changes** button and we'll move on to the next model manager screen.

Inserting Model Records

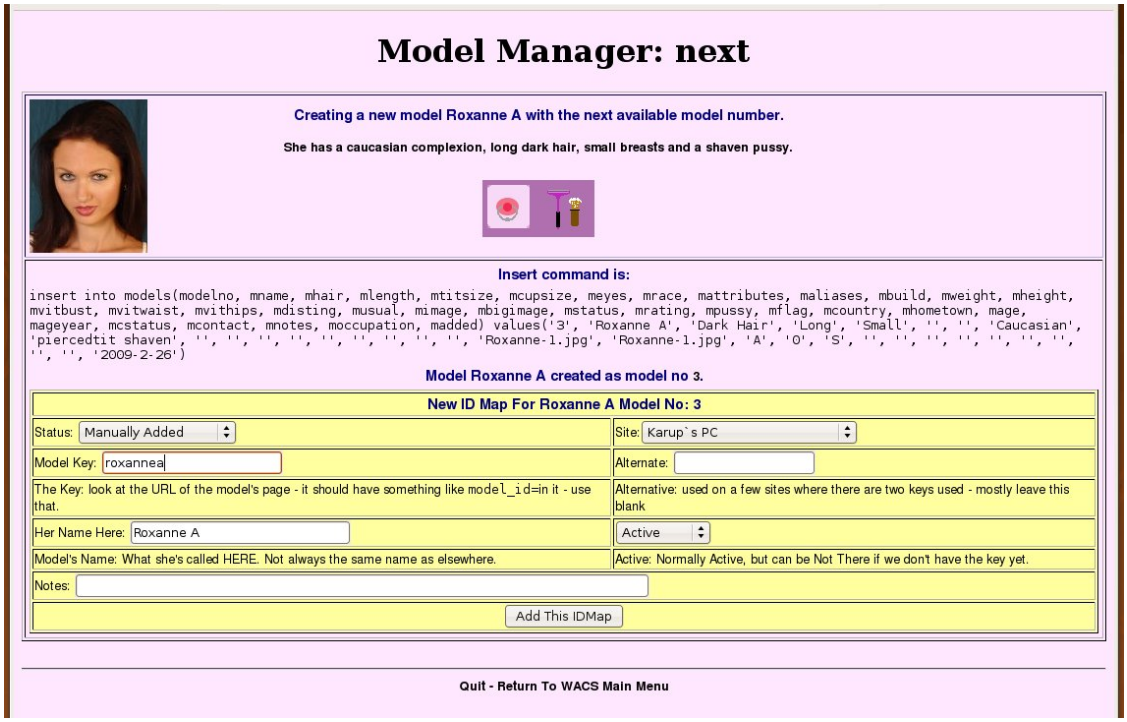
At this point, since we're creating a new model record, we go through the process of inserting a new record -- that is the database terminology for it. The first screen we see after the main model data entry screen, is the one shown below which summarises what the new model record will look like and shows you the headshot that it's going to use so you can confirm it's all as you expect. Notice that we do actually show you the SQL command we're using at this point. Should anything go wrong and you're seeking help - please cut and paste this SQL statement into the request for help - it will make it much easier for us, or whoever you get your support from, to work on solving the problem.



Do take a careful look over the details given here just to make sure that you haven't made any obvious mistakes, like misspellings or ticking attributes that just don't apply. Once you're happy, click on the **Commit Changes** button and you should see this:

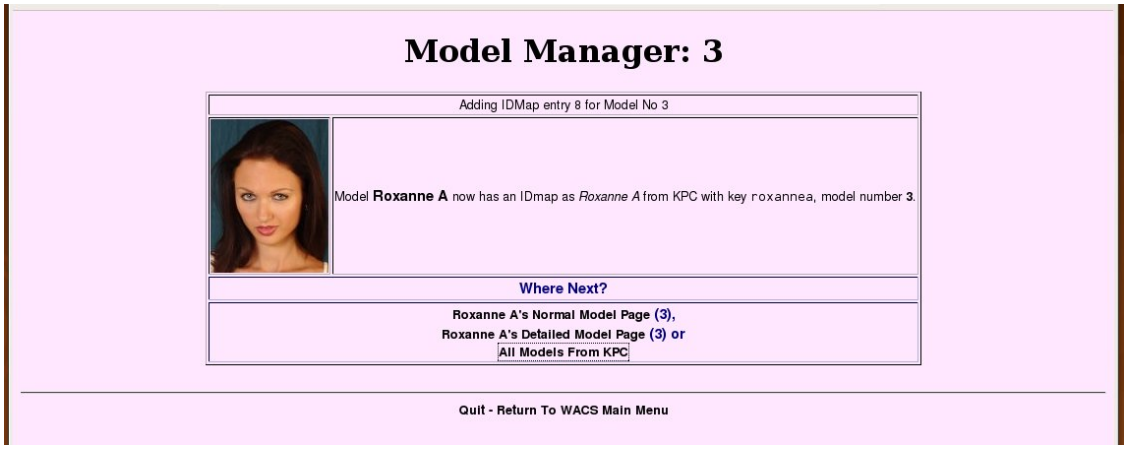


As you can see this confirms that the main model record has been created and immediately prompts you to enter an initial identity. You don't have to do this, you can click on the link over the “created as model no” link or on the **Quit - Return To WACS Main Menu** link at the bottom. The pull down menu allows you to select from the predefined sites. You can add additional sites using the Wacs Vendor Manager (described here: the section called “Vendor Manager”).



Here is a purely fictious example of an IDmap for a fictious model called Roxanne A, but hopefully it'll illustrate how we add a new IDmap. We actually only need to add two items over and above the defaults

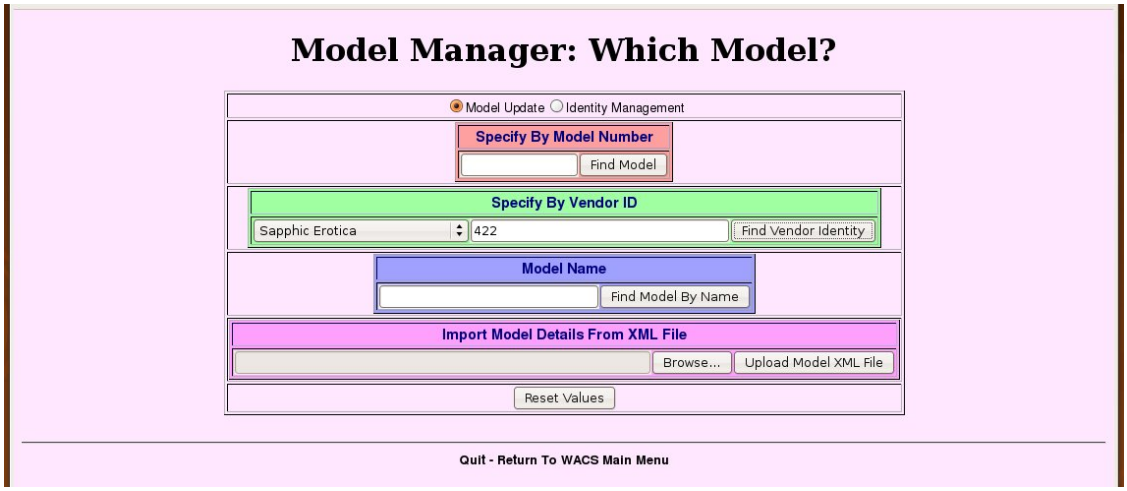
- the first is the **Site** where we've selected `Karup's PC` from the pull down list; the second is that her key here is `roxannea` (which doesn't exist BTW). If we'd been working on Sabrina, we'd have choosen `Sapphic Erotica`, entered a key value of `422` and added a space and a capital *M* after her name as she is known there as *Sabrina M* instead of just *Sabrina*. Once we've got the necessary information entered into this form, we click on the **Add This IDMap** button and the IDMap should be added to.



This page basically confirms that our new initial IDmap has been added and offers us a number of alternative “Where Next?” options. At this point we've basically finished the process of adding a new model to the WACS system - granted it's a little involved, but it is still fairly quick and easy. In addition to adding new model records with the Wacs Model Manager, it is also possible to add new model records using the command-line based **addmodel** facility, although mostly these days that would be used when creating scripts to import models automatically from other systems. More information can on **addmodel** can be found in Chapter 15, *Command Line Tools*.

We have now covered how you create a new model using wacsmodelmgr but that only covers a fraction of what it's capable of. It can also update existing model records, import model records from XML files and add new IDmaps (records of a model's identities on various sites). We'll look at just the updating of existing model records now and cover the rest of these topics in a later chapter (Chapter 14, *More About Model Manager*).

Amending Model Records



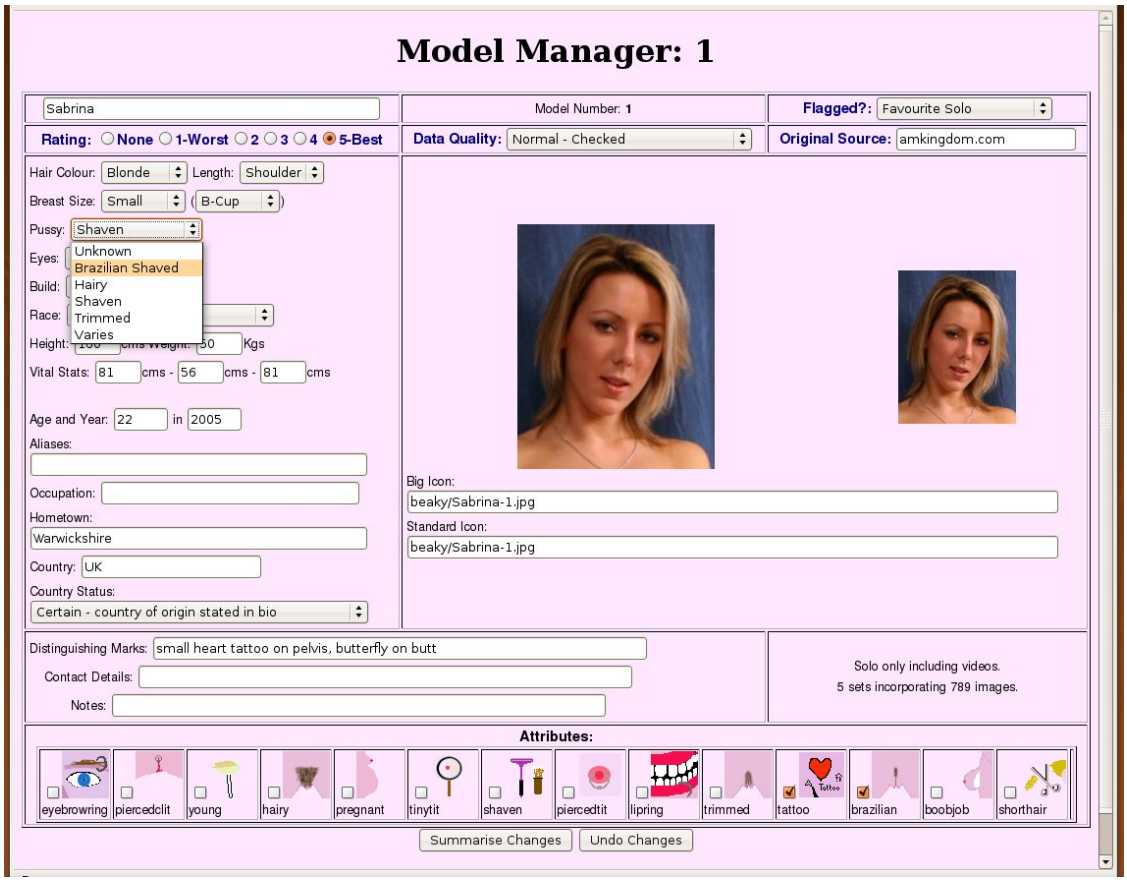
Here we're just illustrating another of the ways to find an existing model through the modelpage front page menu, this time using a site id as the way in - here Sabrina's Sapphic Erotica ID.

As we saw in the previous section, the Wacs model manager is a very effective way of creating new model records, but it also serves to allow you to edit existing model records. When using it for this purpose, we have two choices of how we get to it - we can find the model manager on the **Maintenance Menu** as before, or we can use the **edit model details** link on the right hand side of the masthead of the model page (which is shown only when your role is administrator of course).

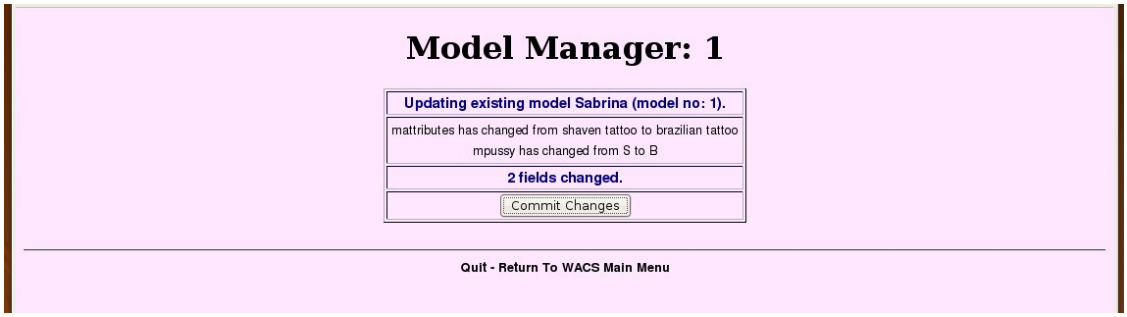
Note

Since we're using a unique ID, it skips straight from the front ways to find a model page to the detailed model entry without showing us a choice of headshots etc as it did before.

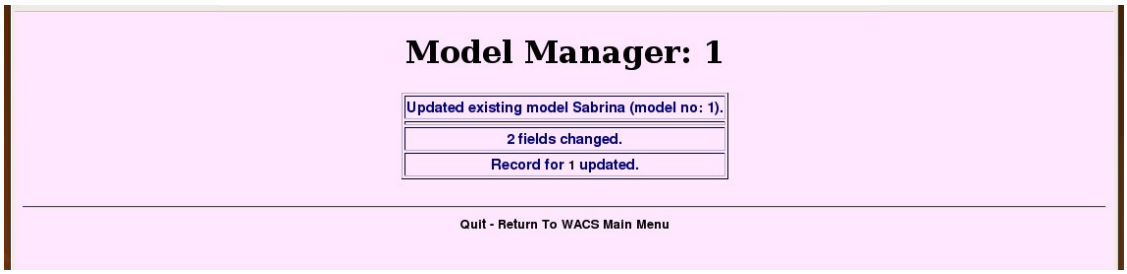
For this example, we've decided that since all of our shots of Sabrina feature her with a Brazilian shaved pussy, as do a fair few of the other sets we have of her, we're going to mark her as being normally Brazilian shaved rather than fully shaved. Do note that her previously marked earlier sets showing a full shave will not be automatically updated - those will remain as they were. Only sets that had no pubic hair related attribute *might* be updated (you'll find out why this is only a might in the chapter on the set manager).



This particular change needs to be made in two places - the first is her basic description, which is in the top left area of the model page - here we pull down the pussy menu and change **Shaved** to **Brazilian Shaved**. The second is down at the bottom in the attributes section where we need to untick the icon with the razor and brush (for Shaved of course) and instead tick the icon for a Brazilian shave. The other attribute we have for her, namely tattoo, as she has a couple of small ones, remains unchanged. Once we've done that, we click on **Summarise Changes**.



This summarises the changes we're about to make - the first one reveals a little about the internal working of the database as the pussy type is actually represented by a single character that is converted to the textual phrase by WACS. It's therefore changing an `S` into a `B`. The second one is a little more transparent as `shaven tattoo` becomes `brazilian tattoo` for the attributes value. It's a good plan to check what changes it says it's going to make for anything unexpected, then click on **Commit Changes**.



The alterations to Sabrina's model record are now complete. Hopefully that is reasonably straight forward. Now, there are a few more complexities of the model record and it's interaction with the Wacs Model Manager that we need to discuss. There are a couple of pull down menus right at the top of the model manager screen titled **Flagged?** and **Data Quality** that probably need additional explanation. We include the ability within Wacs to have a number of global selections of models which we call favourites. We're sure how they'll be used will vary from site to site, but the default Wacs configuration has six of them. There are:

- **Favourite Solo** - intended to mark those with outstanding solo sets
- **Favourite Cuties** - intended for very pretty models but who are not (mainly) involved in any explicit action
- **Favourite Lesbian** - intended for models who work well in lesbian sets with other models
- **Favourite Straight** - intended for models who have interesting or prolific straight sets
- **Currently Featured Models** - intended either for use by web site developers for highlighting those with active additions or for collectors to mark those they're currently working on actively.
- **Placeholder** - this is basically for a wrong or broken model record - either a fake one created to download from sites without proper model indexing, or one that is simply wrong. It'll work as a normal model record but just won't appear in any Wacs index page.

The second menu, **Data Quality** is primarily to allow one to indicate how much work has been done on the model record - the normal state is either **Automatically Added, Not Checked** or **Manually Added, Not Checked** which is a bit of a hang-over from the early days of Wacs but basically means the data about the model is very raw. Once you've checked her biographical data either from an upstream site, or one of the many resources on the internet, you would probably set it to Normal. If you've either spoken to the model herself, or checked a number of sources thoroughly, then you might choose to select Thoroughly Checked

instead. This quality assessment can be searched on if required. The intention is to use it extensively in the Wacs Meta project where we hope to encourage people to exchange model records in Wacs's XML format.

The other field in that area at the top of the screen, **Original Source** merely indicates where we originally found this model's photo sets. A web site owner might choose to put the booking agent or studio used here, but mostly it's assumed to be for collectors.

And so finally we reach the thorny problem of units.... we know this is going to be contentious but we've had to make some decisions and we're happy with why we did. If you're from the USA, you'll probably be spluttering about this; if you're from anywhere else you'll be wondering what the fuss is about!

Note

All weights and measurements *must be given in metric units*, but don't worry full automatic conversion to imperial (American English) units is performed by the functions in the Wacs APIs used by all the tools. Whether or not these conversions should be applied is determined by a configuration variable - units in the layout section of the `wacs.cfg`. In a future release, we will add UK English imperial unit conversions and the ability for the user to specify their units of choice.

So why do we insist on metric values? Two reasons :-

1. Firstly, there is no consistent data type for feet and inches which would make writing a database query like “Show me models between 5ft 6ins and 6ft 0ins tall inclusive” a very complex matter. This is likely to cause no end of bugs in search routines that could be very hard to resolve, whereas using metric values the query becomes simply `between 167 and 183`. There are similar problems with weights where UK English would give the number in stones and pounds, whereas American English would give only pounds so it's not even a consistent set of units within the few countries that use it. And finally there's the fact that vital statistics are given in inches rather than feet and inches - what gives with that?

2. Secondly, it's actually a very small proportion of the world that uses the old imperial english measurements and even there it's on the way out. Schools have taught metric units for many years and the understanding of the imperial units is waning.

Tip

It's not that hard: take the height in feet, multiply by 12, add the inches figure and then multiply by 2.54 and round to the nearest whole number. Take the vital statistics in inches and multiply by 2.54 and round to the nearest whole number. For weights, take the figure in stones, multiply by 14, add the pounds and then divide by 2.2. For typical models: for heights you should expect a figure between 160 and 187, for weights a number between 50 and 75, and for vital stats figures between 60 and 90. It's quite easy to have a calculator open on your desktop to do the conversions.

Warning

Please take our advice on this. You really will run into a whole lot of trouble if you try to use imperial units in the database - we know, we tried, it broke (badly).

Chapter 7. wacsunpackmgr - The Unpack Manager

About Unpacking

Now that we've looked in some detail at the process of creating a model record, we're going to take a look at how to actually add content to a Wacs site. There are a number of ways in which new material can be imported into the system, and we're going to discuss some of them here. Once of the key concepts is that the process is divided into three distinct sections:

1. Unpacking - gathering the materials together in one place and checking them over.

2. Placement - deciding where they should go and what they should be called and getting them into the Wacs system.

3. Rating - Rating, cataloguing and marking appropriate attributes

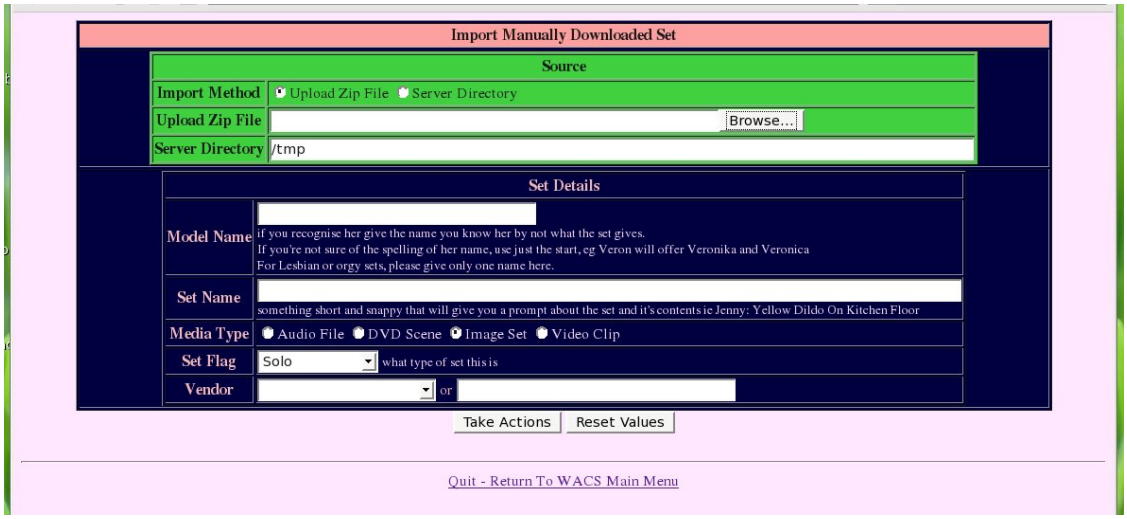
Each of these tasks is actually performed by a separate web application in the Wacs system which are known respectively as wacsunpackmgr, wacsplacemgr and wacssetmgr. In this chapter, we are concentrating on the first of these namely **wacsunpackmgr**. The primary task of **wacsunpackmgr** is to gather together all the materials for the set we're about to create - images, icons, video clips, descriptions; make a few notes on where it came from and which models are involved, and put it in a place where it can be previewed and picked up by the placement manager (**wacsplacemgr**).

Each administrator of the wacs system will be given their own unpack area within the download spool area of the Wacs system. While configurable, this is usually in `/var/spool/wacs/download/unpack/`*`username`*, so for instance beaky's unpack directory would be `/var/spool/wacs/download/unpack/beaky`. Having individual areas allows multiple administrators to import content at the same time, and allows people to take a break during the process (to confirm details perhaps) without troubling others. Any given user can only unpack one set at a time however.

