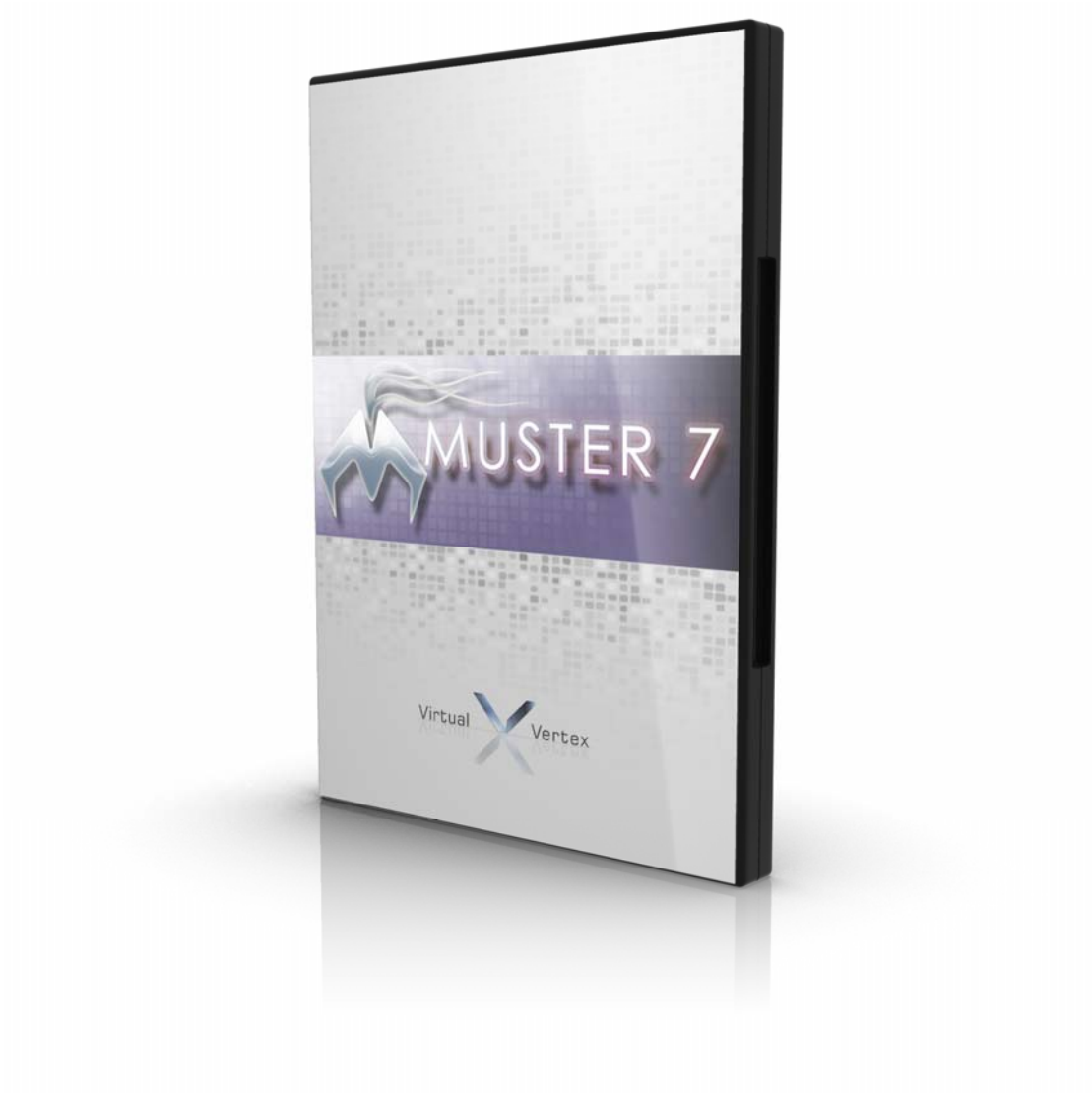
User manual



Virtual Vertex

Revision 1.7 – Muster 7.0.3 – 3 May 2012

Virtual Vertex

Email: vinfo@vvertex.com

Web site: www.vvertex.com

The information in this document is subject to change without notice and should not be construed as a commitment. Virtual Vertex assumes no liability for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

Copyright © 2000-2012 Virtual Vertex.

All rights reserved.

Muster is a trademark of Virtual Vertex.

Microsoft, Windows 2000, Windows XP and Windows NT are registered trademarks of Microsoft Corporation.

UNIX is a trademark of AT&T.

Pentium is a trademark of Intel Corp.

Apple, Macintosh and Shake are registered trademarks of Apple Computer, Inc.

Maya is a trademark of Autodesk.

Softimage and Softimage|XSI and Softimage|3d are trademark of Autodesk.

Lightwave is a trademark of Newtek inc.

After effects is a trademark of Adobe inc.

3D Studio Max is a trademark of Autodesk.

Code credits:

Muster uses a subset of the OpenSSL security library.

Licensing and code available at http://www.openssl.org

Muster uses the sqlite inprocess database.

Licensing and code available at http://www.sqlite.org

Contents

Using this document

This document provides a description of the Muster software suite, its installation procedure, derived tasks and workflows for using your hardware and software equipment.

Muster has been created as an external plug-in for professional 3D content creation packages. This manual gives an in-deep explanation on all of its features but doesn't explain any argument related to 3D production or third-party software specific tasks.

This document assumes that you have a good working knowledge of 3D common procedures and terms meaning as well as a basic knowledge of network environment and basic configuration procedures. It also assumes that you are familiar with the basic concepts related to using operative systems, Windows NT Administration, Unix shell operations and simple management.

Technical support

Muster includes unlimited free technical support via email. With technical support, you can get help when installing or using Muster. We do not normally provide telephone support or training.

Technical support can be requested on our e-mail address vvsupport@vvertex.com. We suggest you to check the validity of the e-mail address visiting our site at http://www.vvertex.com and clicking on the **support** section.

Muster includes free upgrades for a period of 30 days after the date of the purchase. Releases of patches are available for free to all the registered users of the patch family version.

At the time of this writing, Muster is sold only in electronic form. We suggest you to save in a safe place any installer and patch for your registered version.

Getting started

Muster is a suite of applications specifically designed to manage complex and multi-platform render farms. In the digital content creation industry, the term render farm is used to describe a set of computers fully or partially dedicated to the creation of digital images. This process is commonly called "rendering" and the resulting images "rendered images".

The rendering is typically the last stage in 3D animation and 2D compositing software workflow and is a CPU intensive task; the time required for the generation of a single image can range from minutes to hours. In video or motion picture productions where contents are made by several minutes of digital effects, several days may be required to complete the production phase .

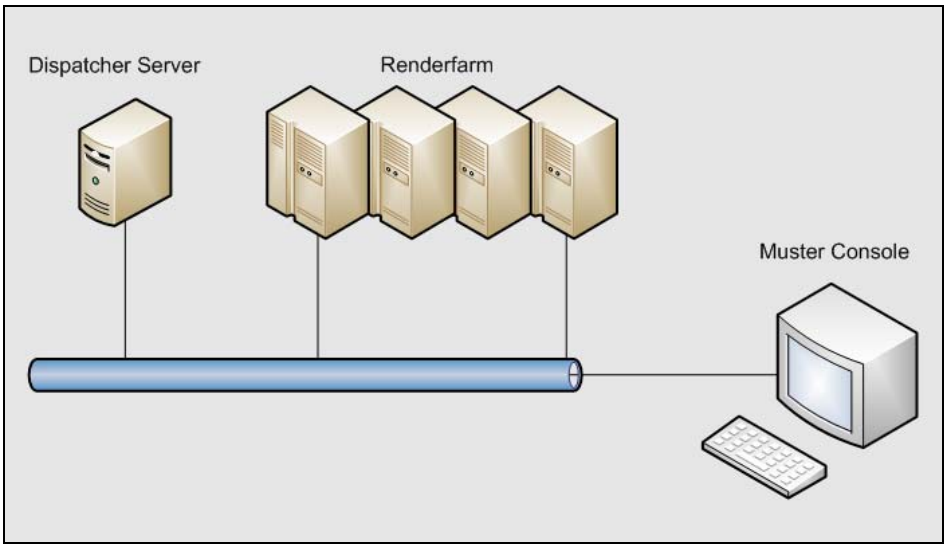
Spreading the render of image sequences on a set of computers dedicated to this task, helps reducing rendering times but tracking their status and managing errors can be really complicated. It's here that comes Muster. Muster is a client/server suite of applications that provides production's supervisor the right tools to manage their render farm made by hundreds or a bunch of computers.