Working With Image Sets

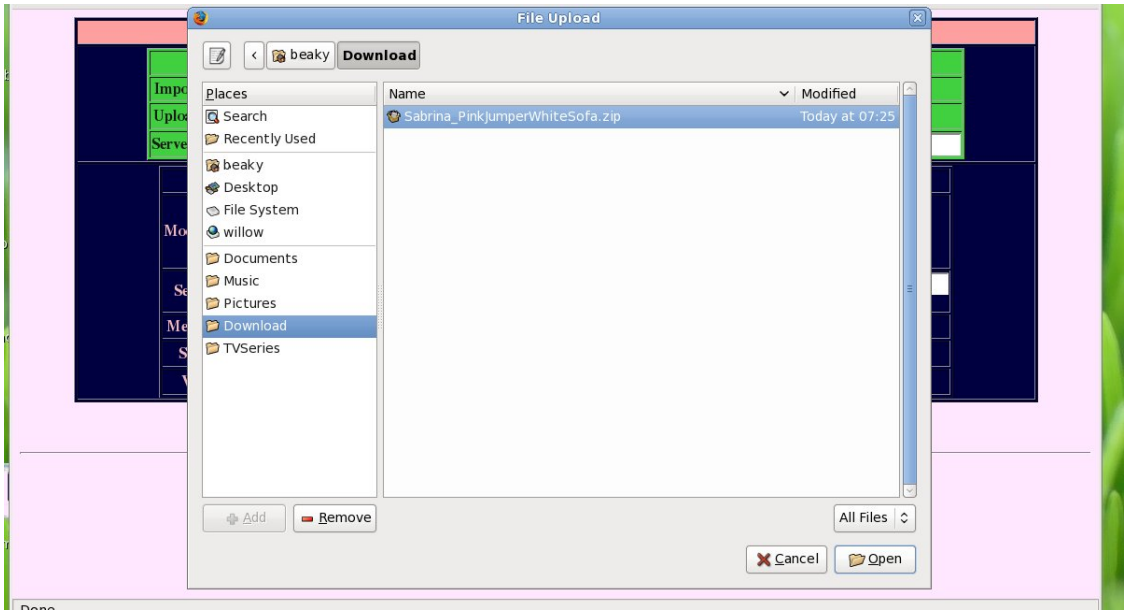
There are a number of ways of reaching the unpack manager, but the one we will look at first is that used for doing a manual set upload. For this you are assumed to have a zip file containing the images that comprise a single set accessible to your web browser. To reach the unpack manager, you select the option **Import set** from the **Maintenance Menu** on the Wacs Front Page. If you see the **Preferences** menu instead, then your account is not properly enabled for administrative activities - for more information on how to fix this see Chapter 2, *First Steps*.

When you initially get to the unpack manager, you'll see a page like this:

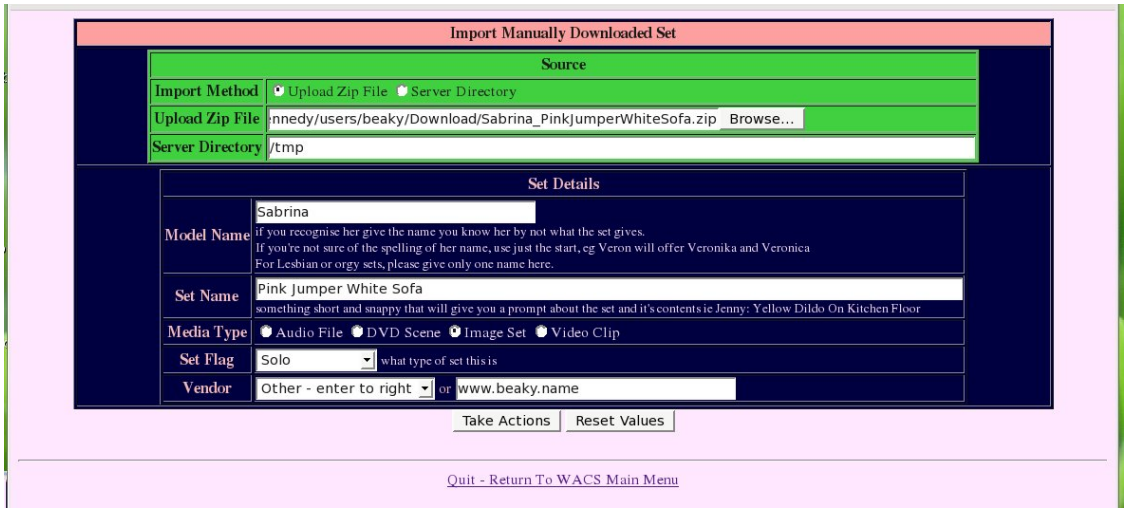


As you can see it offers us the choice of uploading a zip file via the web browser or importing named directory that is visible on the web server. Which of these you use really depends on how your system is laid out - if your Wacs server is on another machine and you're doing the collection administration from a desktop or laptop machine, uploading through the web browser is probably extremely useful. If it's all on the screen attached to your Wacs server, you can merely tell it to fetch the set from the directory by specifying the path.

Moving down below the green box, we have a box for the model's name - note that for a lesbian set this should be only one of the models - the “owner” of the set. Additional models should be added later in the process. We also have a short name for the set, which we'll see used as “the official title” for this set. Next we get to specify the media type and the type of set that it is. Finally we get the choice to specify where it came from, either by pop-up menu of known vendors or by typing it in directly.



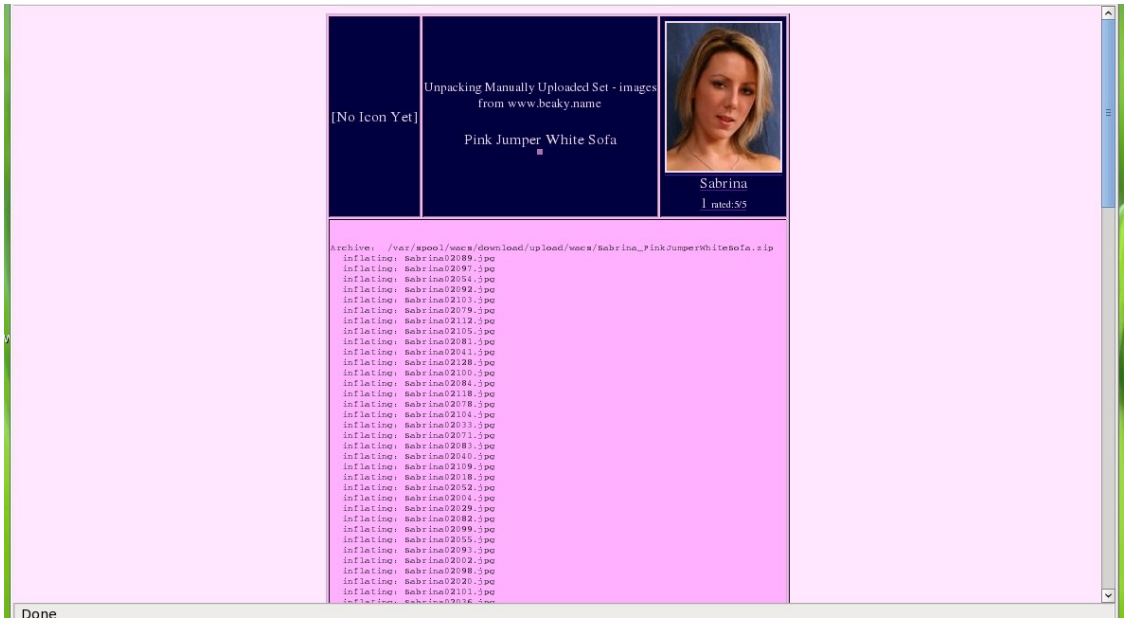
This screenshot shows us browsing for the zip file we're going to upload - we got here by specifying **Upload Zip File** as the *Import Method* and by clicking on the **Browse** button at the right hand end of the box entitled *Upload Zip File*. Once we have selected and openned this file, we should see:



You'll see that we've also filled in the details about the set, we've said the model is *Sabrina*, we've called the set *Pink Jumper White Sofa*, we've specified that it's an *image set* and that it's *Solo* in nature. Finally we've said that it comes from a site that it doesn't already know about called `www.beaky.name` (which it sorta does except it's not actually available there yet...). With these details filled in, we click on **Take Actions** in order to proceed to the next screen.

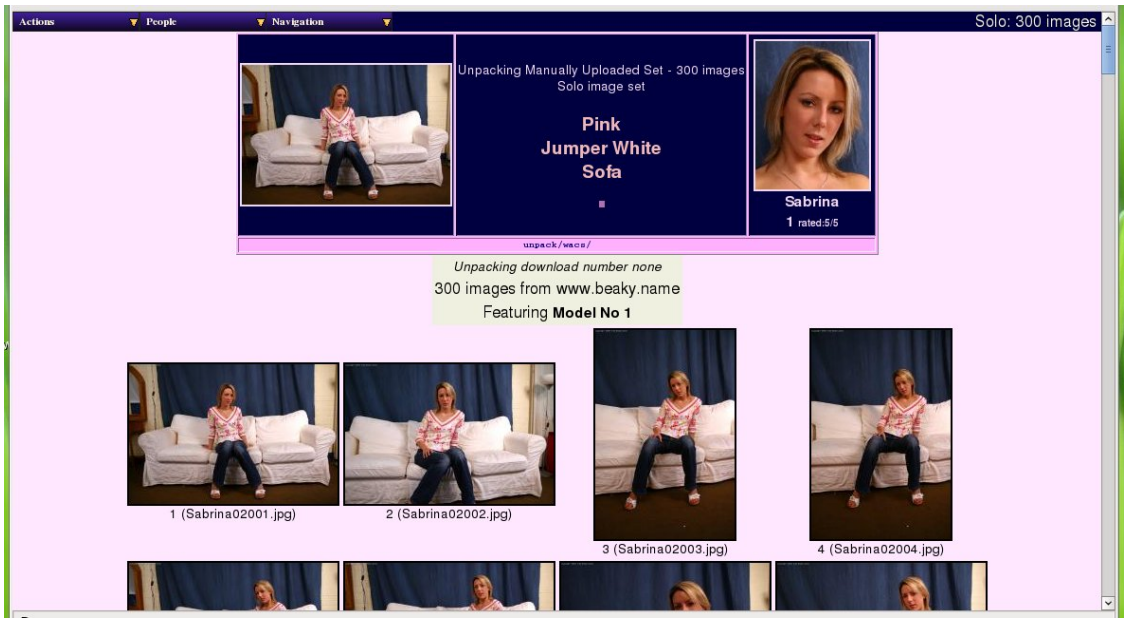
At the top of the next screen, **wacsunpackmgr** has summarised what it has so far. It's told us the name of the the zip file it's uploaded, it's size and date created, and offered us an option on whether we want the file archived or not. This bit in the green box is about the set, while the blue box below is about who is in the set.

Here it's taken the name Sabrina and searched for it amongst the models and discovered that model number 1 is called Sabrina and so it's offering us two choices - accept model number 1 as being the model in this set, or to tell it that it's not model number 1 who features in this set. Had we have had multiple models called Sabrina, it would have offered us a choice of all of them or none of the above. Anyway in this case, this is the model we want so we simply click on **Choose This Model** in order to continue.



Having duly collected all the necessary information, the Wacs Unpack Manager is finally doing it's namesake activity and actually unpacking the zip file it's been given. We're actually going to show you two screenshots of this page; this one shows the start of the unpacking process, while the second one below shows the bottom of the screen where the unpack process has completed (hopefully successfully...). At this point, all of the unpacked images should be in the unpack directory for this administrator in the download area. There is actually also a hidden file present in that directory called `.unpack` which passes across the information we entered about the set itself, the model featuring in it, and so on.

This is the bottom of the final page of the unpack manager session and shows you the steps you can take next. One thing that we have tried hard to do is to make as many as possible of the standard Wacs collection tools know about a special set called *set 0* which is that particular administrator's current unpack directory contents. This means you can browse it and look at it almost as if it were a normal set. The top two links at the bottom of the unpack manager output take you to the standard set browsing pages in either whole set or paged mode. Below we see the set we've just unpacked in the whole set page.

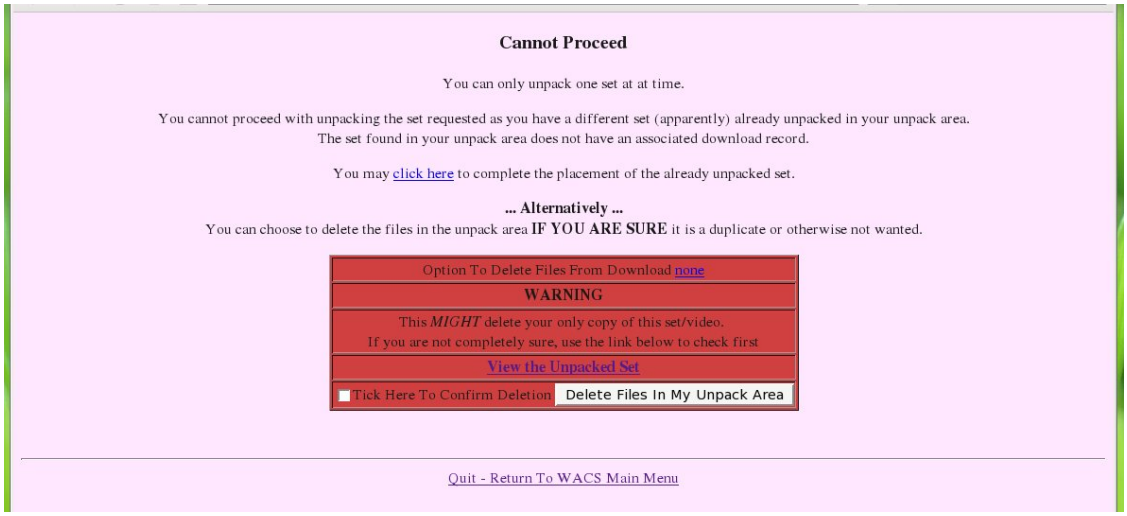


This provides us with the opportunity to browse the set, determine the location, action and clothing that we're going to use in making up our description of this set. Once we've had a good look, we can simply return to the unpack manager's last page and click on the third option to go on to the placement manager.

This completes our initial look at using the unpack manager for image sets; we'll just cover the topics of what happens when you already have and unpacked set and those aspects that are unique to videos and then move on to look at the placement manager in the next chapter. We will encounter the unpack manager again when we look at the download mechanism (See Chapter 13, *The Download System*).

Already Unpacked Warning

We said earlier that each administrator, ie you, can only unpack one set at a time. Obviously there are times when you forget this or some kind of failure occurs and things are left in a messed up state. If this happens, you can expect the unpack manager to present you with a screen something like this:



Working With Videos

The unpack manager works quite similiarly when working with video files except in so much that it typicallty only uses two files: the video file itself and the main icon file. If you don't already have an icon file, you can use a tool like **xine** or **mplayer** to play the start of the movie and save a suitable screen capture. This file should then be resized down to a more reasonable icon size and saved with the same name as the movie file itself but with a suitable image file extension rather than the video one. Therefore if your video file is called `Roxanne-KnittedTopWhiteSkirt.wmv`, then the icon for it should be called `Roxanne-KnittedTopWhiteSkirt.jpg`.

Chapter 8. wacsplacemgr - The Placement Manager

The Placement Process

As we discussed in the previous chapter (Chapter 7, ***wacsunpackmgr*** - *The Unpack Manager*), sets can be imported into Wacs in a number of ways and then end up in a holding area for each administrator where they can be viewed, have icons made for them and so on. Once they are ready to be installed into the main Wacs system, another application comes into play called the placement manager. In this chapter we will look at how to use the placement manager to place a set and complete it's importation into the Wacs system. As you will have seen, once the unpack manager has done it's jobs, it provides links to viewing the set and to the placement manager.

When you initially call up the placement manager for a set, you'll see a web page something like this. The top portion of it is a standard page masthead you will have seen in a number of places around the Wacs system. Since in the unpack manager we both identified the primary model and gave the set a working title, these are displayed here as you might expect. The first group of entries are the various components of the set name - the model's or models' name(s), a description of their clothing, a description of the location and action in the set and the general type of set that it is.

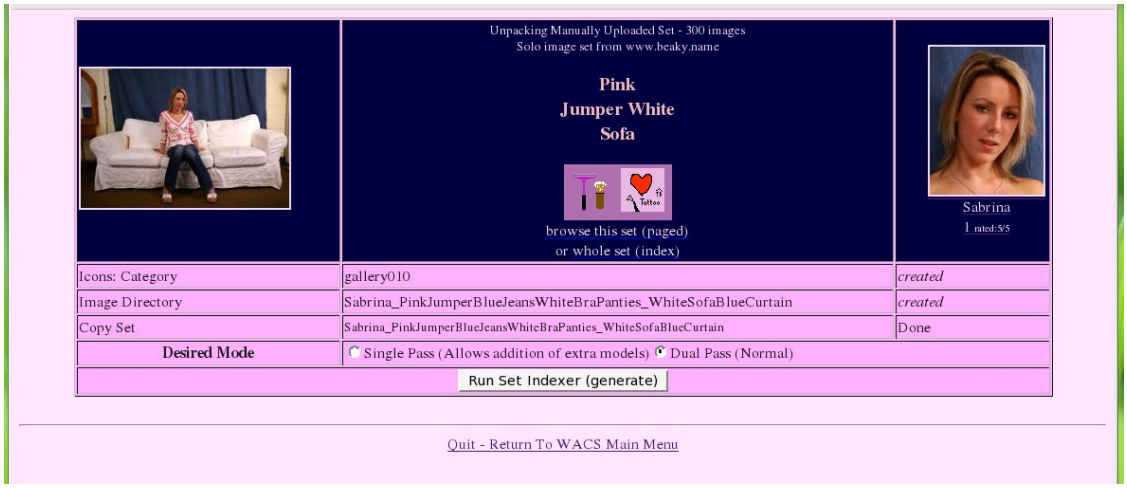
As you will hopefully recall from Chapter 4, *Naming Sets In WACS*, there are a number of special words that will be used in determining extra information on the set. We will cover this topic again in more detail in the section called “Keyword Manager” in Chapter 11, *Other Web Based Tools*. For now you can just describe the set as best you can and add the mark-up later. As you become familiar with the keywording system you can both tailor it to your needs and get used to what keywords will trigger what markup attributes. While you can use spaces here between works, please do remember to capitalise the first letter of each word of your description.

The lower section of the placement manager form covers where the set is to be placed within the appropriate Wacs media tree and therefore which gallery or models section it will be placed in. Additionally there is

the option to add the new set once created to an saved search set (aka tag set). In all cases, the placement manager will try it's best to select reasonable defaults for these values to save you as much typing as possible!

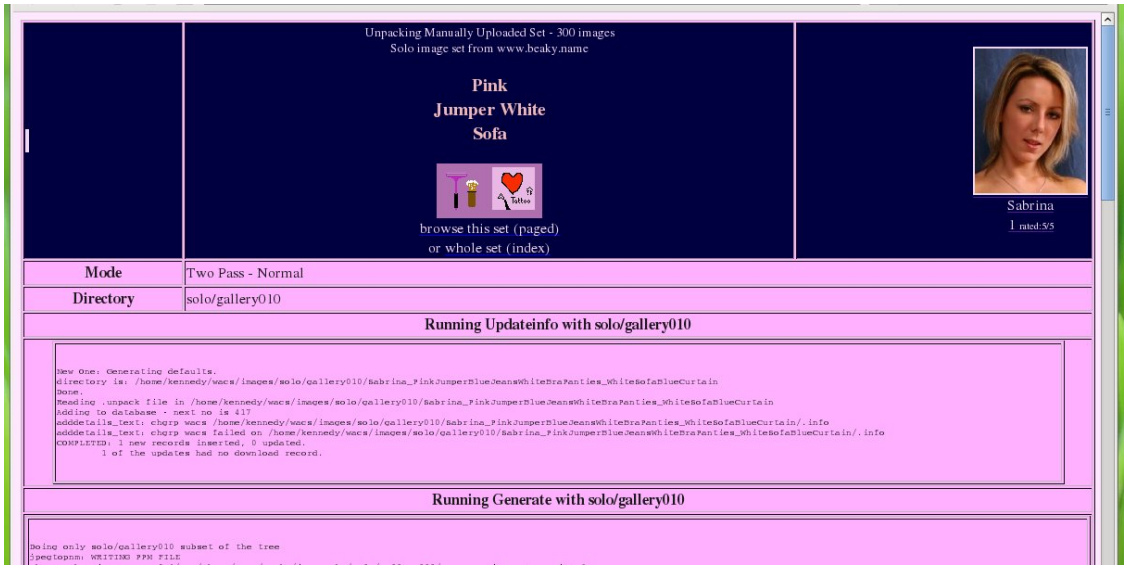
In the above screenshot, we've added to the defaulted values by adding `Pink Jumper Blue Jeans White Bra Panties` to the desciption of Sabrina's clothing, and `White Sofa Blue Curtain` to the location and action sections. The automatic defaults of `Sabrina` for the model's name and `Solo` for the set type are fine in this case and thus left unchanged. Similarly the defaults of `solo` and the next available solo gallery slot in `gallery010` are also fine in this case. We've decided to leave the saved search (tag) number blank in this case, so it won't be added. Once happy, we click on **Confirm Details** and we move on to the next screen:

This screen summarises the details we've entered and shows us the name it's created using the various pieces of information we've given the placement manager. If you're not happy with any of these values, just click on your browsers back key and re-enter. Remember that if you do this, you will then need to click on **Confirm Details** to get it to re-assess the new inputs rather than using the web browsers forward button to return here. If everything is in order, click on **Confirm Placement**.

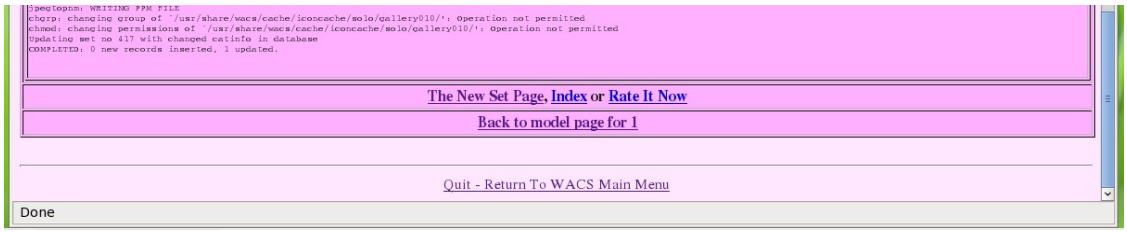


At this point the placement manager will attempt to create the new directories and move the files out of your unpack area and into their final destination. If there are problems with file space or permissions, these should be reported here and you'll have various options on how you resolve the problem. In many cases, the simplest approach, if you still have the zip or video file elsewhere is to use the unpack manager to delete the half unpacked set and start again from the beginning. Assuming all goes well with the placement, the files will have been transfered to the Wacs media tree. However, they will not yet be indexed by the Wacs system - that doesn't happen until either the **updateinfo** or **generate** commands have been run and so the placement manager gives you the option to run these right away from the web browser. In most cases, you simply choose **Dual Pass (normal)**. The other option is used for Lesbian and Group Orgy sets where you can identify another model featured within the sets. As we don't currently have any lesbian sets we have the rights to show, it's kinda hard to illustrate that at the moment. We hope to resolve this shortly...

Clicking on **Run Set Indexer (generate)** will invoke the command line based updateinfo and generate programs which will create the new set in the database and run the various keyword search schemes to determine as much of the set description attributes (metadata) as it can from the descriptive text you gave.