Muster relies on a client/server architecture. A centralized server takes care of controlling several slave modules. There are two client modules groups:

1. The Render Client service installed on every computer in the render farm allows Muster to manage the computer for you. It basically receives commands from the Dispatcher service, starts and stops the Rendering process on behalf of it.

2. The management/notification modules, that connects using socket connections to the Dispatcher service and allows monitoring and management on Muster itself.

   The next figure shows a typical implementation:



This is a complete list of the modules available in a Muster installation:

- Dispatcher service
- Render client service
- Notificator applet
- Console, real time graphical frontend to Muster
- Direct Muster connector for Maya version 6 or greater (tested up to version 2010)

This is a brief explanation for each module:

Dispatcher Service: The Dispatcher service is the most important component of the whole suite. It manages connections with every client, retains a job queue, accepts commands from external client applications and sends notifications to the users. It may run in the background as a Windows Service or as UNIX daemon.

Render client Service: This is a service that must be installed on each render farm node. It retains a connection with the dispatcher service, waits for server commands and starts the scheduled render jobs. It may run in the background as a Windows Service or as a UNIX daemon.

Notificator: This is a very simple component that gathers notifications from the dispatcher when jobs are completed. Based on the user preferences, it will warn logged users about jobs completion. (Windows only)

Console: The Muster console is a complete graphical front-end to the Dispatcher service. It's basically the Muster most used module. It allows the user to schedule jobs, change their order, manage and configure connected render clients and configure the dispatcher service.

Connectors: Those are plug-ins for external software allowing job submissions from within the software's GUI. At present the connector is available for Maya 6.0 or greater (tested up to version 2010).

Choosing your server

An important concept to take into account at the beginning of your installation is choosing the right machine to dedicate to the dispatching process. Before going further, we'll take a look at the roles of the dispatcher service and the way it interacts with your network, distributed file systems and the hosts in your render farm.

As seen before, the main role of the dispatcher service is to retain a pool of persistent connections with your render farm hosts. The connections are performed using the TCP/IP protocol. The decision of using TCP involves several pros and cons. This guarantees messages order and consistency but requires a higher network usage than UDP. This configuration requires a clever pre-setup brainstorming and in some cases, changes to the network routing and configurations.

Before going further, we must clarify an important concept. The data flow sent between dispatcher and render farm hosts is composed by synchronization and management messages. **This means that the files used in your scheduled jobs are not transferred through this connection**.

How content composed by scene files, textures and renderings is transferred across hosts? F**iles are read and written using the standard network shares**. This means that a machine that has enough disk space must be dedicated for this task. Depending on your network configurations and your bandwidth, you can dedicate the same machine running the dispatcher service or a big file repository running on a file server with a big disk array.

Of course choosing the OS, disk configurations and physical layouts depends on personal preferences and budget limits. The flexibility of this system allows further expansions and rearrangements even during productions phases. Just keep in mind those important key concepts:

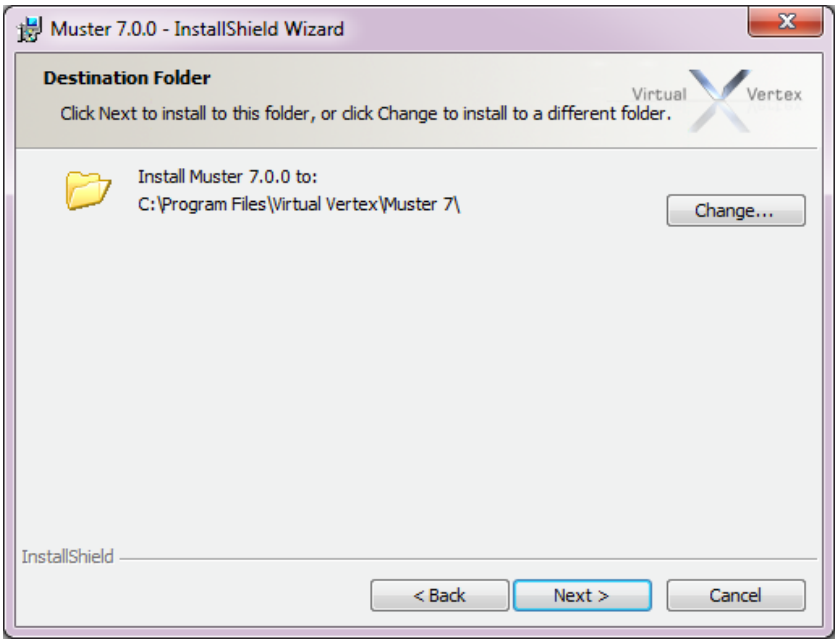
- If you choose a Windows machine to host files, keep in mind that each host accessing files uses a new connection to the file server. Microsoft operating systems that are not part of the SERVER family, allows only 10 concurrent connections to a shared folder. In a production environment, sometimes a few connections remains opened and this means that we strongly suggest using a maximum of 7-8 client hosts.

- When working on a mixed platforms environment, you should be sure that the shared file system is exportable using a network protocol supported on every system. A typical implementation could be made by client tools (like Samba clients to access a Windows shared folder from UNIX machines) or server services like NFS for Windows. **Our suggested solution is setting up a dedicated file server running Linux with SAMBA and NFS support**.

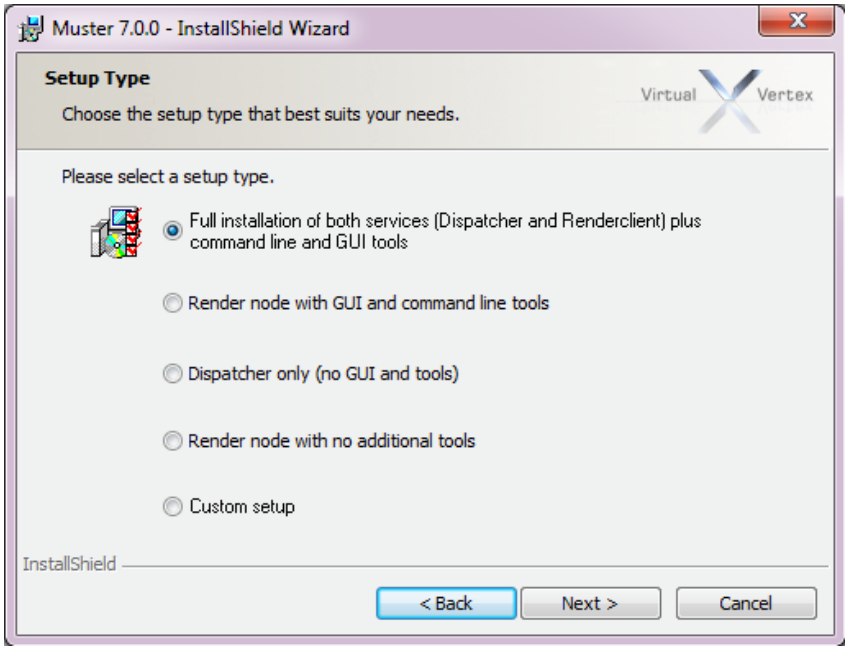
- If available, switch your network to Gigabit or better technology.

Windows installation

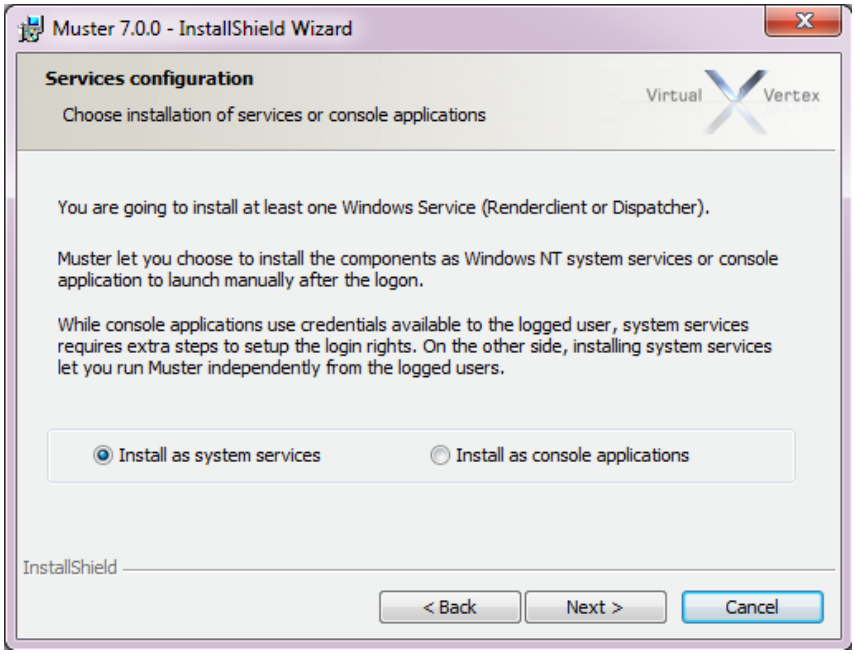
Muster comes with a standard **Windows Installer** package. By clicking the **.msi** file you got from the web, you start the installer as shown below.