When you run the set indexers, they will also consider any other sets in the same gallery while they're creating the new set. This ensures that the gallery is freshly *preened* before people are attracted to it by your newly added set!



Once updateinfo and generate have completed their work, you'll be presented with a number of options for the next action. You can rate the set, view it or see it's info page or go back to the model's page. We'd recommand rating it at this point, and clicking on the **Rate It Now** link will take you to the Chapter 9, *Wacs Set Manager*....

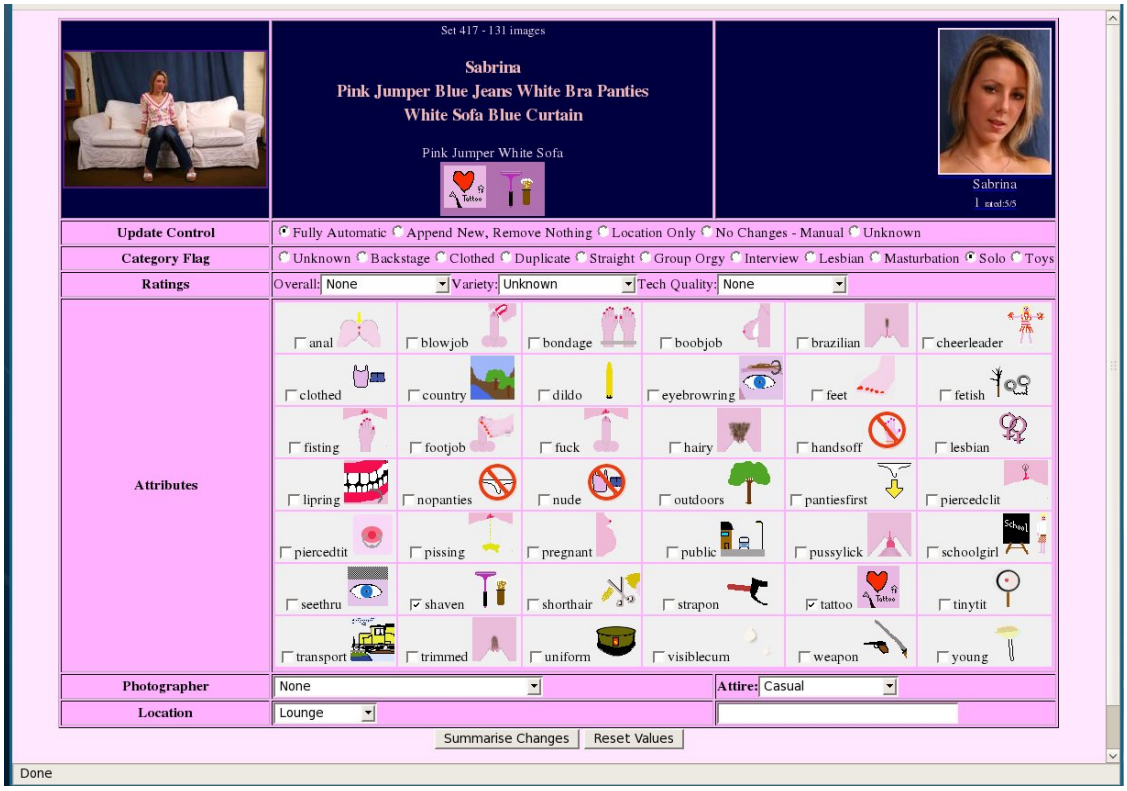
Chapter 9. Wacs Set Manager

Meta Data Manipulation

One of the major features of Wacs is it's ability to catalogue and organise a collection and find things again using a wide variety of search criteria. In order to do this effectively, sets need to be marked with appropriate attributes; part of this is done via the keyword system described in previous chapters (See Chapter 4, *Naming Sets In WACS* and Chapter 8, ***wacsplacemgr*** - *The Placement Manager*). However it is also possible to do a lot more buy directly editing the information about the sets - what is known in technical terms as *the meta data*. Wacs has two tools that allow you to directly edit the meta data for sets, the set manager and the info manager. This chapter covers the set manager and the following chapter (Chapter 10, *The Info Manager*) covers the info manager.

The basic distinction between the two is that the set manager handles data about what the set contains, while the information manager handles information about the set itself. The distinction is a little blurred in places, but that's the general rule of thumb.

What You Can Edit



Here we have a screen shot of the set manager working on our newly added Sabrina set. There are seven basic parts to the set manager display: the masthead (summary of the set details), Update Control, Set Type, Ratings, Set Attributes, Photographer/Clothing and finally Location. The masthead is common, but the second of these is a bit more complex.

As we saw in the previous chapter, the placement manager runs a command called **updateinfo** which does all of the keyword searches and importation of model attributes to kickstart the attribute marking process. Each time it visits a directory, it will attempt to update the meta data if it can - mostly this is a good thing because if the set name includes a keyword that was not recognised first time (for instance if it's been added since the last run), the new attribute will be added to the set. However, there are occasions when that system doesn't work as desired and it's necessary to be able to override the automatic guessing process. This is done through the **Update Control** flag - there are three commonly used modes - **Fully Automatic** where the attributes are changed automatically to whatever the new keyword scan produces; **Append New, Remove Nothing** where newly added keywords will get their attributes added but none of the previous entries will be removed, and **Location Only** which only allows changes to the media file location details. Of these **Fully Automatic** is the default, **Append New, Remove Nothing** is selected as soon as you add additional attribute information to a set using the set manager; and **Location Only** will not update any of the attributes. You would typically use **Location Only** or **No Changes - Manual** where something was giving a false positive on a keyword search. An example of this was a model called `River` who's name triggered the `country` and `outdoors` keywords. These flags of course make sense normally, just not in her case.

As we continue down, the next attribute section is the options for the various types of set available - hopefully this is reasonably straight forward and we've discussed it several times before. The next one down is for the ratings - again we've looked at these values before - see the user guide and schema reference section of the programmers guide for more details.

Now we reach the big section - all the possible mark-up attributes for sets. Hopefully all of these are reasonably self-exaplanatory and again we have discussed these before, particularly in the user guide. The next section has the Photographer and the Attire pull-down menus - the photographer list includes all currently defined photographers. Photographers get incorporated into the Wacs system in two ways; either by being imported from the supplied initialisation XML file when you create the database tables (see the installation manual for more info, and the entry here photographers.xml) or by being created using the photographer manager which we will be looking at in a later chapter (the section called “Photographer Manager” in Chapter 11, *Other Web Based Tools*). The attire settings are a little bit more complex as it basically offers you a choice of any the existing options that have already been used. There is of course a slight problem here in that options won't be listed here until they've been used at least once, *BUT* we've done our best to ensure that there is at least one keyword that will match each of our standard wordings for the attire value. If there are new words you want to use you can either:

- add a new keyword using the keyword manager that will give you that attire value you want
- set the attire manually using SQL for the first set after which point it will appear in this menu

An example on how to do this latter option is given in the section called “Adding A New Type Of Attire” in Chapter 16, *Simple Tasks In SQL*. The final values relate to the location and detailed location fields, which again are usually derived from the keywords database. As before you can define new locations by either editing the keywords to add a new one, or by manually adding one entry to the sets database using SQL whereafter it will appear in the pull down menu.

Doing An Update

We're going to look at using the set manager to make an update or two to our newly added set of Sabrina on the white sofa. We're going to rate it and modify the attributes to reflect the fact that Sabrina has a Brazilian shaved pussy rather than fully shaven in this set. To do this we select the three ratings - for overall we've choosen **3 - Good** , for variety **2 - Cute Twist** as there's a couple of neat moments when she's sitting up on the back of the sofa and **3 - Good** for technical quality as it's taken in a well equiped studio with a range of camera angles.

Having selected our ratings choices from the pull down menus, unchecked the tick box next to **shaven** and ticked the box next to **brazilian**, we're ready to make the change so we click on **Summarise Changes**.

Here we see the summary of the changes we're proposing to make, along with an additional indication that because we're made a manual change which would not be picked up by an automatic rebuild, the set's update method has been altered from **Fully Automatic** to **Append New, Remove Nothing**. If we then click on **Commit Changes** we come to the final screen where it confirms the updates made.

As you can see here, it has now updated the database entry for this set with these changes. It then offers us a selection of links to things with might wish to do next, including updating the information for this set which takes us to the Wacs Information Manager and the topic of the next chapter...

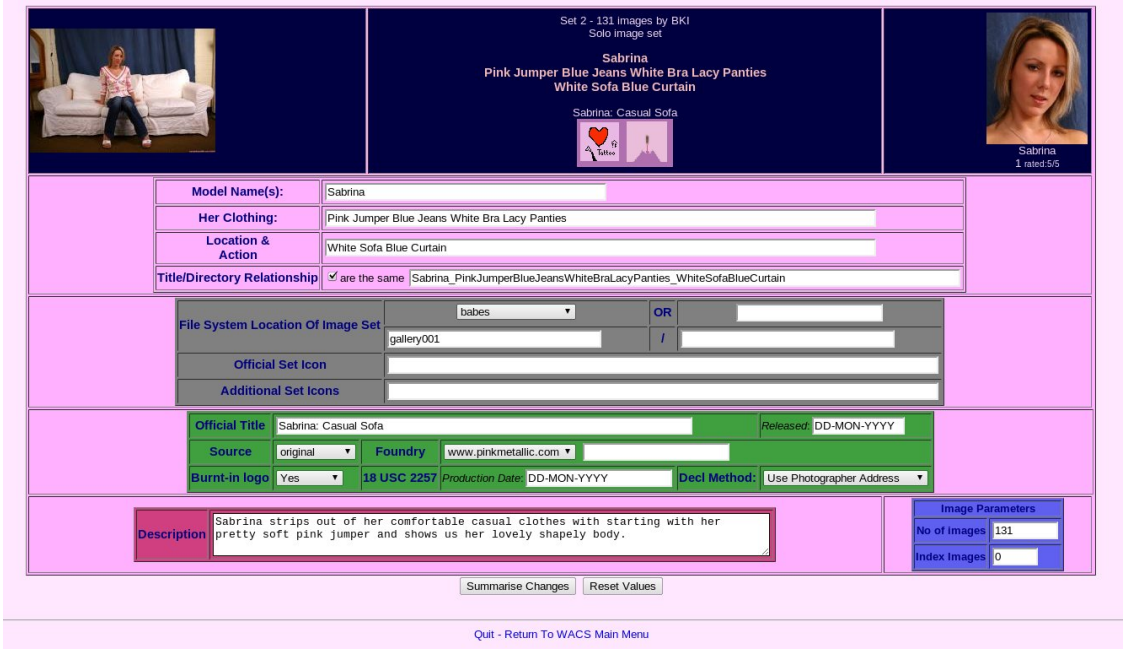
Note

There are other links here allowing you to view the set again, add another model to it and to add it to a connection. We'll cover these topics in later chapters.

Chapter 10. The Info Manager

Managing Additional Information

In the previous chapter we looked at **wacssetmgr** (See Chapter 9, *Wacs Set Manager*) and how it is used for changing set type, assigning ratings and attributes, and for selecting the attire and location attributes for a set. This however doesn't begin to cover all the information we need to hold about the set and you might have noticed that it didn't include any way to rename or move a set. There is a second web application that handles all of that and since it relates to information about "a set" as opposed to what is featured within the set, we call it the Set Information Manager or **wacsinfomgr** .



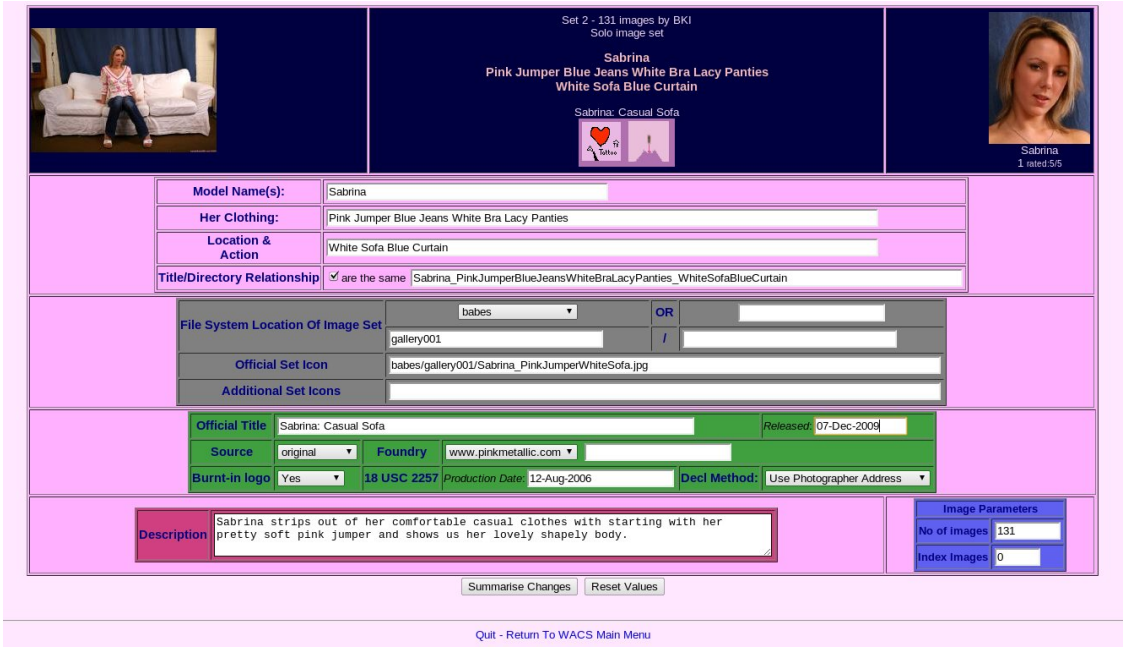
The set information manager also features the standard masthead just so we know what we're talking about. Below that it has five boxed areas of information. The first of these returns to the three components of a set title - the models, the clothing, the location and action - plus an indication of the actual location (or filename in the case of a video) - there is also a tick box to indicate if the two are directly related to each other. For image sets it is likely that they will be, for videos much less so as the video file name is often a lot shorter. When the tick box is ticked, changes made to the description will be automatically changed in the directory/ filename box. If you try and change both independantly, have the box ticked and don't make them the same, you'll get an error message. Generally just change the description and **wacsinfomgr** should do the right thing.

The second box section (the grey one) covers the actual location within the media tree of the set and any official and additional icons attached to it. Changing the settings in here will cause a relocation of the set itself if that makes sense. The third box (the green one) covers the official name of the set, where it came from and how we got hold of it, and allows us to flag it as having a burnt-in logo and provide details of how to derive the 18 USC 2257 declaration for those people required to provide it by US law. It also now provides the ability to set the 18 USC 2257 production date and the date of release.

The fourth box (the red one) allows you to edit the long form text description of the set - this is entirely optional but it's presence is indicated by a small information *i* symbol on the model page and image list

pages and it is shown in full on the set info page where it has been used. The final box (the blue one) allows editing of the image count details for image files and the duration and aspect ratio for movie files.

Example Of Using wacsinfomgr



Here we're just going to walk through a simple example of using the **wacsinfomgr** to modify a few details of our Sabrina set on the white sofa. We're going to do three things: set the production date to `12-Aug-2006`, set the release date to `07-Dec-2009` and add an official icon at `solo/gallery010/Sabrina_PinkJumperWhiteSofa.jpg`. The first step is to enter these values into the screen as shown above and then click on the **Summarise Changes** button.

At this point we get the usual confirmation summary letting us know what **wacsinfomgr** is about to change. As usual, check it over and when happy click on **Commit Changes** button. As before to correct a value use you web browser's back key, but do make sure you use the **Summarise Changes** button to return to the confirmation screen with the revised values.

Here we see the confirmation of the changes made and the confirmation that they have been saved to the database. If set moves were involved here you would see the move process taking place and the generate command being re-run to re-index the new set, generate a new thumbnail and re-scan the titles for newly added keywords. As usual, a range of links to other related options are offered on completion of the update.

Note

With release 0.8.6 we believe we've now fixed the outstanding issues with the automatic relocation of icons when their location is changed. If you find a bug in this area, please report it.

Chapter 11. Other Web Based Tools

In the last two chapters we have been looking at **wacssetmgr** and **wacsinfomgr** and how they can be used to update and modify key parts of the Wacs database for sets. In addition to the high profile database tables for sets and models, there are a number of other database tables that affect the operation of a Wacs system. Many of these have their own tools for administering and updating them - in this chapter we will provide a quick overview of those tools.

What's Available

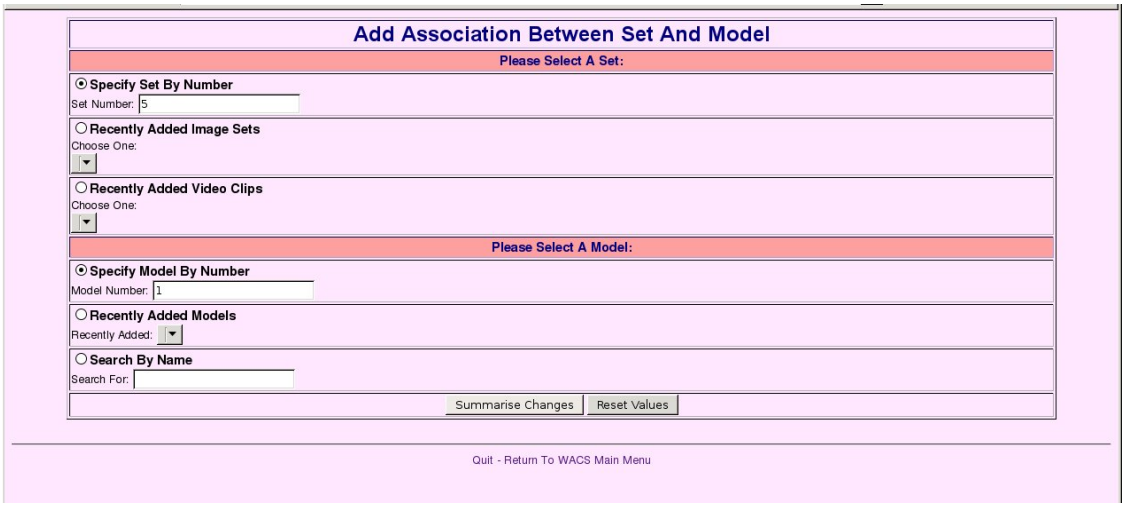
In this chapter, we're going to be looking at the following tools:

Application	Table	Notes
wacsaddassoc	assoc	Adds a new ad-hoc link between a model and a set (image or video).
wacsconnmgr	conn	Adds a set (or a model) to a connection - a way of keeping collections of sets together, often uniting those of a similar theme to each other.
wacskeywordmgr	keyword	Updates the keyword to attributes, attire and location associations
wacsvendmgr	vendor	Updates details of source sites, cross-link sites, etc.
wacsphotmgr	photographer	Updates the list of known photographers

All of these tools are listed on the **Maintenance Menu** which appears on the Wacs front page when your account has the status of administrator (see Chapter 2, *First Steps*). The Add Association (**wacsaddassoc**) and Add To Connection (**wacsconnmgr**) commands are also offered on the Administrators version of the Actions menu on all set pages and on the completion pages of both **wacssetmgr** and **wacsinfomgr**. In these cases the set number is automatically passed across and you only need to provide the other parts of the information needed.

Association Manager

The **wacsaddassoc** command allows the addition of an artibitary association between a model and a set. Normally adding the set to the WACS system links it in with a model, so most often this command is used to add additional participant links to a set (image set or video clip) of a lesbian encounter or a group orgy. It can also be used to add links at a later time to a set that was initially created without a model link.



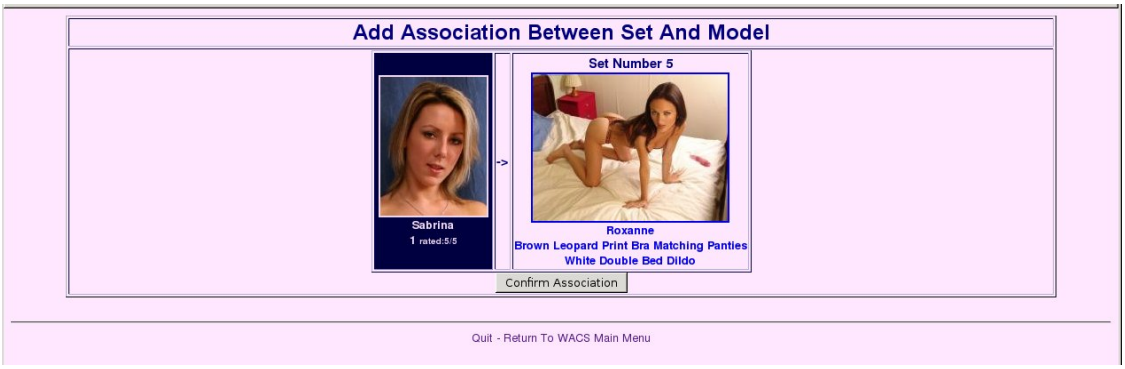
As you can see from the above screen shot, **wacsaddassoc** allows you to use three methods to select the set and three methods to select the model. The choices for selecting the set are:

- Enter it's set number
- Choose from a pull-down list of recently added image sets
- Choose from a pull-down list of recently added video clips

Similar choices are available for models with options of:

- Enter the Model number
- Choose from a pull-down list of recently added models
- Search for a model by name

Once you've made your choices, and remembered to click on the corresponding radio button to show which choice you're making, click on the **Summarise Changes** button to see confirmation of the selected model and set as shown below.



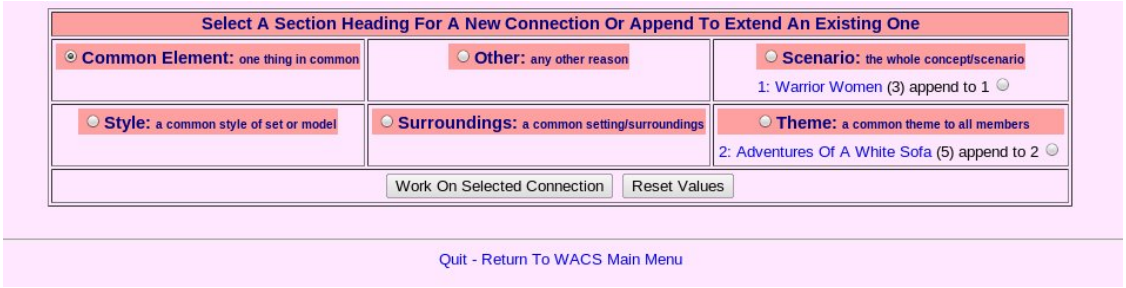
If they association you're trying to add already exists, you will be warned about it at this point - otherwise you'll be able to click on **Commit Changes** to perform the addition of this association. Since we don't have any lesbian sets yet in our demostration site, the change we're showing doesn't actually make any sense on the sets we're showing, but it demonstrates how the **wacsaddassoc** application works.

This is the confirmation screen you will see once the association has been successfully added.