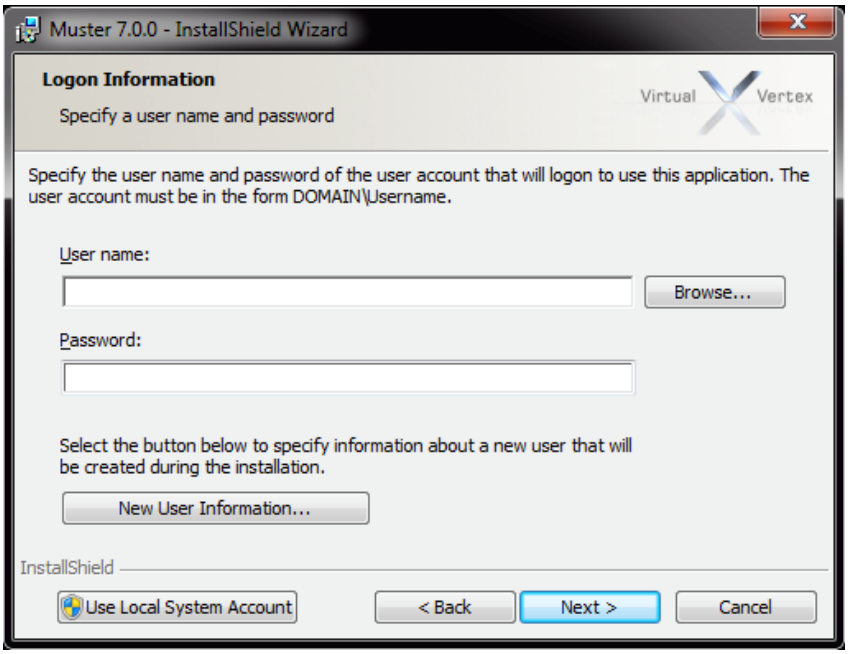
In the first you're asked to specify the final destination for Muster files. This may be left untouched unless you need to install it in a different location.



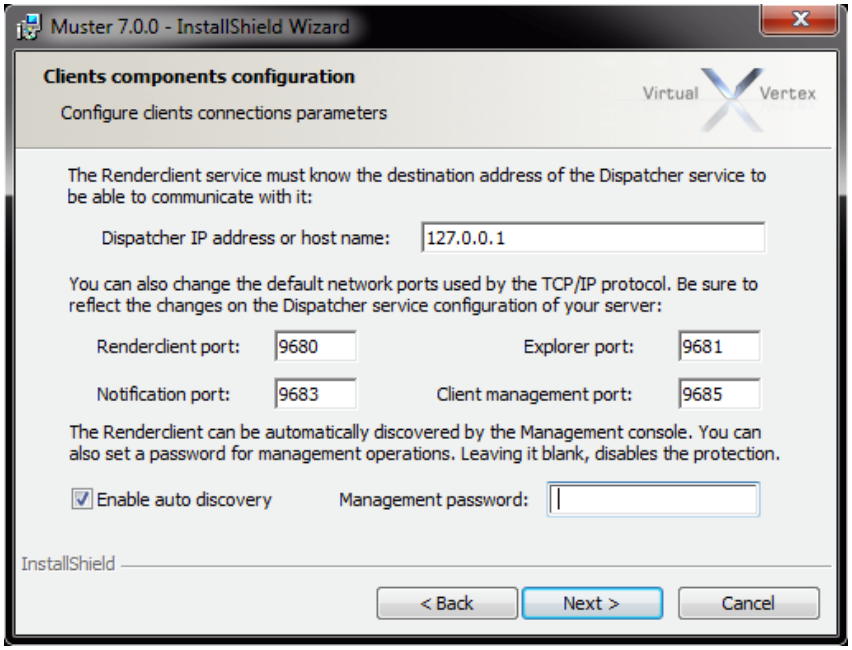
The next steps asks you to select which Muster components will be installed. Select the option that best suits your needs.