Connections Manager

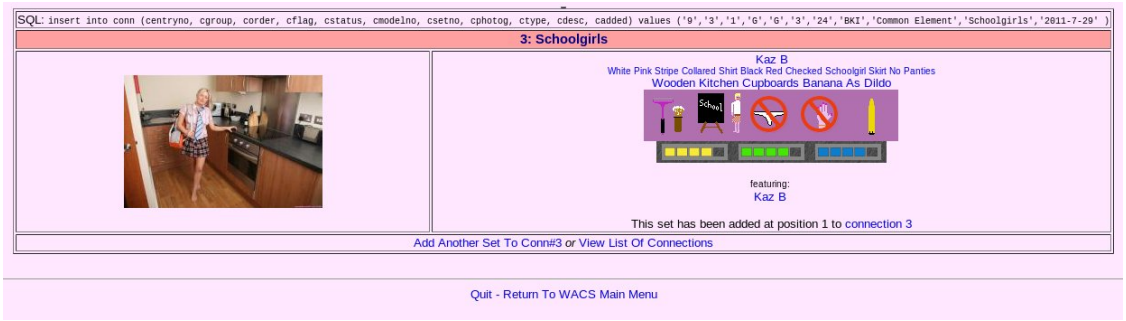
This is the second of the web based administration tools we'll be looking at in this chapter. Use of this one is entirely optional - they're intended for content promotion, highlighting and blog commentary activities by allowing a group of sets to be grouped together. We think connections are a useful extension to the WACS system and are very similar in operation to saved searches except that they are exclusively managed by the system managers and are permanent. Where a set is a member of a connection, this will be highlighted whereever that set is shown in an index. The normal way is to use them to tie sets featuring totally different models together around a common theme.

This shows choosing the `Add xx To A Connection` menu option in the Actions menu from the set page in administrator mode. This calls **wacsconnmgr** passing it *set24* which it will then work on.



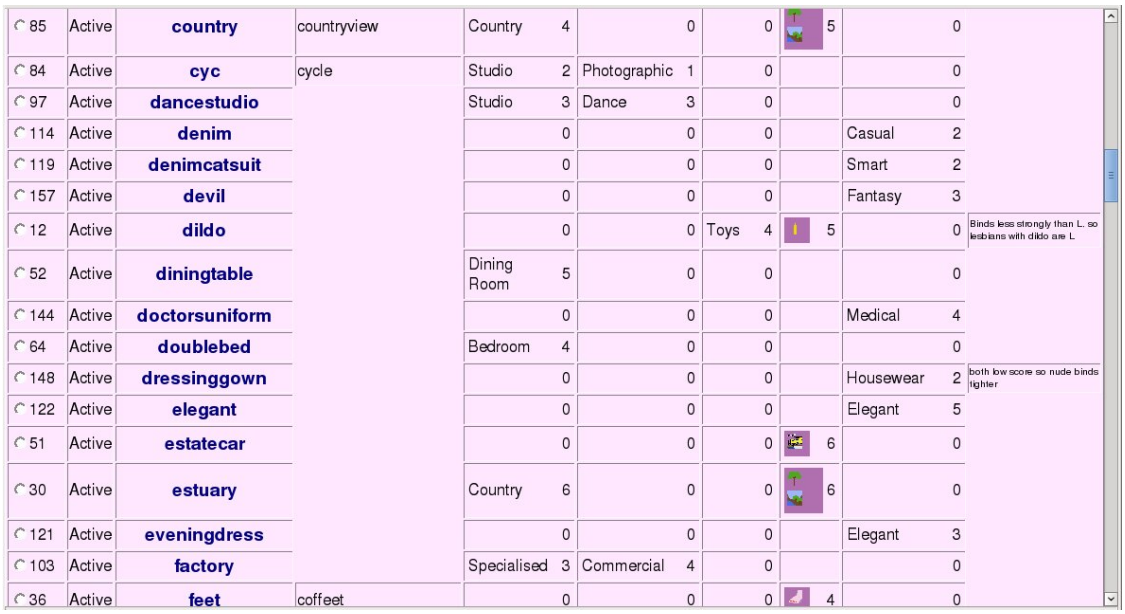
Here we choose Common Element as the nature of the connection we're creating as we're going for *Schoolgirls* as the theme for this connection. If other connections had been defined already, each one would be listed under it's category with it's basic details (number, name, number of sets) along with a button to select it. When we select an existing connection, **wacsconnmgr** and it already knows the *target* , it'll jump straight to the confirmation page below. Once you've either selected an existing connection or clicked on the button in the title bar to create a new one of that type, it's then a click on the button marked `Work On This Connection` and we get the next screen below:

Here we've already got the set number of 24 already filled in, so we just need to give it a description. In this case we're calling this connection simply *Schoolgirls* so we type that in. Then click on `Add This Set To Connection` and you should get this confirmation page.



Keyword Manager

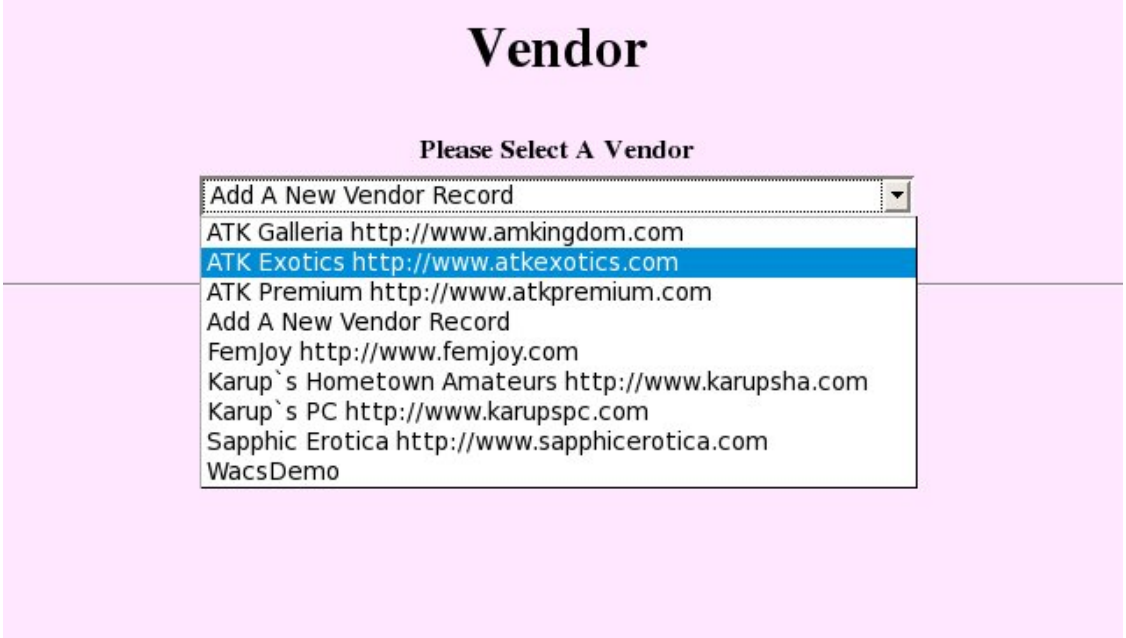
The third of the web based administration tools we're going to look at in this section is the keyword manager. This allows the examination of the pre-defined keywords that are loaded into the Wacs system during database creation along with the editing of entries, creation of new entries and even the deletion of ones you don't want. We discussed the basic concept behind this in Chapter 4, *Naming Sets In WACS* along with some examples of how it works.



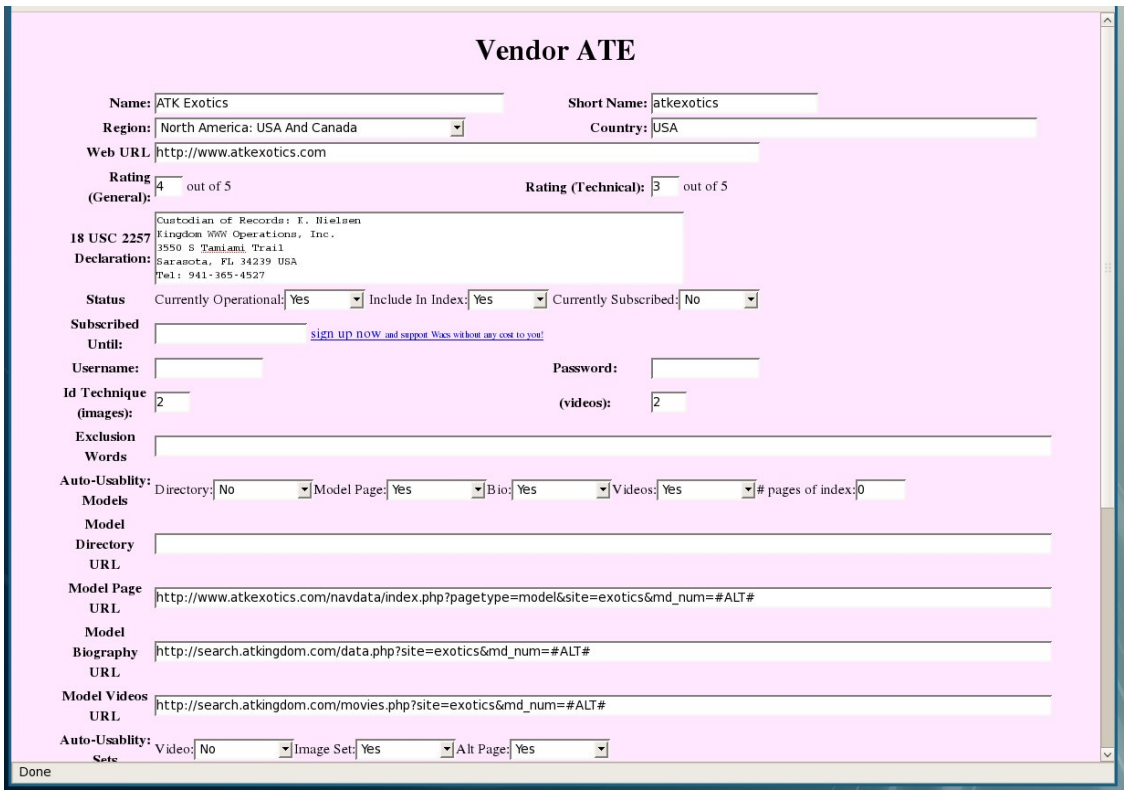
85	Active	**country**	countryview	Country 4	0	0	5	0		
84	Active	**cyc**	cycle	Studio 2	Photographic 1	0		0		
97	Active	**dancestudio**		Studio 3	Dance 3	0		0		
114	Active	**denim**		0	0	0		Casual 2		
119	Active	**denimcatsuit**		0	0	0		Smart 2		
157	Active	**devil**		0	0	0		Fantasy 3		
12	Active	**dildo**		0	0	Toys 4	5	0	Binds less strongly than L, so lesbians with dildo are L	
52	Active	**diningtable**		Dining Room 5	0	0		0		
144	Active	**doctorsuniform**		0	0	0		Medical 4		
64	Active	**doublebed**		Bedroom 4	0	0		0		
148	Active	**dressinggown**		0	0	0		Housewear 2	both low score so nude binds tighter	
122	Active	**elegant**		0	0	0		Elegant 5		
51	Active	**estatecar**		0	0	0	6	0		
30	Active	**estuary**		Country 6	0	0	6	0		
121	Active	**eveningdress**		0	0	0		Elegant 3		
103	Active	**factory**		Specialised 3	Commercial 4	0		0		
36	Active	**feet**	coffeet	0	0	0	4	0		

Vendor Manager

The vendor manager is a very simplistic update program that is basically used to add and update new sites so that they can be referred to in download records and identity maps. Typically we give each site a short name, usually all in capitals by which we can refer to it. A number of example records are added as part of the database initialisation process. As you create new vendor entries, we would very much appreciate your sending them to us for inclusion in future Wacs releases.

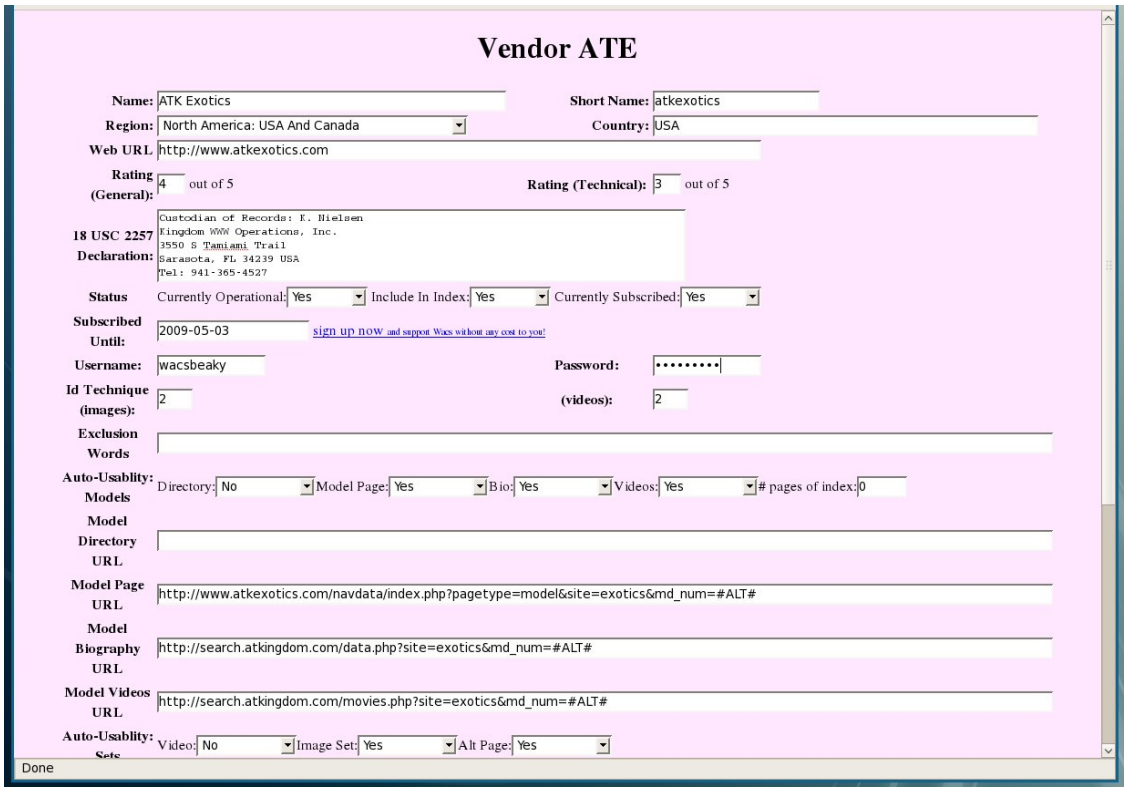


The first screen of vendor manager shown here merely lets you choose between modifying the existing entries, or adding a new entry of your own. In the example, we're selecting ATE otherwise known as *ATK Exotics*. Once we've selected this option, we click on the **Submit Query** button and see this screen:



As you can see, this is a pretty long and complex screenform which includes a lot of fields. The vast majority of them are related to the automatic download system and consist of templates to help it recognise the various different types of link you're likely to find on a site and how to interpret them to find various

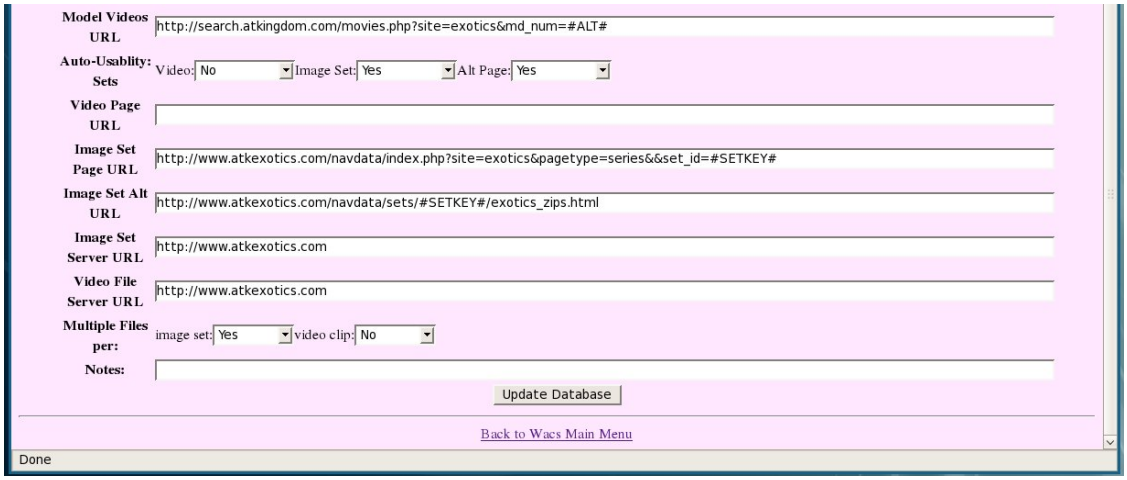
things like biographies, photosets and video clips. For simply using this for identification purposes, the vast majority of these fields are simply not needed and can be left blank.



As with most of the other web applications, we're just going to walk through the process involved in making a change so that you're familiar with the dialog and flow of using this tool. In this case, we're telling Wacs that we have an active subscription to this site, what our username and (fictious) password are, and when the current subscription period will end. This will basically enable the download system to log into this site and update it's information on the models we have that we have told it come from this site.

Note

The download system is in desperate need of a re-write and will be recieving significant attention in the near future. Do expect aspects of this interface to change significantly.



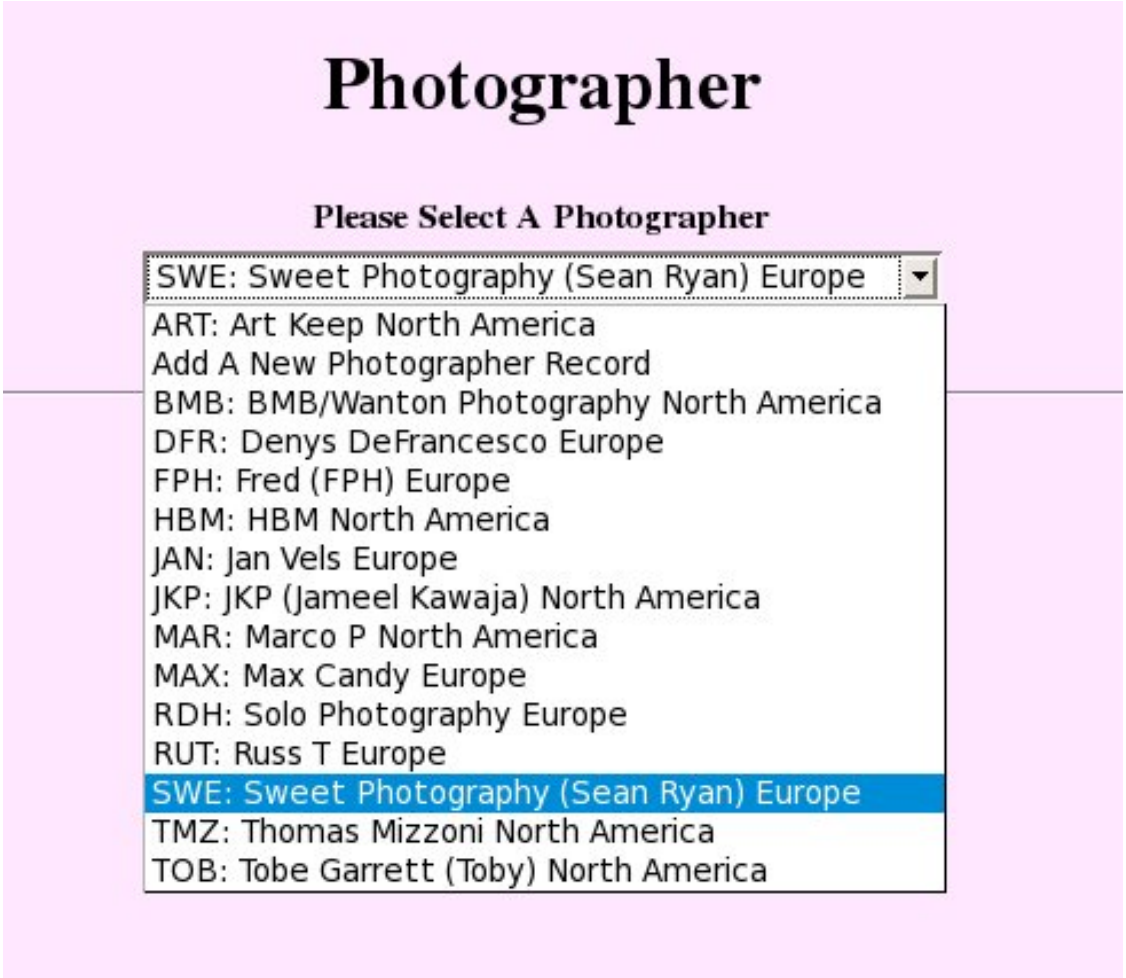
This screenshot shows the lower half of the same form including the **Update Database** button. Note that this is one of the few applications that doesn't seek confirmation of it's changes first - it just goes ahead and makes them anyway.

And this screen shot shows you the confirmation that the changes you requested have been made.

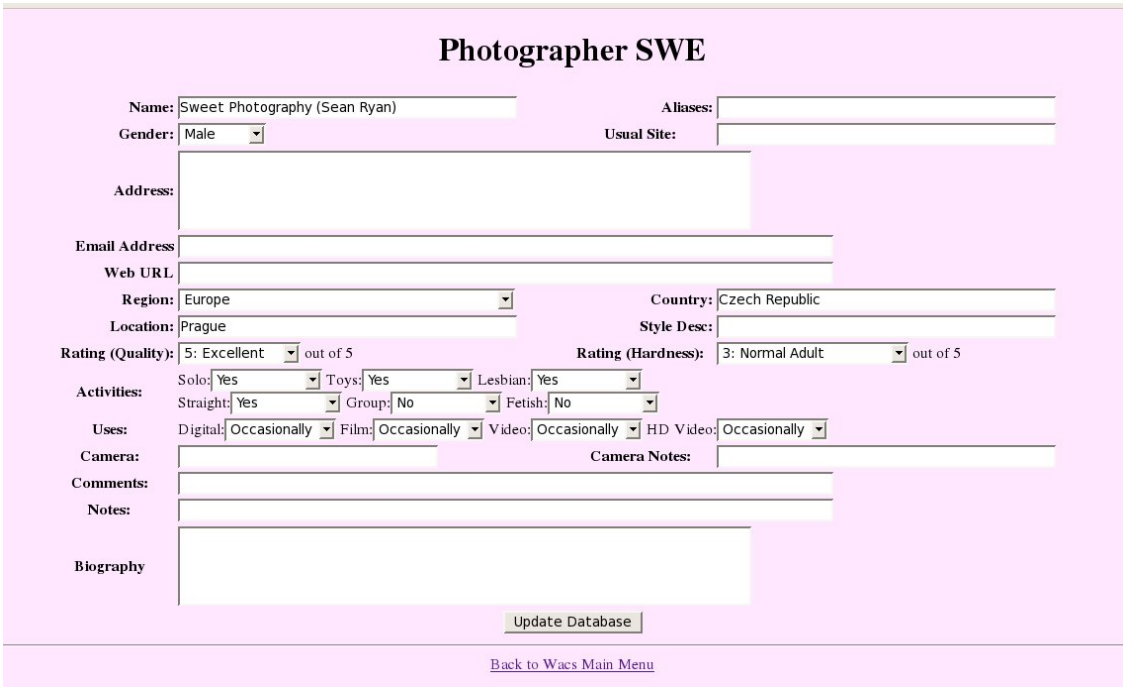
Photographer Manager

The aspect of determining the photographer who took a set is one of the more tricky subjects which Wacs seeks to tackle. Outside of the AT Kingdom group (AMK, ATE, ATKP and so one) and a few sites that pride themselves on their photographic prowess (like Met-Art and Femjoy) or are specific to a single photographer (like EuroNudes), relatively few other sites explicitly identify the photographer which is a huge shame. Those of us with significant exposure to what is out there will however come to recognise the style and favourite locations of certain photographers to the point of being able to make educated guesses as to who took the photos. One of our hopes for the future projects in exchange of Wacs meta information is that we can share these guesses and thus provide photographer cross references across a number of sites. We preload a small number of photographer details into the Wacs database, mostly those where we have made a positive identification of their work on more than one web site.

Here our worked example is looking at the work of Sean Ryan, aka Sweet Photography, who is prolific, very recognisable in his style and standard wardrobe and very good (in our opinion). Beaky also particularly likes his taste in models!



As usual we select the desired subject from the pulldown menu and click on **Submit Query** in order to retrieve the record. In absense of any better scheme, we've choosen to adopt the scheme used by the ATK sites of using an appropriate group of three initials in capitals to identify photographers.



Photographer SWE

Name:	Sweet Photography (Sean Ryan)	**Aliases:**	
Gender:	Male	**Usual Site:**	
Address:			
Email Address			
Web URL			
Region:	Europe	**Country:**	Czech Republic
Location:	Prague	**Style Desc:**	
Rating (Quality):	5: Excellent out of 5	**Rating (Hardness):**	3: Normal Adult out of 5
Activities:	Solo: Yes Toys: Yes Lesbian: Yes Straight: Yes Group: No Fetish: No		
Uses:	Digital: Occasionally Film: Occasionally Video: Occasionally HD Video: Occasionally		
Camera:		**Camera Notes:**	
Comments:			
Notes:			
Biography			

Update Database

Back to Wacs Main Menu

As you can see, we've included quite a lot of information about the photographer allowing us to build a fairly detailed biography page at some later date should the opportunity arise. Again only a small amount of this data is actively used at the present time, and so long as the abreivated code and name is present, it should work fine.

Chapter 12. Migration Tools

About Migration

Collecting data about sets and models is an interesting activity and something that Wacs is designed to support as well as it possibly can, but there are limits to what can be done completely in issolation. It has long been in our plans to offer a way of exchanging data about models and sets between systems and one of the basic enabling steps is to allow the export and import of both model and set data. Additionally in the commercial arena, it is useful to be able to move sets between different servers for a range of reasons related to deployment, scaling and upgrade planning.

To facilitate the moving of sets between different Wacs installations we have developed a number of representations of the Wacs database records in the XML (eXtensible Markup Language) format. There are two main formats - the one that represents model data and the one that represents set data. Each of these include extensive cross-referencing features using both identity maps (see ???) and download records. In all cases, for the best results import the model information first and then the set information.

wacsexport - Exporting Model Details

The creation of the Wacs Model XML files is the task of the **wacsexport** command. This is currently only available as a command line application, although in the future we plan to include an XML download option somewhere in the Wacs system. To export a model's data as XML, you basically just run **wacsexport** followed by the model number. Thus if you want model number 18 exported, you would type:

```
% wacsexport 18
%
```

This will create an XML file in the current directory called the model's name, a hyphen and then the model number. If Sabrina is model no 18 on your Wacs server, then a file called `Sabrina-18.xml` will have been created.

wacsexport accepts only one modifier argument which is `--noimages` which instructs it not to include any images in the output. Normally it will include both headshot icons within the XML file. This option is to remove any copyrighted images for which you do not have re-distribution rights for prior to posting to the internet, etc.

The XML file contains details of a specific model including her model record, head shot icon in both normal and big forms if available, idmap records and the download records for all her sets that have a known source. This file has the binary data (typically both of the headshot icons) encoded in base 64 so that it can be safely emailed, cut-and-pasted, etc.

wacsimport - Importing Model Details

There are currently two ways to import Wacs Model XML files into a Wacs site - you can either use the **wacsimport** command or the web-based **Upload A Model XML File** option of the model manager. Both systems will do their best to avoid duplication by checking first if any of the identity maps of the model detailed in the XML file are already present in the importing Wacs server. If it is a new model that Wacs installation has not seen before, it will proceed to create the new model's record adding headshot icons and download records as appropriate. To use the **wacsimport** on our sample Sabrina file, use:

```
% wacsimport Sabrina-18.xml
Keyless ID map for JAFN
%
```

The error message above indicates that one of the Identity entries we were importing was incomplete and therefore merely a place marker that we can't use for positive identification. Generally such messages are purely informational and can be safely ignored so long as there is at least one other ID map that can serve for identification purposes. The assumption here is that if the model is recorded by one site as being a certain ID on a remote site, any new information about her should be loaded onto the record of a model who has a reference to being the same ID on the same named remote site. So, if the XML file says Sarah is “sar001” on site AMK, “55” on site XYZ and “12789” on site TVW and we have a model who is “sar001” on AMK, that model's records will be updated with extra ID mappings as “55” on site XYZ and “12789” on site TVW.

Warning

By default, a brand new model number will be issued on the new system unless **wacsimport** determines that the same model already exists there, quite possibly under another model number. If you are not sure of your ID maps being comphrensive, it's probably best to check for the model in the alphabetical index beforehand.

Note

At the moment (Wacs 0.8.6 release), both the **wacsmodelmgr** and **wacsimport** applications have relatively little ability to add additional information to an existing model's record by importing an XML file from elsewhere. This aspect is being actively worked on and will improve in coming releases. At present the model manager in Identity Management mode can import additional identities (IDmaps) and download records - please see Chapter 14, *More About Model Manager* for details.

wacsxmlout - Exporting Set Details

In parallel with the **wacsexport/wacsimport** commands for the exchange of model data between Wacs installations, there are corresponding commands called **wacsxmlout** and **wacsxlmin** for the exchange of set data. It is important to realise that unlike the model XML file exchangers, these do not carry anything more than the information and icons. The actual set contents themselves must be transfered seperately. It is quite possible to send someone just the Wacs Set XML file and for them to combine it with a zip file downloaded off a subscription web site using their own credentials on that site.

To use wacsxmlout, you call it much as before this time using just the set number for the what you want exported - thus to export set **417**, you would use:

```
% wacsxmlout 417
%
```

This will create a file called `set417.xml` in the current directory. In addition to import it, you will need a suitable zip file with a matching name; for set **417** this would be something like `set417.zip`, or `set417.wmv` or `set417.mpg` if it's a movie clip. These names should match what it's saved as when you download it from a Wacs installation. This command does imbed an icon of the set inside the XML file; a future release will add the `--noimages` option.

wacsxmlin - Importing Set Details

This is by far the most complex of the Wacs XML commands as it has the most work to do. This command basically reads the Wacs XML file it is given, tries to find the matching zip or movie file, unpacks it, chooses a location for it, moves it there and adds it's details to the database. At it's simplest you use it as follows:

```
% wacsxmlin set417.xml
Destination Area: solo
Category: gallery010
%
```

It additionally supports two extra command line options, `--clone` which says put it exactly where the XML file says it came from and is ideal for copying content between in-house and production internet web servers. The other one is `--default` which merely does whatever the default answer would be on this site. For a site with gallery layout, this should pretty much do the right thing.

wacsselout - Exporting Selections

Just as we can export model and set information, there is also a tool for exporting selections such as saved searches and connections. This tool is called **wacsselout** as it exports selections. It can handle both selections and connections - simply prefix the number you give with `c` for connections and `t` for saved searches (aka tag sets).

```
% wacsselout c3
%
```

If you export connection 3 for instance, a file will be created in the current directory called `conn3.xml`. If you export a saved search 3, a file will be created called `tag3.xml`.

Warning

At present there is only one way to use a WACS selection XML file and that is to use the **wacsselin** utility. This is very simplistic and ignores any sets in the selection that are not present on the importing server.

wacsselin - Importing Selections

The **wacsselin** utility is new in Wacs 0.8.6 and is rather simplistic. It reads the selection XML file and recreates the entries in it that refer to sets or models that the destination WACS server currently has installed. It ignores - but not necessarily silently - any entries in the file that refer to sets it doesn't have. In future versions it will hopefully create download records and maybe even models to facilitate the more thorough process of adding the entries it needs. Using **wacsselin** is simple enough:

```
% wacsselin conn30.xml
%
```

The above command will import all entries from connection 30 as saved in an XML selection file of a connection into the connections area of the current WACS server. It will be allocated the next available

connection number, unless the command line option `--clone` is specified in which case it will be the connection number as specified by the XML file.

Chapter 13. The Download System

Overview

One of the significant features of Wacs is it's ability to do automatic downloads of sets from a number of subscription web sites. The same mechanisms can also serve to do automatic updating of internet-facing production web sites from in-house content preparation systems when used in a commercial context.

The download system works on the basis of using the template details contained in the vendor database (see the section called “Vendor Manager” in Chapter 11, *Other Web Based Tools*), filling them in with a model's details from the Identity Map (IDMap) and firing off a web request for her model page to the remote web site. With the model page retrieved, the HTML of the web page is then parsed looking for links which match the template for either video or image set links on that site. Once matching links are found, they are added to a list of known sets for that model (download records). Additionally in due course we hope to promote the exchange of download records between people on the internet, so that they can find more sets by their favourite models. From a commercial web site owner's viewpoint, they can use the IDmap and vendor templating system to link to other web sites gaining a commission for referals from the other site.

Once a download record has been created for a specific set on a specific site, Wacs includes an automatic download tool that can use a quiet time of night to go fetch those sets from the remote server. Once collected from the remote site, the wacs model page and download list willl show their presence and allow you to unpack them using the unpack and placement manager much as described earlier for normal sets. The integration of the download mechanism allows additional clues such as set type and photographer to be extracted from the information given by the upstream site.

The Changing Face Of Downloads

Over the time we've been developing Wacs, things have moved on and website owners have become very concerned about the issue of *site scrapping* - the sucking of all the content off their site by automated downloaders. This not only causes huge additional bandwidth usage and server load, but in some cases costs them real money if they have to pay for the bandwidth used from their servers. As content providers ourselves, we understand their pain, but at the same time having started out as merely avid collectors, we really want to know that we have everything we can get by the models we're interested in.

After initial private experiments with fully automatic downloading, we fairly rapidly progressed to a viewpoint of providing tools that focused only on pre-defined models because the automatic tools were both against the terms and conditions of most sites and because the quantity of material was simply too much to cope with. The public versions of Wacs have always been based on the idea of creating a model definition for those models you like and working from there.

Some recent developments have made it even harder to do automatic downloading with various techniques including extensive use of short-lived cookies and image capture challenges being used. Additionally our usual web browser of choice, **firefox** has given up on storing it's current cookie collection in a readable text file making it even harder to recover those vital cookies. As a result, starting with Wacs version 0.8.4, we've started a new policy of using web pages with links on as a way of guiding you through the download process. These give you links to what needs downloading next but including the human element to ensure the challenges and other tricks are seen and answered. The older command line based tools are still there and will remain usable for those sites that do not impose additional restrictions beyond username and password requests.

The topic of how the download mechanism is initially set up is rather too complex to discuss here and we would direct you to the Wacs developers list at our sourceforge site [http://wacsip.sourceforge.net/]

for more information and discussion on this. For now we're just going to give a brief overview of how the mechanism can be used.

Components Of The Download System

chkmodel

The **chkmodel** takes a single argument, a model number and proceeds to look at all the IDmaps for that model - for each IDmap, it checks if it knows a username and password for the relevant site and if it does, downloaded the relevant model page from the site. It then checks through that model page for the image set and video clip links, and adds any new ones it finds to the collection of download records for that model. Depending upon what it finds, it will update the models IDmap to show when it was last checked and when it last found a new set for that model. This way those models with an actively growing portfolio are checked more regularly than those who are no longer working and producing new sets.

The **chkmodel** is also capable of working on a saved web page in an off-line mode for those web sites whose pages are marked as unusable (ie the `vmpaguse` field in the vendor database entry for that site is set to `N` for no). Here it will look through the model index page seeking links to set pages and recording the details of what it finds that it did not already know about - in most cases this will probably be an incomplete (download status `X`) download record. The model's record for that site will be updated to show that it was last checked on the day on which the model index page it processed was saved to disc. The additional details needed can be added using the **wacsdnlframe** web app at a later time (see wacsdnlframe for more information).

Note

The enhancements to **chkmodel** to allow it to use saved files well were added at Wacs release 0.8.4; additional work to make the naming of saved files easier is expected to be added at a later release.

To save a page for chkmodel to read, it needs to go in a very specific directory - the `modelpage-cache` directory of the site directory in the download area. In a default package installation, and for a site called *wacsdemo*, this would be: `/var/spool/wacs/download/wacsdemo/modelpage-cache/`. The file then needs to be called a very specific name - the first part is the site code (for wacsdemo this might be `WACSD`), then a hyphen (-), then the model's key (model number/reference on that site) such as `123`, and then another hyphen (-) and finally her model number on this Wacs installation such as 345. Thus the page would need to be saved as `WACSD-123-345` for this model.

You would then run **chkmodel** `345` to scan that saved page for new sets. Any icons found in `WACSD-123-345_files` would also be transfered to the model's icon directory in `icons` in the site download directory (eg `/var/spool/wacs/download/wacsdemo/`). This naming convention is a little cumbersome and exposes more of Wacs's internal workings than is strictly desirable, and we do plan on addressing that in a future release. Right now we want to get the technology available to people who want to use it on sites where automatic download is not supported.

refresh

The **refresh** command is designed to be run by a nightly automatic job (aka **crontab** and calls **chkmodel** with the appropriate arguments to check a specific number of models each night. It limits itself to just a few so as to not be a "bad citizen" and outstay it's welcome. Refresh does actually use some fairly carefully designed metrics to optimise the way it looks for models, biasing it's checks towards those who are receiving regular updates and within those selecting those that have gone longest since the last check. It is generally pretty thorough in what it does.

getarc

The **getarc** command completes the picture by actually downloading the sets previously highlighted by the combination of **refresh** and **chkmodel**. Once getarc has obtained the necessary zip archive or movie file, it marks the set as pending at which point it can be unpacked using the standard Wacs unpack and placement tools invoked from either the download list or the model page (full version).

Again it is normally expected that **getarc** will be run at a quiet time of night by the **crontab** timed execution system.

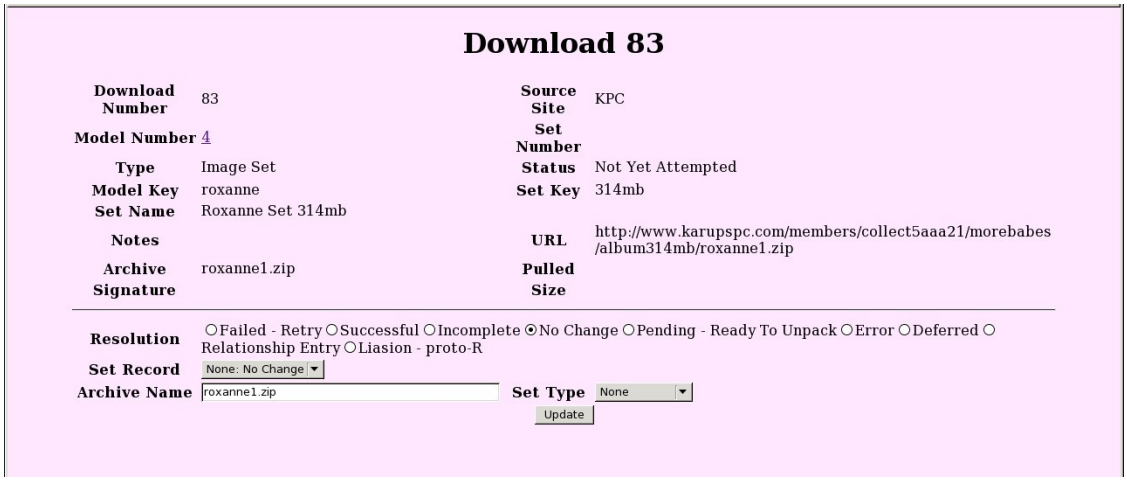
wacsdnlnext

The **wacsdnlnext** web app is designed to assist you and guide you through downloading sets from those sites where getarc is not suitable for use. Detailed instructions on using it can be found in the reference section, wacsdnlnext.

Warning

We do plan to overhaul the download system in the relatively near future to better cope with cookies and other techniques that sites use to ensure their security. We also currently invoke the command line based web download tool, **wget** for all downloads and feel it would be better if it did the work internally using the web download library for perl (LWP). Doing this should also improve the portability of Wacs to other platforms because expecting Unix/Linux command line semantics in this is a significant problem on other platforms, especially Windows.

Wacs Download manager



The Wacs Download Manager, **wacsdnlmgr** allows the resolution of some of the most common problems with download records. It's primary purpose is to allow manual changes to the status and set associations of download records. If you find a record you've downloaded has a broken zip file, you can use the download manager to list that download as **Failed** so it gets re-downloaded at some point in the future. If you have both models in a lesbian set having a download record in their own right for the set, you can use the download manager to mark the second download record as either successfully resolved or as a relationship entry. This usually depends on which model's name comes first - if the model upon the strength of whose record we unpacked it is the first model mentioned, the other(s) is a relationship entry; if not, both are

marked as being successfull. It can also mark a set as being in error if the model featured in it is patently not the model to whom it has been attributed.

The easiest way to access it is via either the link on the download number on the detailed form of the model page, or on the entry in the download list. The first option via the model page only works while the download record is marked as unresolved (ie not **Successful, Relationship Entry** or **Error**.) To access a model record that is considered resolved, administrators will find a link on the download number listed in the details in the set info page.

Configuring Automatic Download

As we've discussed, one of the significant features of Wacs is it's ability to monitor subscription web sites for updates to your favourite models, and while these tasks can be done manually “on-demand”, they can also be automated in conjunction with your operating system's timed execution facilities. For this to work of course, you do need to make sure that you've added your account details for each site that you want monitored into the vendor database using **wacsvendmgr**. You also need to make sure that each model's identity on that site has been entered as well; this can be done using the Identity Management mode of the **wacsmodelmgr** which will be discussed in the next chapter (See Chapter 14, *More About Model Manager*).

Defining Sites

If you're using a pre-defined site record for the Vendor site you wish to download from, you should have an easy time; just enable the site, add your password, and then add a few model details using the Wacs Model Manager and it should all start working. If you're creating a new Vendor site description, it can be rather more difficult.

The first thing you need to understand is how to set the Image and Video download id types to the correct values (vidtimg and vidtvid in the vendor schema - see section 3 of the programming manual). The actions take place in two places - first the chkmodel script must actually find the desired zip or video file name and create a download record requesting it be fetched. Second, the getarc script must actually perform the necessary retrieval and saving actions. The following table details how the various values currently work:

Type	Value	Description
Image Sets	*1*	Default - just get the URL specified in the download record modulo the addition of the specified image server URL if the download record URL doesn't start with `http://`
Video Clips	*1*	Default - just get the URL specified in the download record modulo the addition of the specified video server URL if the download record URL doesn't start with `http://`

Alternative values you may see invoke custom code designed for specific sites only which are not expected to be applicable to more general use. These typically include work-arounds such as re-navigating the lead-in web pages as the final target URLs are liable to change, or guess that given a certain stem such as `jennifer1.zip`, additional zip files in this set will be called `jennifer2.zip, jennifer3.zip` and so on.

Using cron For Unattended Download

On Linux, MacOS X and other Unix like systems, the automatic execution system is known as **cron** and it's operation is managed by a somewhat cryptic program also called **crontab**. Normally **cron** will be running

already since it is used by the operating system itself for housekeeping. It is important to understand that crontab does not "wake up" a machine that is switched off or suspended, and so setting to run an overnight job when you turn it off each evening as you go to bed will not prove useful because the job will never run. If the machine is left on, as most servers probably will be, a time in the early hours of the morning will probably be fine; if the machine is normally switched off overnight, choose a time when the machine has a good chance of being switched on. The tasks will run in the background and will not affect other users of the system particularly.

To make use of the automatic download feature, you need to enable two more programs to run at a suitable time: **refresh** and **getarc**. **refresh** scans registered models and checks for updates; **getarc** fetches a number of the updates that refresh has found. To enable the automatic scanning feature, you need to open a terminal window as a suitable Wacs administrator user and type:

```
% crontab -e
```

and add the following line to the crontab file:

```
22 0 * * *      /usr/local/bin/refresh
```

Tip

Crontab usually calls up the **vi** editor - you can recognise this by a tilde (~) symbol repeated all the way down the left hand side of the screen. This editor is a bit cryptic if you don't know it, but basically you need to press the lower case letter `o`, type the line above, press `Enter`, then press `Escape` and then type `:wq`. That should get the job done.

This will cause the **refresh** command to run at twenty-two minutes past midnight each day. Note that by default **refresh** will only check those models the Wacs system currently considers to be “Active”, ie those who have been found to have new sets within the last three months or so. An additional option to refresh, `-a` allows it to scan all models, including those marked as dormant. It is recommended to run this occasionally so that it will double check the dormant models at a much slower rate; you might wish for instance to check dormant models twice a week, and to do this you would add a second crontab entry of:

```
42 0 * * 2,5    /usr/local/bin/refresh -a
```

Once **refresh** has been run, any newly discovered sets will be registered in the WACS system - they can be examined using the Download list **wacsdnllist** or are listed per-model in the detailed model page **wacsmodelpage**. In order for them to be downloaded automatically you will also have to invoke the getarc command at some suitable point - I choose 6:45 in the morning. To set this up, you add the following to the crontab in the same way as above:

```
45 6 * * *      /usr/local/bin/getarc
```

The **cron** command when it runs will automatically email you with the details of what happened for each of the commands you've set it up to run in the **crontab** file.

Chapter 14. More About Model Manager

Additional features

When we previously looked at the Wacs Model Manager back in Chapter 6, ***wacsmodelmgr*** - *The Wacs Model Manager*, we were concentrating primarily on it's features related to creating and amending model records. It is actually capable of quite a bit more than this, particularly in the areas of IDmap and download record handling. In the section called "**wacsimport** - Importing Model Details" in Chapter 12, *Migration Tools* we mentioned in passing that the model manager can also handle the importation of model XML files. In this chapter we're going to look at those additional features and at the roadmap of where the development of the model manager application is going.

A key thing to realise is that the Wacs model manager is still the subject of pretty active development and is likely to see significant improvements over the next couple of Wacs releases. To this end, and to help clarify why we're covering only some features and not what might appear to be obvious others, we've tried to lay out what the current state of play is. The table below gives some indication of what works now and what is expected to work in the near future.

Table 14.1. Current and Planned Features in Model Manager

Feature	In 0.8.6?	Description/Notes
Add New IDMaps	Yes	Use Identity Management Mode
Edit/Delete IDMaps	No	Planned for future release
Import New Model From XML	Yes	Use Model Update Mode
Update Model Details From XML	No	Planned for future release
Add new IDmaps From XML	Yes	Use Identity Management Mode
Add new download records from XML	Yes	Use Identity Management Mode
Update IDmaps from XML	No	Planned for future release
Update download records from XML	No	Planned for future release
Update Web Site Owner Info	No	There are some currently inaccesible fields in the model database for use by Web Site owners in relation to models. This function will allow use of these fields.