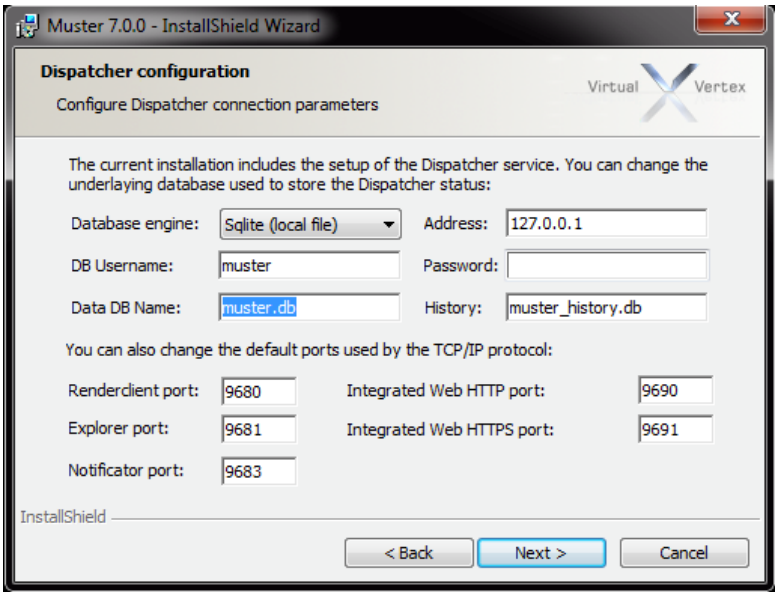
Muster is able to run as a system service. A system service is an executable application that runs into the background. If you choose to start Muster as a system service, you won't need later to logon on the machine and start the software manually, it will be up and running as soon as your boot sequence reaches the login window. Having the software installed as system service requires an additional step later in the setup sequence. Considering it won't be bound to any logged user, a service requires its own user setting in order to access the network and be recognized by servers. Installing Muster as a system service is more complicated compared to the standard installation, but allows you to have independent modules.



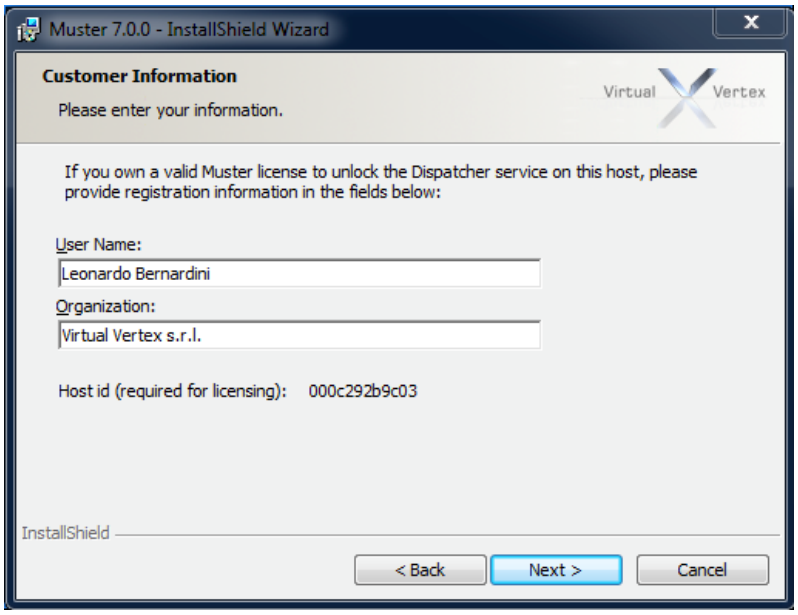
If services is your choice, this is the window you'll get during the installation process to configure the user assigned to the Muster services. You can either configure a user now or choose **Use Local System Account**. This will install the services with a **default** user. The setup will continue, but this user won't be probably able to access the network content in most cases, so you'll need to configure the user later as explained in the last part of the setup section of this manual.



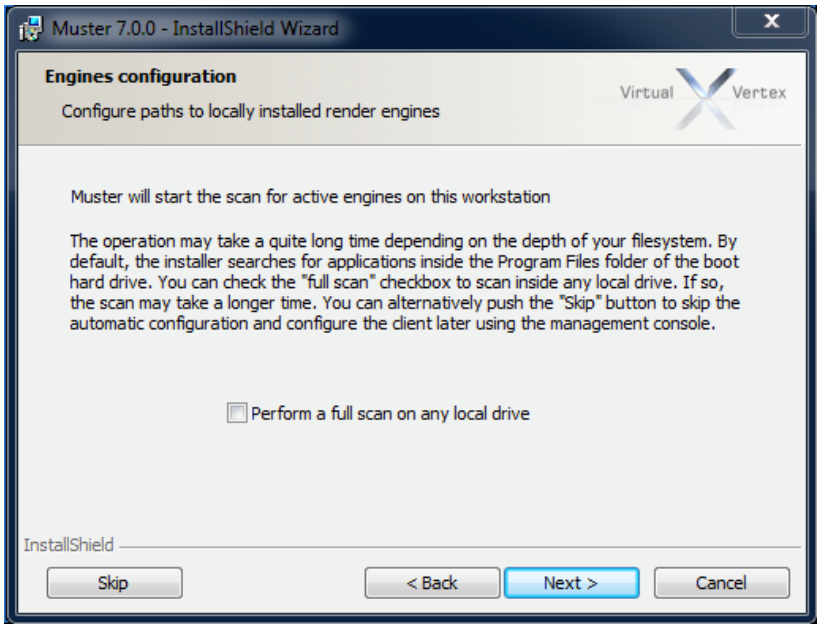
If you choose to install the render client service, the installer will prompt you for changes in the default configuration. Unless you've a deep knowledge of network topics, the only field you should change is **Dispatcher IP address or host name**. This is where you need to enter the IP Address or the host name of the host where you installed the Dispatcher service. If you're installing both the Dispatcher and the Client, just type the local system name in the IP Address field.



If you choose to install the dispatcher service, the installer will prompt you for changes in the default configuration. Unless you've a deep knowledge of network topics, you should not change the network ports settings. In this window, you can also choose the way the Dispatcher stores its data. If you want to store the data in an external database, change the settings according, otherwise, leave them unchanged.



If you choose to install the dispatcher service you can track down the **HOST ID** prompted by the installer from this window, you may need it later to request the license through our website.



If you choose to install the renderclient service, you can tell the installer to look for known applications in your local file system and registry. Remember that this process may require a certain amount of time and will scan in know locations on your system drive or your entire file system or your entire file system, if you choose "**perform a full scan on any local drive**". You can always skip this step and configure the paths to the render engines later through the **Console**.

Pressing the "next" button, the installer will start the installation process. It will copy the files in your file system, starts the services if required, and then completes. If you did not configure a user for the services, read carefully the next section about services users rights.

Defining Windows Services user rights on your network

A key concept of Muster is the understanding of how it works on the network and the steps you've to perform to avoid common pitfalls related to network access from the services.

When you work on a shared folder, the process that requests a file must supply valid credentials to the file server. "Credentials" means at least a valid username and password pair that's valid on both side of the communication (the client running the process and the server hosting the files).

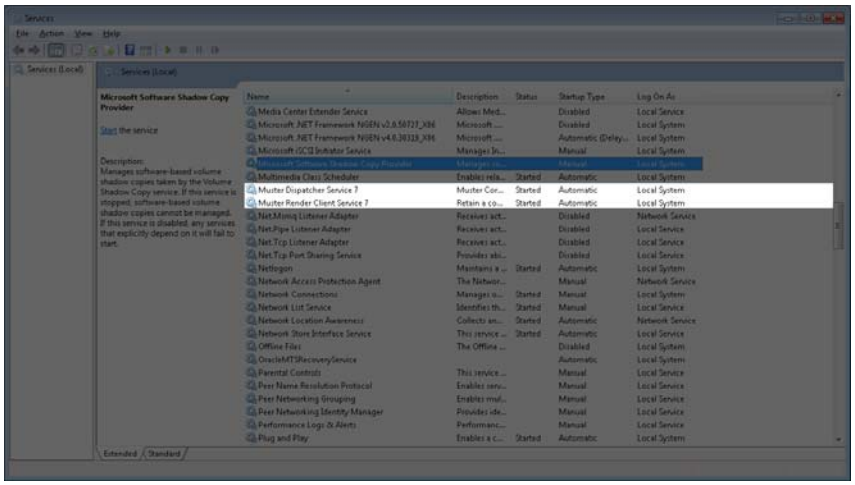
When you access files on your network, the entire process is hidden to the user. The client sends the login and password pair of the user running the current process to the server.