Importing XML files in wacsmodelmgr

The Wacs model manager can handle the XML files created by the **wacsexport** command, but initially only to a limited extent. If you consider the state of play as regards any given model and an XML file containing her details, you can see that there are five obvious states:

1. The model doesn't exist on our Wacs system

2. The model does exist and the XML file has additional information

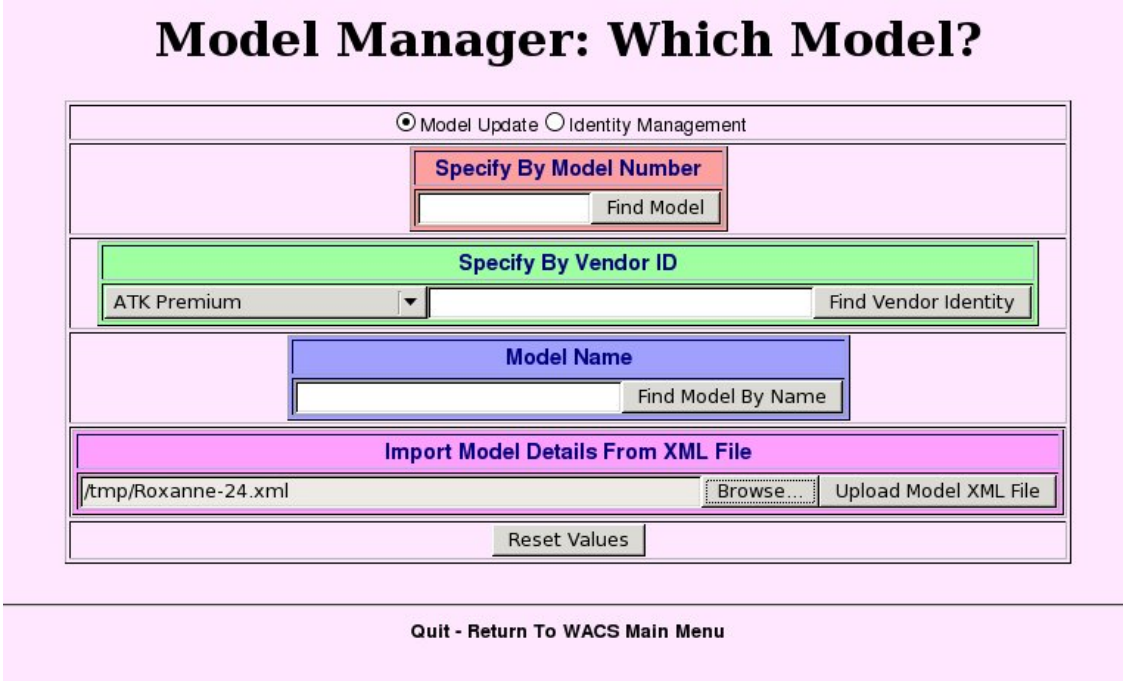
3. The model does exist and the XML file has less information than we do

4. The model does exist and the XML file has *conflicting* information to what we have

5. The model exists and the information is the same.

As of Wacs 0.8.3, the first of these is implemented in full - upload a Wacs model XML file and if the model isn't known, her record will be created. The third of these isn't really a problem for us although it might be nice to contribute back whatever additional information it is we know. The fifth of these is also relatively easy as the model manager just loads up the relevant model's record to say “Hey, we know her, and here's her details”. It is the second and fourth options that require more work and do not currently work as you might expect in Wacs 0.8.3 - at present if the model is known to Wacs, her record will simply be shown but no indication will be given of differences. The second will be the next part to recieve attention so that any field that is empty in our Wacs database will be filled in by values from the XML file.

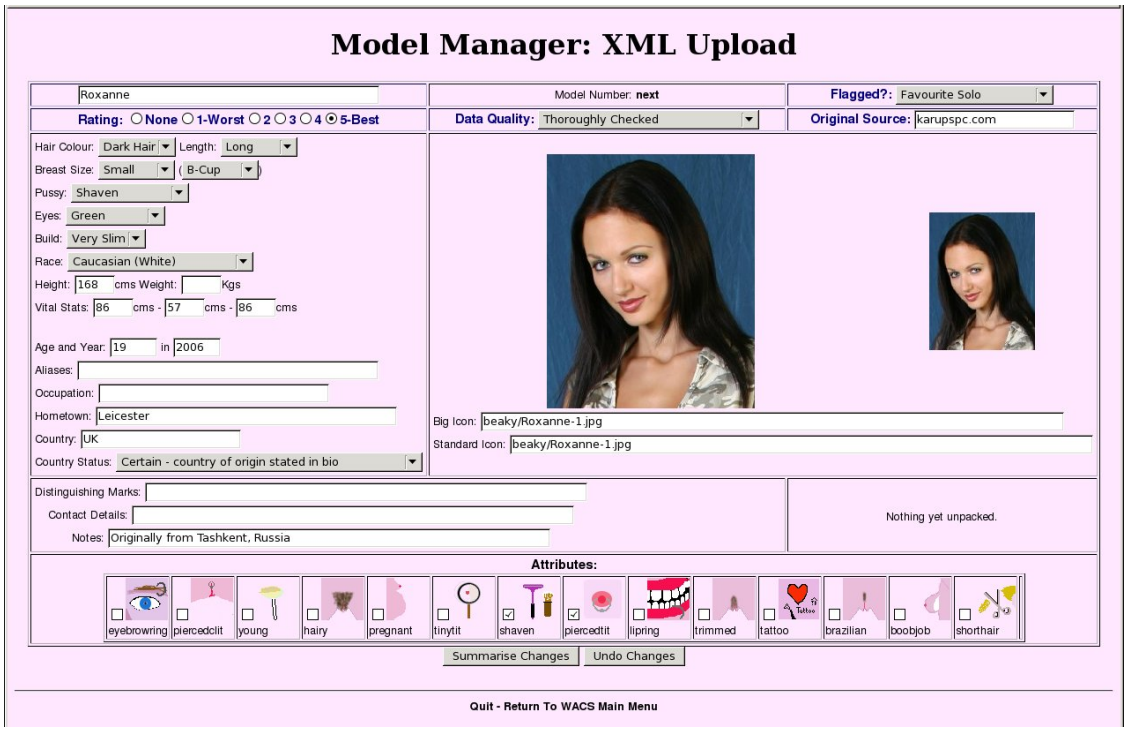
Now that we've defined the terms of reference and you have some idea of what will work and what does not, lets just look at the steps involved in importing a new model record from the XML file.



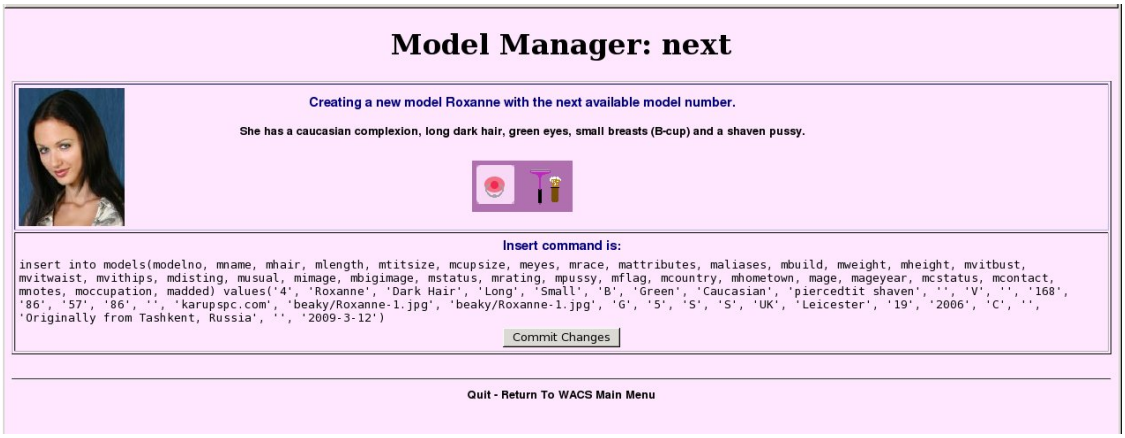
It is quite a simple process to browse to whereever the XML file you want to load is, choose it and then click on the **Upload Model XML File** button. Do remember that the sample files we're using here can be found in `/usr/share/wacs/samples/models` on versions of Wacs installed by package, and in the `samples` directory of the unpacked tarball if installing from that.

Warning

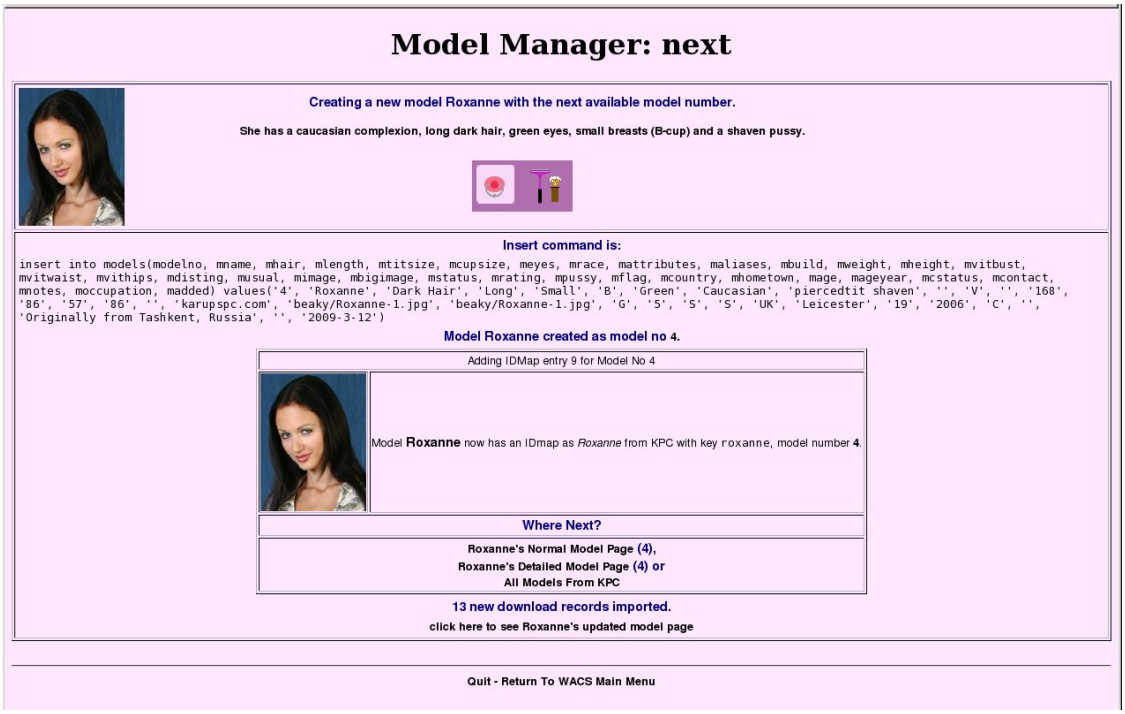
The model manager XML import system uses existing IDmaps as a way of determining if it already has the model concerned in it's database. If you have a number of models with no IDmaps set, it could easily import a model twice. If in doubt, check your model directory before importing the XML file.



Here we see all the information from the XML file filled into the normal model update screen as we've seen before with the usual **Summarise Changes** button included. The only sign that this is an XML import is the title at the top of the screen (although hopefully that is screamingly obvious!). As usual, check it over, particularly looking at the model attributes which could be different on your system than on that which created the XML file in the first place. It's quite possible to change any of the values you disagree with here before creating the record. Once you're happy, just click on the **Summarise Changes** button to proceed.



We now see the summary of the information that the model manager is about to use in creating this record including the actual SQL is has written to do the deed itself. So long as you're happy with that, just go ahead and click on the **Commit Changes** button.



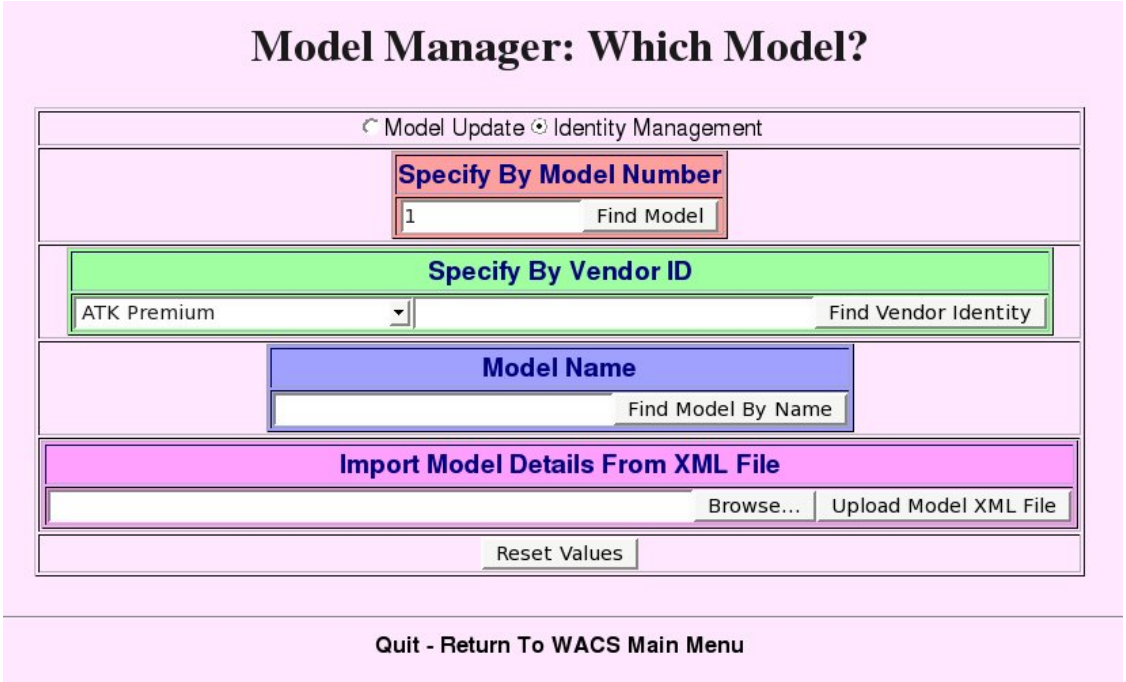
This is the final screen of the XML Upload conversation and as you can see it's fairly busy. The top bit relates to the actual creation of the model record itself and is the repeat of what happened before modulo any changes that have happened since (like someone else creating another model record in which case the model number we get will have changed). The next bit shows that we've added an identity map entry, in this case for KPC (Karup's PC), and the final piece mentions that it has also imported thirteen download records for this model. That is a part of the model manager code that is unique to the XML import feature - it can also be used to import new IDmaps and download records for an existing model but we'll discuss that feature in the next section.

Identity Management Mode

You may have noticed that on the model manager's front screen there are a pair of radio buttons right at the top - these are marked **Model Update** and **Identity Management**. Everything we've done so far has been in **Model Update** mode, so now we're going to look at **Identity Management**. As you have probably guessed from it's name, it focuses primarily on the issues of Identity Maps and their related download records.

Manually Adding An IDmap

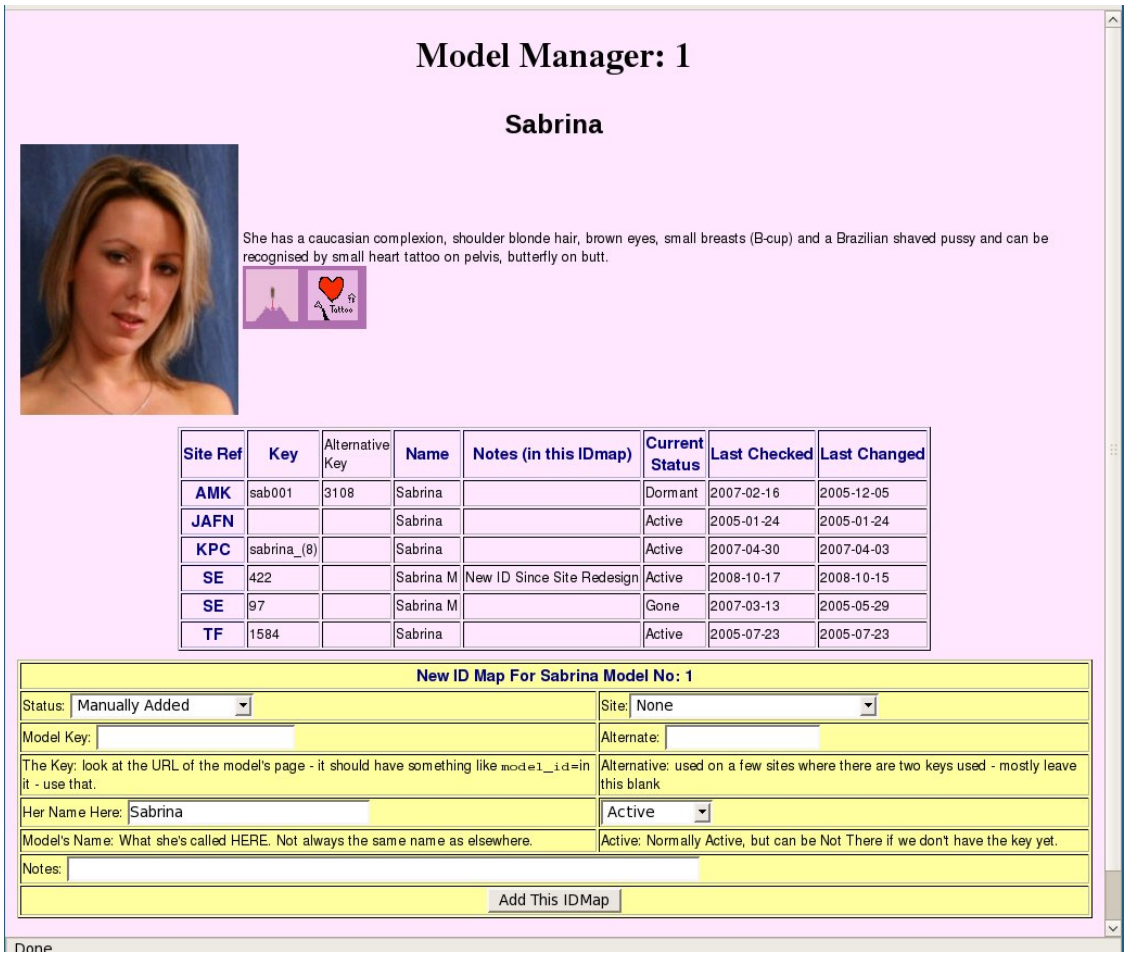
The first example we're going to work through is simply to add a new IDmap to an existing model, in this case Sabrina and we're going to simply record that she is model no *1* on our demonstration site (called WACSD aka WACSdemo).



The first steps as shown above are simply to select the **Identity Management** radio button and type in the desired model number as before. All you then need to do is click on **Find Model** and you go straight to the IDmap page.

Note

For **Identity Management** mode the first three ways of finding the model work as you might expect from **Model Update** mode. The final one, uploading an XML file, doesn't do the same things at all and we'll cover that later in this chapter.



Model Manager: 1

Sabrina

She has a caucasian complexion, shoulder blonde hair, brown eyes, small breasts (B-cup) and a Brazilian shaved pussy and can be recognised by small heart tattoo on pelvis, butterfly on butt.

Site Ref	Key	Alternative Key	Name	Notes (in this IDmap)	Current Status	Last Checked	Last Changed
AMK	sab001	3108	Sabrina		Dormant	2007-02-16	2005-12-05
JAFN			Sabrina		Active	2005-01-24	2005-01-24
KPC	sabrina_(8)		Sabrina		Active	2007-04-30	2007-04-03
SE	422		Sabrina M	New ID Since Site Redesign	Active	2008-10-17	2008-10-15
SE	97		Sabrina M		Gone	2007-03-13	2005-05-29
TF	1584		Sabrina		Active	2005-07-23	2005-07-23

New ID Map For Sabrina Model No: 1

Status: Manually Added | Site: None

Model Key: | Alternate:

The Key: look at the URL of the model's page - it should have something like `model_id=` in it - use that. | Alternative: used on a few sites where there are two keys used - mostly leave this blank

Her Name Here: Sabrina | Active

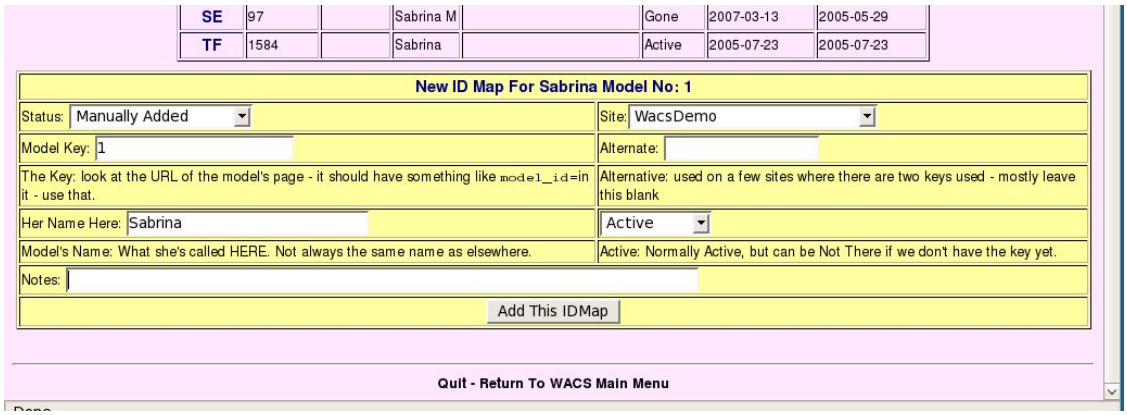
Model's Name: What she's called HERE. Not always the same name as elsewhere. | Active: Normally Active, but can be Not There if we don't have the key yet.

Notes:

Add This IDMap

Done

This is the list of existing IDmaps for Sabrina and as you can see she has quite a few! Of particular interest, and I know we've mentioned this before is that she has two on Sapphic Erotica (SE) and as you can see from the notes, the id changed after they did a site redesign. At the bottom of the screen, you see the web form we can use to add an additional entry. Something which we're now going to do:



SE	97		Sabrina M		Gone	2007-03-13	2005-05-29
TF	1584		Sabrina		Active	2005-07-23	2005-07-23

New ID Map For Sabrina Model No: 1

Status: Manually Added | Site: WacsDemo

Model Key: 1 | Alternate:

The Key: look at the URL of the model's page - it should have something like `model_id=` in it - use that. | Alternative: used on a few sites where there are two keys used - mostly leave this blank

Her Name Here: Sabrina | Active

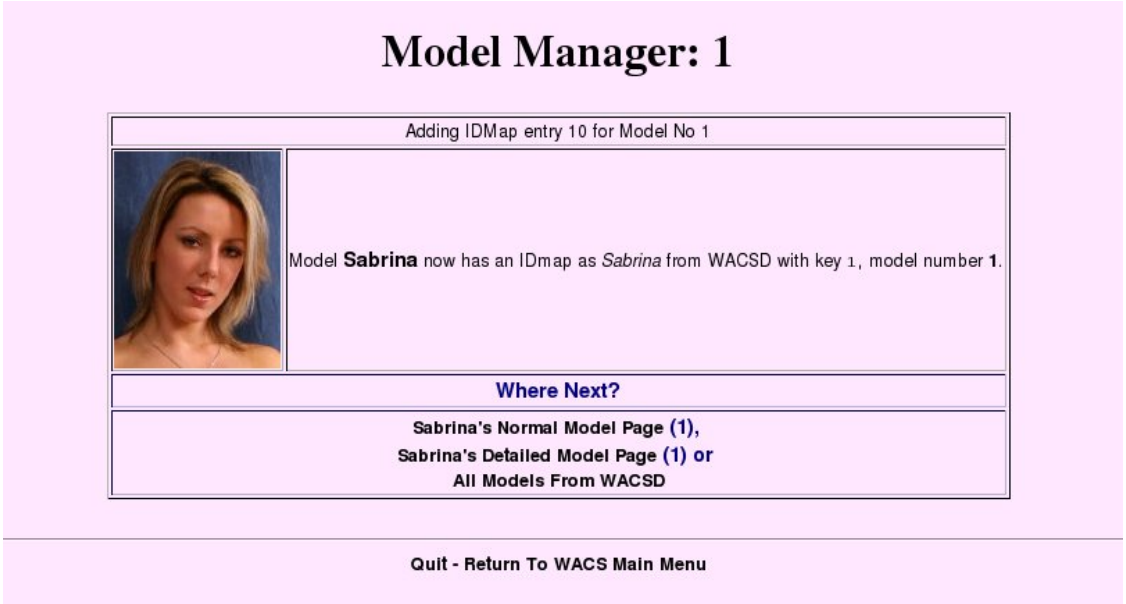
Model's Name: What she's called HERE. Not always the same name as elsewhere. | Active: Normally Active, but can be Not There if we don't have the key yet.

Notes:

Add This IDMap

Quit - Return To WACS Main Menu

This is simply a case of choosing the WACS Demo site from the pull-down menu and working out and adding in the model number or reference on that site as the key. If the site you want to add isn't in the list, it can be quickly and easily added using the Wacs Vendor Manager (described in the section called “Vendor Manager” in Chapter 11, *Other Web Based Tools*). Once entered, you just click on **Add Idmap** and you should see the following screen in confirmation:

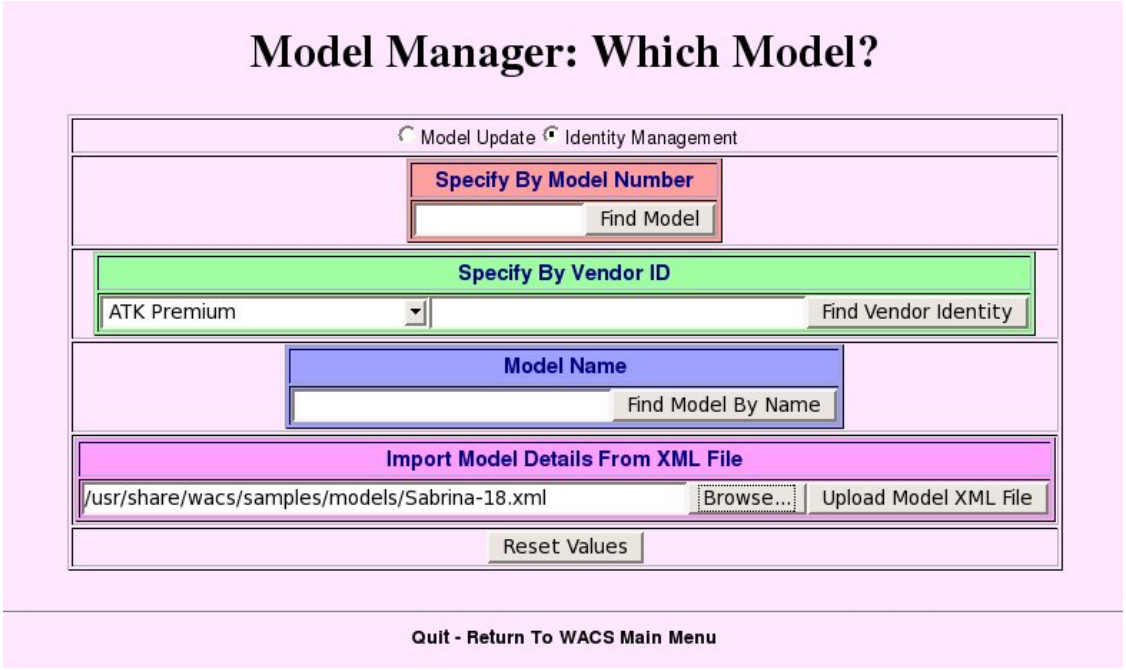


Note

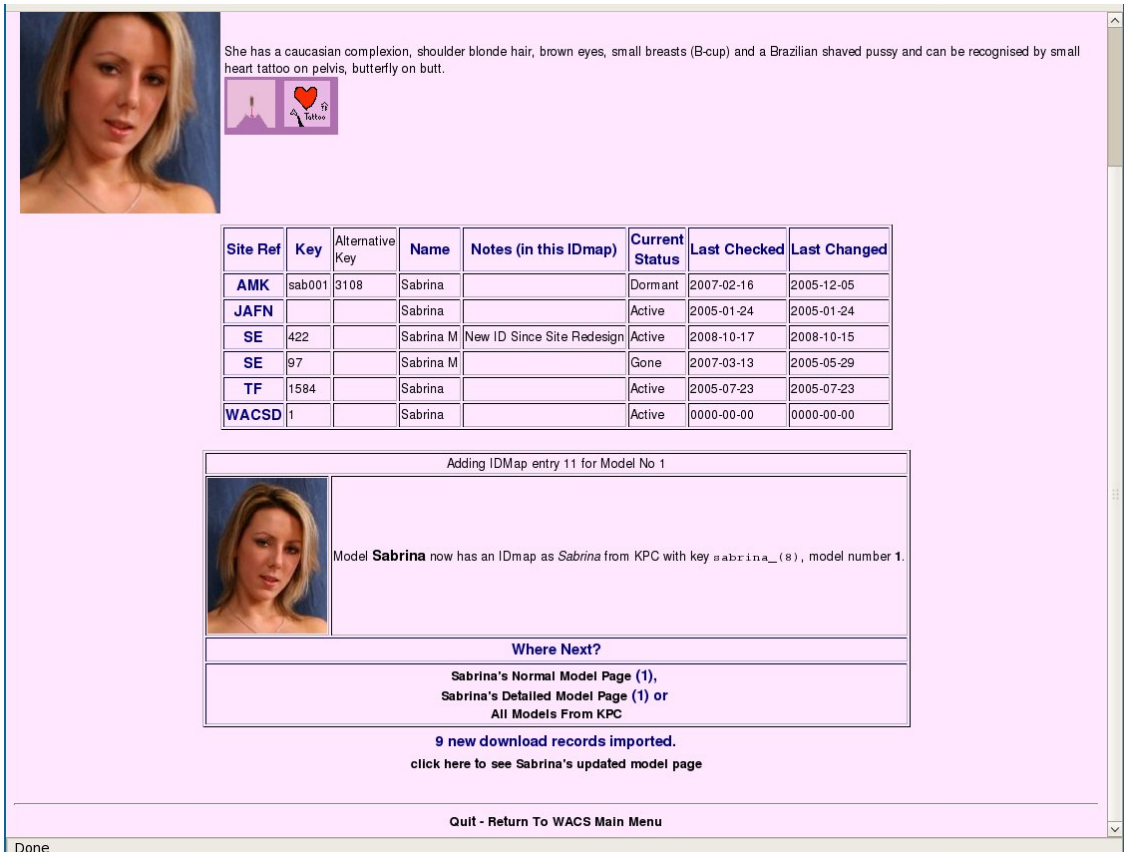
We're planning to allow editing and deletion of IDmaps in addition to simple creation of them from within model manager. Hopefully this feature will be in the next release.

Importing IDs From XML

We mentioned earlier that using an XML file in **Identity Management** mode was rather different from doing so in **Model Update** mode. For the purposes of this little demonstration, we've deleted all the download entries for Karup's PC (KPC) and the IDmap from Sabrina's records.



We're then going to proceed to import the supplied sample XML file (usually `/usr/share/wacs/samples/models/Sabrina-18.xml`) in **Identity Management** mode and you'll see what it does. The first step of course is to select **Identity Management**, browse for the XML file and click on **Upload Model XML File**.



As you can see, the model manager has found our record for Sabrina, and realised that we're missing the KPC identity and added it. Additionally it has discovered that the XML files has some extra download records it didn't have and has added those too.

Note

This trick only works correctly because Sabrina still has other uniquely identifiable (ie complete) IDmaps which can be used to associate her with the IDmaps from the XML file. That is we have some IDmaps that are common ground and indicate the link between our model Sabrina and the model detailed in the XML file.

Chapter 15. Command Line Tools

Overview

The Wacs system has actually been around quite a few years now and in the earlier releases almost all the collection administration was performed using command line tools. While we've been busy developing new web-based tools over the course of the Wacs 0.7.x and 0.8.x release series, the old command line tools are most definitely still there and perfectly functional. In a few cases (addassoc, delset) they are the only way of doing a specific obscure task; in others they may simply be useful building blocks for your use of Wacs. For instance, the addmodel script might be useful in writing a migration script from your previous storage method.

In this chapter, we're going to look at what exists and what they can do for you... there are still a few tasks for which they are the only way to do certain things. Firstly, lets summarise what is available:

Table 15.1. Command Line Tools

Command	Web Equiv	Description
addmodel	**wacsmodelmgr**	Creates a new model record
addassoc	**wacsaddassoc** (new in 0.8.4)	Associates a model with a pre-existing set
delset	None	Deletes a set from the Wacs database
generate	**wacsplacemgr** and **wacsinfomgr**	Creates new icons from the image sets, then runs **updateinfo** (see below)
genvideo	**wacsplacemgr** and **wacsinfomgr**	Creates new video sets and updates existing ones (actually used directly by the web-based tools)
updateinfo	**wacsplacemgr**	Creates new image sets and updates existing ones (actually used directly by the web-based tools)
updatestats	None	Updates the statistics in each model's records based on the sets she's associated with - should be run nightly

We're going to split these command line applications down into two categories - those that handle data manipulation and those that perform processing tasks that require no other intervention. The **addmodel, addassoc** and **delset** commands fall into the former category; **generate, genvideo, updateinfo** and **updatestats** into tha later.

Data Manuipulation

The addmodel Command

Note

Unless you're using this from a script or similar, you will find that using the model manager (see Chapter 6, ***wacsmodelmgr*** - *The Wacs Model Manager*) is a much better idea. Almost all models created using **addmodel** will require additional work afterwards.

The **addmodel** script is a command line interface to creating models (and actually IDmaps as well). It takes the form of the command **addmodel** itself followed by a row of space separated **name=value** pairs. A simple example session with addmodel would look like this:

```
% addmodel name=Sarah hair=Redhead length=Normal titsize=Small pussy=Sh
aven attributes=shaven image=Sarah-1.jpg
Sarah is not known
Name: Sarah
Hair Colour: Redhead
Hair Length: Normal
Titsize: Small
Attributes: shaven
Pussy: S
Aliases:
Usual Source:
Image: Sarah-1.jpg
Source:
RefNo:
OK to add (x for exclude) ? (y/n/x): y
Model number is 1
SQL: insert into models (modelno, mname, mhair, mlength, mtitsize,
mattributes, maliases, musual, mimage, mstatus, mflag, mpussy, madded)
values( '1','Sarah' ,'Redhead','Normal','Small','','','','','A','N',
'S','9-OCT-2006')
%
```

It is worth noting that this program currently calls the old "xv" Unix/ Linux image viewer to display the model icon, but could easily be changed to run **eog, display** or some other image viewer. The variable *imgviewer* at the top of the program controls this. It is not critical if it's not present or can't make a connection to an X-server. In most cases, addmodel will actually prompt for the major items of data if they are missing from the command line. It currently prompts for hair colour, length, titsize, pussy and attributes. If a source is specified but no usual, it will default it (in most cases).

At this point, it's probably worth visiting the Newly Added Models section of your WACS web site to check that Sarah has indeed appeared as intended. The following table lays out the names that addmodel recognises along with the legal values for each one - neither name or value is case sensitive except for the model name that is expected to be naturally capitalised, any value that contains a space should be enclosed in double quotes:

Table 15.2. Attribute Names And Legal Values In addmodel

Name	Legal Values
hair	blonde, brunette, redhead, dark
length	short, shoulder, normal, long
titsize	tiny, small, normal, large
attributes	*any valid attribute keyword or space separated list of keywords*
pussy	hairy, trimmed, brazilian, shaven
aliases	*any name or comma separated list of names*
usual	*site name of where her sets were first found*

Name	Legal Values
image	*pathname to her headshot file* - the same name will be checked for in both bigicon and normal icon areas; if present in the bigicon area, it will be set as well. The value given here will be set for the normal icon even if the file doesn't currently exist.
source	SE, ATKP, AW, JAFN, AMK, TF, IFG, ATE, KPC, KHA
refno	*the primary model key on this site*
altref	*the alternate model key on this site*

So long as you specify both a source site and a *refno* , the appropriate IDmap will be added too. Here is a more complex example of using **addmodel**:

```
% addmodel name=Erika hair=Brunette length=Long titsize=Tiny pussy=Sha
ven attributes='shaven tinytit tattoo' refno=eri046 altref=12475 sourc
e=AMK usual=amkingdom.com image=amkingdom/Erika-1.jpg
[...]
%
```

If we just review the extra entries used here, we'll see that the setting of *attributes* adds some extra information to aid searching by models - in this case, her unusual attributes that differ from what is "normal" are that she has exceptionally small breasts, a completely shaven pussy and a few tattoos.

The next keyword *refno* normal contains her model number at the source site; usually this is just a simple number. However in the case of the AMKingdom web site, there is a corporate wide “identity” which takes the form of three letters and then three numbers; in her case eri for Erika and 046 which we assume means she's the 46th model added with the first three letters of eri. Some of the other AMK group sites, such as ATK Premium navigate using the corporate id as the "model key", but the AMKingdom site (aka ATK Galeria) uses an older system based on a numeric reference. This is the *altref* specified here. We have also encountered a system (now thankfully gone from the site concerned) where the video index used a different model key from the image sets index. Generally having the dual key support available in the IDmaps just adds to the flexibility of the system. The *source* parameter gives us the “domain” in which these keys apply. In this case we're setting it to "AMK" for AMKingdom.com (which is now known as ATK Galleria but we've stuck with the old abrieviation). With these three specified, we'll be adding a record in the idmap table as well, mapping Erika with our model number 2075 to being “Erika” from AMKingdom.com with corporate id `eri046` and model key `12475`. Additional id mappings (for instance Erika also appears on ATExotics) can be easily added using the **Identity Management** feature in the model manager or can be added manually using SQL.

The next name *usual* merely makes a comment in the models table of where we usually find this model, in practice this usually means where we first found her! (This is defaulted to whatever source is set to if not specified.) The final attribute *image* specifies an icon for her (conventionally resized to 120x156) and in this case the icons have been subdivided by source site, so her icon appears in the amkingdom sub-directory. It also checks for the same name of file existing in the bigicons directory, and if it does, that is added too.

The addassoc Command

The **addassoc** command provides a way of manually adding an association between a model and a set. To use **addassoc** you need to be in the directory where the set directory itself is located - so if you're trying to connect a model called `Sarah` to a set in a gallery called *nudity/gallery003* you would need to do the following:

```
% cd /home/wacs/images/nudity/gallery003
% addassoc Sarah Sarah_*
1. Sarah with Long Redhead hair and Small tits - model number=1
Name: Sarah
Hair Colour: Redhead
Hair Length: Long
Titsize: Small
Attributes:
Aliases:
Usual Source:
Image: Sarah-1.jpg
RefNo: 1
Continue (y/n)? y
DB record found: setno 1, title: Sarah PinkDressNoPanties GardenBenchDildo
Added association between Sarah (1) and set 1
%
```

Tip

A a set has to be known by WACS before you can associate it with any model, so in most cases the **generate** command discussed below needs to have been run first.

If you now click on the link on Sarah's image or name on the Newly Added Models list, you should see her model-thumbs page with the new photoset featured on it. You might also wish edit the sets details using the set manager or to re-run the **updateinfo** command to import any of the model's attributes (shaven, tattoo, etc) into the set details as well.

The delset Command

Delset is passed a single argument of a directory name (which is assumed to contain an image set known to Wacs) or video file name and proceeds to delete the database records related to it in the correct order. Once complete it moves the files into the “delset_trash” directory of the download directory where they can be checked on more time before the files themselves are erased.

```
% cd /home/wacs/images/nudity/gallery003
% delset Sarah_PinkDressNoPanties_GardenBenchDildo
Set Number 14 to be deleted.
The set in Sarah_PinkDressNoPanties_GardenBenchDildo has been deregist
ered from Wacs.
The contents has been placed in /var/spool/wacs/download/delset_trash.
You may wish to check them and then remove them.
%
```

Processing Applications

In earlier chapters as we looked into the way that the web interface functions, you may well have assumed that the set inserting process in Wacs was very rigid and structured. We were trying to give you that impression because it is the best way to work but it is absolutely not the case at all! Wacs will actually index pretty much anything you throw at it - the basic ground rules are that there need to be a number of

directories and in each directory there are a number of recognisable image files. The only real rule is that where there are directories containing images, all the directories there should contain images - there should not be a mix of directories containing just images and those containing sub-directories at the lowest level.

What the two image set processing applications (**generate** and **updateinfo**) actually do is to look at an existing filesystem structure and try their best to digest the content into metadata and insert it into the database. The **generate** actually calls updateinfo with the values it was called with, so they are effectively the same. It's just that updateinfo doesn't create icons.

The generate Command

Running generate is actually very simple, to import a tree of file directories (say `/home/wacs/images/redheads/smalltits/Sarah`) you just do:

```
% generate redheads/smalltits/Sarah
[...]
COMPLETED: 12 new records inserted, 7 updated.
Calling make_index() with redheads/smalltits/Sarah
%
```

In this example, **generate** discovered 19 directories containing image files in the directory specified, 7 of which it already knew about and 12 of which were new to it. This is normally detected by the presence of a hidden file called `.info` which gives the basic set number information for each set within each directory. When the web interface is used, the **Wacs placement manager** (see Chapter 8, ***wacsplacemgr*** - *The Placement Manager*) creates a file in the directory called `.unpack` which **updateinfo** uses to gain clues about the model featured, the download record "satisfied" by this set and so on. If this file is found, **updateinfo** reads it and uses the values contained therein - if not, it tries a few other heuristics to determine what it's looking at and if all else fails just simply creates a new set without any model or download associations.

These heuristics vary by source site but basically resolve around looking for an "unsatisfied" download record with a known filename stem that matches the files found in the directory being considered. When it finds such a match, it will retrieve additional information like model number, set type, source site and photographer from the download record and merge that information with what it can determine from the name of the directory and the model's attributes. The download record will be marked as having been successful and will disappear from the model's detailed page. This method really only works when the image file names are long and quite structured in their naming but this is the case for a surprisingly large number of sites.

If a set is added without any associations, it can quite simply be used without any model associations or you can manually add the associations using the appropriate addassoc command. As mentioned in connection with the discussion of set naming, certain keywords are recognised within the directory name and used to define various special attributes in the default metadata for the set. Generate can recursively scan from the top of the images tree (or from any point on the sub-tree), but probably best avoided if you have a large number of directories in the system.

The genvideo Command

The **genvideo** command performs much the same functions as the combination of generate and updateinfo for the importation of video clips into the Wacs system. It actually also includes aspects of the updateinfo program but re-written for the video context. However the video naming convention is often a lot simpler

and what metadata there is in inferred from the download record as very little information can be gleaned from the name of the file itself. In the case of videos, the `.unpack` mechanism used by the Wacs Placement Manager (see Chapter 8, ***wacsplacemgr*** - *The Placement Manager* provides the best way of getting metadata about video sets into Wacs.

Note

Although the relative pathnames appear the same, genvideo operates in a parallel space under the videos tree in the file system. Additionally icons are automatically referred to by the same pathname but in the vidicons directory. Therefore for *`sarah01.mpg`* in *`videos/`*`redheads/smallbreasts/Sarah` , you would place the corresponding icon file as *`sarah01.jpg`* in *`vidicons`*`/redheads/smallbreasts/Sarah`.

The updateinfo Command

updateinfo is actually what does most of the hardwork for generate including all of the database interactions. If called via generate, the icons will be made first - if called directly, no icon manipulations will be undertaken. When the placement manager wants to do two passes of the updateinfo process (in order to merge in model attributes and add additional models), it normally uses the direct updateinfo first and only generates the icons on the second pass through.

Automatic Collection Maintenance Tools

These tools are designed to be run automatically, typically on a nightly basis, to keep parts of the system up to date. The main task at present is to keep model statistics up to date, and this is the job of **updatestats**.

The updatestats Command

The **updatestats** command is there to ensure that a model's record is kept up to date with respect to the sets she is associated with. This is done because the model record actually keeps some basic statistics about the sets, number of images and video clips and types of activity featured in those sets in order to better describe the model and her activities (on camera). Of course, much of this information is ultimately derived from the sets she features in; for instance, if she features in a lesbian set, the assumption is that she does lesbian scenes on camera! Obviously examining every set each time a model page is loaded is unlikely to be very practical, so the schema design has this data cached within the model record.

Generally **updatestats** needs to be run after each time updates are made to the Wacs collection, but in practice it's probably easier to simply configure it to run once per day. To do this we once again make use of the **cron** timed execution facility (described in the section called “Configuring Automatic Download” of Chapter 13, *The Download System*). In this example, it is being set to run at just after midnight (modulo the advice above about making sure the machine will be likely to switched on at that time):

```
% crontab -e
```

and add the following line to the entry:

```
15 0 * * *      /usr/local/bin/updatestats
```

Tip

As of Wacs 0.8.6, **updatestats** produces a summary on the standard output of what has changed. It mentions who has new sets, new types of sets and what totals they've now reached.

Chapter 16. Simple Tasks In SQL

What is SQL?

SQL standards for *Structured Query Language* and provides a command line interface to the majority of database packages available on the market today. It supposedly uses plain English language phrases but in reality is so tightly structured that you have to know the precise style and format of the requests you wish to pass to it.

Each database package provides a command line SQL interpreter which can be used to interact with the database directly. Most of the time you will not need to use it, but having some basic familiarity with it can help in solving some fairly intractible problems, so it is worth getting an idea as to how to use it.

Table 16.1. SQL Command line interpreters

Database	Command	Typical Usage
Oracle	sqlplus	sqlplus wacs/password
MySQL	mysql	mysql -u wacs -p

In general they will give you a prompt something like `SQL>` and will expect each query to be completed with a semi-colon (`:`) at which point they will execute what you have given them. At the end of the session, typing `quit;` to the `SQL>` prompt will usually log you out again. There are a lot of differences in the way you edit and recall existing queries, but these basic steps are the same for all major database packages.

Example Tasks in SQL

Here are a few examples of simple tasks that you can carry out using the SQL interface.

Adding A New Type Of Attire

As we discussed back in the chapter on the Wacs Set Manager (Chapter 9, *Wacs Set Manager*), there is a potential catch 22 scenario when trying to mark a set as featuring a particular type of attire. This is that the pulldown menu will only offer you types of attire that have been previously used in another set. The usual way of dealing with this is to add a new keyword that will match that word and therefore create the new category of attire for you. If for some reason you didn't want to do this, you can use SQL to create a new attire entry that will appear in the pulldown menu.

For example, maybe you have a thing for ripped denim jeans and want to add a new attire category of **Ripped Jeans** to the possible options. You've just added a set which features ripped jeans and it's set number **14**. The SQL command you would need to update set 14 to this new attire would be:

Example 16.1. Using SQL to set Attire

```
SQL> update sets set sattire = 'Ripped Jeans'
  2  where setno = 14;

1 record updated.
SQL> commit;

Commit complete.
SQL> quit
```

Once this has been done for set 14, **Ripped Jeans** will appear in the pull-down menu in the set manager for subsequent usage.

Creating A Video Download Record

If you want to be able to move a record sensibly between different hosts via the migration tools, it really does need some kind of download record to uniquely identify it. It is a fairly simple matter to create a new download record for this purpose using SQL. The first step is to find out what the highest currently used download number is which can be done with this query:

```
sql> select max(downloadno) from download;

 2672

1 row selected;
sql>
```

With this duly determined, add one to the value (in this case `2672` becomes `2673`) and use that in the SQL query you create. You will also need the related model number, which for the purposes of this illustration will be `2249` for a model called Sarah. The next value is `XYZ` which is the site id for an imaginary site called XYZ. The next value "`V`" is the value for the type of the file - `V` is for Video Clip, `I` is for image set. The next value is the set key which is the number obtained from the URL from the originating web site if possible or if not using the name of the video file itself as here. We then give it a name and repeat the name of the video file as the "archive" that this download record is expected to deliver.

```
sql> insert into download (downloadno, dmodelno, dsite, dtype, dsetkey,
   > dsetname, darchive)
   > values( 2673,2249,'XYZ','V','sarah04.mpg',
   > 'Sarah - Pink Dress - Garden Bench','sarah04.mpg');

1 row added.
sql> commit;

commit complete.
sql> quit
```

Chapter 17. Using SQL: Advanced Topics

Introduction

In this chapter, we're once again going to venture into directly manipulating the database underlying the Wacs system using the SQL Structured Query Language. Before you start in on this, it's important to have had a good look at the Schema Reference section of the Wacs Programming Manual, even if you're not a programmer, as this will give you a good idea of what information the database does actually store. Of particular utility is the list of what the assumed values are for certain key fields are as this should help you avoid upsetting the web-based WACS tools and applications written using the Perl and PHP APIs.

We're going to try and introduce these topics by working through a few examples of tasks you might wish to perform that are not currently covered by web based tools. While in due course there may well be tools that can do these things, it does never-the-less show how you can “step outside the box” within the WACS system.

Merging Models

One scenario that we do encounter from time to time on our large scale test system is where we discover after the event that two models we have listed separately in the database are actually one and the same girl. After making absolutely sure that we are talking about the same person (and hopefully making a note in the **Distinguishing Marks** field in the model record about how we confirmed that), we need to make it clear in our own minds what we're going to do.

This involves making a decision on which of the two model records we're actually going to keep and then checking the one that we intend to delete for any pertinent information, such as biographical data, which is not contained within the record we intend to keep. We can of course easily use the Model Manager (see Chapter 6, ***wacsmodelmgr*** - *The Wacs Model Manager*) to do this.

In the example we're going to work with we've discovered that Dianne from Sapphic Erotica who is our model number *163* is also Lena from ATK Galleria who is our model number *2474*. We've taken the decision to retain her lower model number of *163* and delete the new addition, namely *2474*. We've already copied across the extra biographical information that we had on the “Lena” record, and in fact we decided that we preferred the headshot we had of her from ATK Galleria and so modified that too. These are all very simple to do using the Model Manager, so we'll not cover the exact steps taken again here.

The first step we're going to take is to move the identity record for Lena over to Dianne so that future references to Lena on ATK Galleria automatically summon up Dianne's record. We do this with the following SQL command:

```
SQL> update idmap set imodelno = 163 where imodelno = 2474;

1 row updated.

SQL>
```

For the next step we're going to change the sets that we have marked as having Lena in to being marked as having Dianne in. This is actually quite easy to do because Wacs doesn't depend on the name that we have

in a set record in anyway - it's merely a matter of convenience. Even after Lena's records have become Dianne's, there is no problem with her still being called Lena there. In someways that is desirable because it retains the connection with what she was called on the site where those sets came from. In other ways, it's very confusing as in “Who is this Lena girl who keeps appearing in Dianne's sets?”. We'd definitely recommend adding the name Lena to Dianne's aliases list in the model record because the top of the model page will then say about Dianne “Sometimes also known as Lena”. If you want to change the title of the set, you can easily do this with the Info Manager (see the section called “Managing Additional Information”). As a byproduct of changing the name using the Info Manager, the other fields in the set database that have model info in them will also be updated.

```
SQL> update assoc set amodelno = 163 where amodelno = 2474;

11 rows updated.

SQL>
```

While these two updates have changed most of the user visible entries, the are actually three more changes we could have to make. There are in fact five database tables that make reference to model numbers and all of them do need to be updated to match each other. These are:

Table 17.1. Tables That Reference Model Numbers

Table Name	Description
idmap	Identity Map - record of who she is on what site
assoc	Associations - which model appears in which sets
download	Download - records of where sets came from (establishes set identity)
tag	Tag - a member of a search set (ie that one of the searches found her)
conn	Connections - that she features in one of the connection collections

This is where you have to know your database. The last thing you want to do is to leave a record lying around refering to something that no longer exists - that's extremely bad practice and is almost bound to cause some web application to fall over. Most databases simply won't let you do this - it's known as *enforcing referential integrity* - but unfortnately MySQL 5 in it's standard form is not one of them. This means you could easily delete a model who still has some download records or tags (individual elements of a search) still attached to them. Allowing this to happen is just storing up horrors for later.

As a general rule, it's best to try and change the model numbers in each and every database table that *might* contain them and just accept that in some cases there may well be nothing to do. An update that does nothing is really not a problem to any SQL version we've ever encountered. So we proceed with the next three steps on the basis that it's quite possible that they'll do nothing. Just understand that it's better to execute commands that often seem to do nothing than to end up with the chaos caused by connections left hanging. If you know your database well, and know it reliably enforces referential integrity, then you could skip those you know have no entries in them *but we don't recommend this*.

So we reach the third step, that where we move the download records, and remembering that these are an identification of which set is which in a portable way even when we have no plans to use the download system. In fact, it's good practice for commercial sites to allocate download records to their own sets to ease future server-to-server migrations and the like. Anyway, here is the SQL we need for this:

```
SQL> update download set dmodelno = 163 where dmodelno = 2474;
```

```
14 rows updated.

SQL>
```

Now on to the fourth and fifth steps, which we'll do together which are most likely not to find anything, updating the tag and connections records:

```
SQL> update tag set tmodelno = 163 where tmodelno = 2474;

4 rows updated.

SQL> update conn set cmodelno = 163 where cmodelno = 2474;

0 rows updated.

SQL>
```

So as you can see, in the above example it turned out that we did actually have Lena tagged in no less than four different saved searches, so it's just as well we updated them. The final step is to delete the old model record and complete the model merge process. It's good practice to include a commit at the end of the process - on a good relational database like Oracle, none of the other steps will have actually have been done to the database itself until that final commit is done. The beauty of this is that all of the actions will be taken at exactly the same time and no one browsing the Wacs site at the time will find a half-moved state - the change will literally happen in an instant despite the fact that we've been checking and composing the queries for probably a couple of minutes at least - quite possibly longer if you're reading this document at the same time. The alternative to using commit is to type **rollback;** which will undo all the changes. Unfortunately in MySQL 5 the commit and rollback functions, while present, don't work quite as they should. We can only hope that they will improve with time and further development.

```
SQL> delete from models where modelno = 2474;

1 row deleted.

SQL> commit;

Commit complete.

SQL> quit
```

So there we are. The two model records for SE's Dianne and ATK's Lena are now one and the same and we have a better quality entry for this lovely Ukrainian blonde in our database. All in all a good result.

Part II. WACS Admin Tools Reference

This is the reference manual for the WACS Administration tools which provides an overview of what each tool does and details of the options it accepts via the invocation URL.

Chapter 18. Collection Management

Web Based Tools

- wacsmodelmgr - Wacs model manager
- wacssetmgr - Wacs set manager
- wacsinfomgr - Wacs info manager
- wacsunpackmgr - Wacs unpack manager
- wacsplacemgr - Wacs placement manager
- wacsaddassoc - Wacs association manager
- wacsconnmgr - Wacs connection manager

wacsmodelmgr - Wacs model manager

This web application is designed to duplicate the text-based addmodel script and additionally provide a tool for updating model records and identity maps (jobs that previously had to be done directly in SQL). If you have administrator role defined for your account (see the section called “User Class”) it will appear on the Maintenance menu of the front page and as a link from each model's index page. When used from the model index page, it updates the model's details - used from the Maintenance menu it can add new models and idmaps as well as modifying existing model's records.

When called from the maintenance menu (or directly) it offers you two modes - *Model Update* or *Identity Management* - and three ways of finding the model to work on - by model number, by site and their id on that site or by name. If you choose name, both the main name and aliases in the models schema and the per-site idmap name will be searched and a selection of headshots and details of models with a similar name offered. You are also given the option to create a new model (if in Model Update mode - you can't create an IDmap for a model that doesn't exist!). This will allow you to create a new model or add a new idmap to an existing model. In a future release, it will become possible to edit existing IDmaps as well as model details.

wacssetmgr - Wacs set manager

The Wacs Set Manager, **wacssetmgr** allows the updating of set attributes through a web-based GUI. You get to it by clicking on the "rating" link on the model pages - this link is only offered to users in the administrator list. It can alternatively be invoked directly with the set number in the URL, for instance to edit details for set 12345, you'd give the URL ending with **wacssetmgr/12345** . This presents a startforward web form through which you make your changes, click **Summarise Changes** and you'll see a list of what you've changed; if happy with it, click on **Commit Changes** to make those changes in the database.

wacsinfomgr - Wacs info manager

The Wacs Info Manager, **wacsinfomgr** is effectively a partner for the Wacs Set Manager that provides for editing the more nuts and bolts fields within the sets table. One of it's major features is the ability

to rename and relocate sets, but it covers several other areas including official and additional icons, textual descriptions and USC 2257 attribution management. It is invoked simply using the set number as a parameter in the URL, so to edit set no 12345 you give a URL ending with **wacsinfomgr/12345**.

wacsunpackmgr - Wacs unpack manager

The Wacs Unpack Manager, **wacsunpackmgr** is used to unpack downloaded zip or video files so that they can be examined, described and imported into the Wacs system. This can be done by one of three methods: taking a downloaded set from the Wacs download area, by uploading a zip or video file through a web browser, or by copying it from a specified directory on the Wacs server itself.

wacsplacemgr - Wacs placement manager

The Wacs Placement Manager, **wacsplacemgr** is used to place the unpacked set into the wacs image or video clips archive, naming it appropriately for it's content. The name you give it is then scanned for keywords which are used to make an initial first guess at the appropriate attributes to mark the set up with. These can be modified and improved using the **wacssetmgr** - Wacs Set Manager at a later time.

wacsaddassoc - Wacs association manager

The Wacs Assocation Manager, **wacsaddassoc** is used to add new associations between models and sets. Its functionally is almost identical to the text-based **addassoc** utility but is somewhat easier to use being web-based. **wacsaddassoc** offers you three choices for the method of selecting the target set - a specified set number, a pull-down list of recently added image sets and a pull-down list of recently added video clips. Similarly it offers you three ways to select the model to be associated with the set: by specifying her model number, by selecting from a list of recently added models, or by searching for her by name.

wacsconnmgr - Wacs connection manager

Newly introduced in WACS 0.8.5, the **wacsconnmgr** provides a long needed interface to creating and adding to connections. The initial version is somewhat simplistic but never-the-less allows connection sets to be created quickly and easily. It is very similar in operation to **wacsaddassoc** in offering you a number of pull-down menus to select from recently added sets or models. It also provides for direct entry of set numbers and model numbers.

Chapter 19. Lookup Data Management

Web Based Tools

- wacsvendmgr- Wacs Vendor Manager
- wacsphotmgr- Wacs Photographer Manager
- wacskeywordmgr- Wacs Keyword Manager
- wacsusermgr- Wacs User Manager
- wacsuserlist- Wacs User List
- wacsdnlmgr- Wacs Download Manager
- wacsdnllist- Wacs Download Status List
- wacsdnlframe- Wacs Download Frame
- wacsdnlnext- Wacs Download Next Prompter

wacsvendmgr - Wacs Vendor Manager

The Wacs Vendor Manager, **wacsvendmgr** is used in the maintenance of the vendor database which is used by some of the model and set display pages to profer a link back to the original site and heavily in the download infrastructure. It can display, update or create new vendor records in the vendor database. When first invoked, it will offer a choice of vendor sites which have been preloaded into the system using the the console tool vendor table population command, vendors.xml. A number of sample vendor records are distributed in the XML file, `vendors.xml` which is read by the the section called “The **wacspop** command” command. This is typically done automatically by the **easyinstall** command when WACS is installed, or manually if the RPM packages were used.

The *Add A New Vendor* option gives a simplifed form for the basic details; once created the record should be modified to add whatever extra information is also known. Please contribute details for any good quality sites you know of to the authors to be included in a future version.

Please note that the URLs stored in the vendor database include a number of keywords sandwiched by hash (#) symbols which are substituted for by WACS before calling the described page. The current recognised keywords are:

Table 19.1. Substitution List For Vendor URLs

String	Data Source	Description
#NAME#	iname from idmap	The model's name on this site
#KEY#	ikey from idmap	The key for this model on this site (ie model no or reference)

String	Data Source	Description
#ALT#	ialtkey from idmap	The alternative key for sites like atkexotics and atkgalleria that use two keys.
#SETKEY#	dsetkey from download	The relevant set's key or reference number.
#SESSIONKEY#	unique number for sessions	some sites use Dynamic DNS to give a unique short-lived URL for set retrieval - this allows that URL to be passed downwards.
#MODELNO#	the WACS model number	designed for site prototyping and other purposes where the site may be self-referencial.
#SETNO#	the WACS set number	designed for site prototyping and other purposes where the site may be self-referencial.

wacsphotmgr - Wacs Photographer Manager

The Wacs Photographer Manager, **wacsphotmgr** is used in the maintenance of the photographer database which is used by many of the model and set display pages to profer to work by the same photographer. It can display, update or create new photographer records in the photographer database. When first invoked, it will offer a choice of existing defined photographers, and an option to add a new one. As with vendors, there is a corresponding command line utility called photographers.xml which will import the provided `photographers.xml` sample collection.

wacskeywordmgr - Wacs Keyword Manager

The Wacs Keyword Manager, **wacskeywordmgr** is used for viewing, adding and amending keyword look-up rules used by the **updateinfo** command in determining the automatic values for a named set. It does not take any URL arguments, you merely scroll through the list presented, tick the adjacent tick box and then click on **Edit** or **Delete**. To add a new keyword, simply click on **Add**. Scores for each attribute that is infered from the keyword start at *1* for least likely up to *9* for most likely. To investigate how a given set of attributes produced the result it did, set the debug level for *`tool_updateinfo`* to something like `5` and the score card used will be displayed.

wacsusermgr - Wacs User Account Manager

Note

This is new in WACS 0.8.5 - it is not available in previous releases

WACS has two options for how to authenticate users for access to the collection - *hostauth* which authenticates users by asking the web server host itself and *database* which uses the user database table within the Wacs database itself. The usual default for Wacs has been to use the **hostauth** method which works well for both small systems with local users and for bigger systems with distributed authentication like NIS, LDAP or even Active Directory. Where this is not so useful is for commercial sites where all you want your users to access is the WACS-managed collection and nothing else. Here the extra overhead of maintaining a second independant authentication mechanism is a positive boon for security and demarcation.

The **wacsusermgr** webapp by default creates new users with the next available userid, but if passed a user id on the URL will allow you to edit an existing userid. With this application you can manage user accounts and access rights with reasonable ease.

Tip

What you will notice about wacsusermgr is that some of the functionality is not available - these pieces handle aspects such as address and contact info, automatic sign-up, renewals and referral commissions. The WACS developers do want to see some return on their work and felt the best way to do this was to offer an added value product for the really serious commercial sites and felt the absense of these kind of features was no great loss to the regular free software community - the system is fully functional without it. As a result, a commercial add-on package including the plug-in for wacsusermgr, auto sign-up and customer relationship management features called *WacsPro* is available through Bevtec Communications Ltd [http://www.bevteccom.com/]. All the relevant data structures we use *are* distributed as part of the standard Wacs distribution, so you are at liberty to write your own plug-ins if you wish to. The Wacs developers will even accept and welcome contributions back into the Wacs source tree in this area if people wish to submit them.

wacsuserlist - Wacs User Database Account List

Note

This is new in WACS 0.8.5 - it is not available in previous releases

This is a straightforward index of the current user accounts along with a link to allow you to edit each of them using wacsusermgr. **wacsuserlist** is a bit simplistic and has no search features but should be very usable up to about a hundred users or so. A more advanced version with search, auto-reminder and data analysis features is part of the *WacsPro* toolkit available as a commercial add on from Bevtec Communications Ltd [http://www.bevteccom.com/].

wacsdnlmgr - Wacs Download Manager

The Wacs Download Manager, **wacsdnlmgr** allows the resolution of some of the most common problems with download records. To invoke the download manager for download record 1234, call wacsdnlmgr with `wacsdnlmgr/download1234.html`. The wacsmodelpage will automatically link to this page for users with role “admin” in the access control lists.

wacsdnllist - Wacs Download Status List

This gives an overview of the current state of the download activity over the last two weeks. It's primarily informational although it does link to the download manager to allow modification of the download records.

Note

Both **wacsdnlframe** and **wacsdnlnext** are new in Wacs 0.8.4 and are not available in previous releases.

wacsdnlframe - Wacs Download Frame

For some web sites, due to their security measures, it is not possible to perform an automatic download. For these sites, it is possible to have the Wacs download system parse a saved version of the model's index page to compare the set numbers the system knows about with the new version of the page to see if any new sets have been added. If there are new ones found, in most cases the information found is incomplete - mostly it is missing the actual URL (Web Address) for the downloadable zip or movie file - making the download record incomplete. The **wacsdnlframe** web application splits the web page into two halves - the top half has some input fields; the bottom half has the relevant set page from the site displayed in it.

To make use of **wacsdnlframe** you find the link to the zip or movie file, do a right mouse click on the link and choose `Copy Link Location`. You then move up to the input field in the top half of the page, and simply paste the saved URL into it. You then click on the "Complete Download Record" button and the URL for the set is saved into it's download record and the status updated to show that it is now a complete record and usable for a download.

wacsdnlnext - Wacs Download Next Prompter

The **wacsdnlnext** command provides a web-based alternative to the standard command line download tool **getarc** by creating a web page containing currently pending downloads. It gives both where a set should be saved and a link to each set to be downloaded. It also provides a tick box which can be ticked when each set has been downloaded - it will then check for the presence of the downloaded files and if they're present, mark the download record as pending unpacking.

The recommended sequence for using **wacsdnlnext** is to select the full pathname where the file should be saved, copy it into the paste buffer, then right click on the link on the archive name and choose `Save Link As` - in the dialog box that pops up, you then paste the destination directory from the paste buffer. Finally you tick the `done` box once the download is complete. Once you've done all the downloads you intend to in this batch, you can click on the `Update To Do List` button at the bottom and the ticked downloads will be checked and if present and of a reasonable size, their download records will be updated to indicate that they have been downloaded.

Tip

There is no problem with downloading out of sequence. You can do all the images first for instance and then leave the videos running while you do something else. Just tick the ones you've done and **wacsdnlnext** will do the rest.

Chapter 20. Database Population Tools

Command Line Tools

The task of the database population tool, **wacspop** is to read the XML files which come as part of the distribution and use the contents of those files to populate some of the database tables with some sensible initial values. At present it will create any new records that have not been previously seen but will not update existing records in any way.

Note

The **wacspop** is new in WACS 0.8.6 and replaces three previous seperate commands - **vendpop**, **photpop** and **keywordpop**. It is not 100% compatible with the older files as they had insufficent data in them for a single combined loading program. It is easy enough to convert the files, see the section called “Updating XML Files For **wacspop**” below.

The wacspop command

There are currently four XML file types that understood by the **wacspop** command and they are:

- vendors.xml- Vendor database initial load
- photographers.xml- Photographer database initial load
- keywords.xml- Keyword database initial load
- attrib.xml- Attributes database initial load

vendors.xml - Vendor database initial load

Here **wacspop** reads the vendors.xml file and populates the vendors table with some useful example site (aka vendor) description records. These provide examples on how to configure the automatic download system as embodied in the chkmodel and getarc programmes, and should be directly usuable for any of the sites listed as soon as you add an appropriate username, password and subscription expiry date and set the subscription active.

photographers.xml - Photographer database initial load

Here **wacspop** reads the photographer.xml file and populates the photographer table with some useful example records. These will typically be photographers whose work has been encountered on a number of different websites and who have a “corporate” presence on the web. An initial selection of the most active circa 2009 is included in the XML file.

keywords.xml - Keyword database initial load

Here **wacspop** reads the keywords.xml file and populates the keyword table with some example keyword settings and priorities. There are about one hundred and fifty entries in the distributed file, which should

provide some ideas on how the system works. Basically each keyword has entries for attributes, attire, locations and set types that it may provide clues to - each one of these is scored. The trick is to set the scores such that the presence of a sofa implies it's in a lounge, unless something else which binds stronger is also there. So if a photographer drags a sofa out into a garden for a given shoot, the "Garden" means Garden, Outside and takes preference over the "Sofa" means lounge rule. Take a look at Chapter 4, *Naming Sets In WACS* for more information and also use the **wacskeywordmgr** to have a look at the definitions we have already created for you.

attib.xml - Attributes database initial load

This is a bit of an odd one as the attributes table isn't actually used by any other application in Wacs 0.8.6; this is here in preparation for the upcoming Wacs 0.9.0 release and contains incomplete sample data to aid in the testing and development of the Wacs 0.9.x code. Still, the files are here to serve as a guide as to what is coming in Wacs 0.9.x.

Updating XML Files For wacspop

With the introduction of **wacspop** the file format of the XML populate files has changed slightly. This is simply down to the need for the file to identify which schema it relates to and is done in the format of the other WACS XML files. The following attributes need to be added:

Table 20.1. New XML Elements For wacspop

Variable	Value	Comments
coreschema	vendor	Schema name
version	WACS_POPULATE_VERSION_1	Will be 2 for Wacs 0.9.x
revisiondata	09-APR-2011	Date data last updated

At the simplest level, to convert an existing XML populate data file for the vendor database to work with **wacspop**, you edit the file to add these three lines just after the `<vendors>` line at the top of the file:

```
<coreschema>vendor</coreschema>
<version>WACS_POPULATE_VERSION_1</version>
<revisiondate>30-JUL-2011</revsiondate>
```

Index

U

V

W