To gain full access, a user should be successfully authenticated by the server (this means that he should be available on the server's user list and its password should match with the one stored on the server).

In case of failure, depending on network policies, the system could pop up a dialog that asks for the user password and retries the connection.

The Muster Render Client Service must be able to access the network resources but its behaviour is a bit more complex. Windows services are basically software modules that run independently from the logged user.

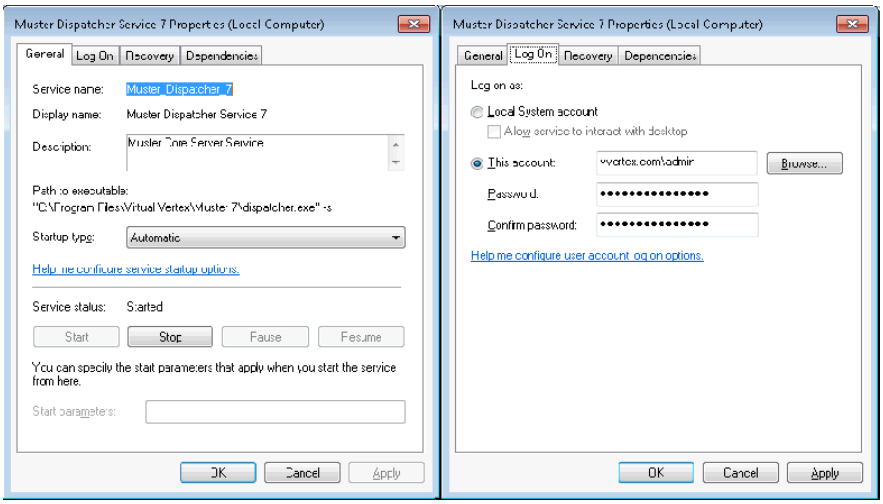
The next figure shows the services installed by Muster. Depending on installed features, you will find at least one of the services listed below:



Services run even if no user is currently logged on the machine. For this reason, they have a dedicated configuration window where is possible to specify which user the service have to impersonate.

When a service impersonates a particular user, it runs as it was started from the user desktop and supply user credential to the various file servers.

The next picture shows the services properties window where you can configure the service properties and the service log on properties:



During the setup process, Muster installer will prompt for the username and password used to install the services or alternatively you can select to use the local system account. Even this account will let you to start the services, it won't give any rights to the services to access the external network. That's why you may need to manually create a dedicated account.

There are two scenarios: creating the user on a Windows Domain or creating it on a Windows Workgroup. You should refer to your network Administrator to obtain information about your current setup.

This is a description of the basic differences when setting up Muster on Domains or Workgroups:

- **Workgroup based networks**: A Workgroup is basically a group of Windows computers that must share the same user accounts to allow access through each one. This means that there's no server hosting a common list of users and their relative permissions. When a user tries to access a folder on another host, the authentication function simply checks that the user exists on the remote machines and that its credential matches. For this reason, the Muster installer, when detects a workgroup, creates the same user account of every host with same credentials and assign it to the Administrator group of the machine. In this way, each host can authenticate the Muster account and allow read/write access to the shared folders.

- **Domain based networks**: The situation on a domain network is a bit different and much easier. A Domain is a group of computers that shares a server that hosts information about user accounts and permissions (the Primary Domain Controller). As you can image, setting up Muster in this configuration requires to just create the user on the Domain Controller. Automatically all the machines will recognize the user and will give access to it. The only important thing to be aware is that each machine must recognize the user as a member of the **local Administrator group**. This is different from the Domain Administrator group. In you decide to customize user creation or the Muster installer is

unable to perform the setup for some domain policy, you should do this setup manually on each machine even if you assign the user to the Domain Administrators user group.

When you perform the default installation, Services should be already configured by the Muster installer. If you skip the user creation, depending on your Windows version, you should open the Services Control Panel applet and assign log on information to the installed services.

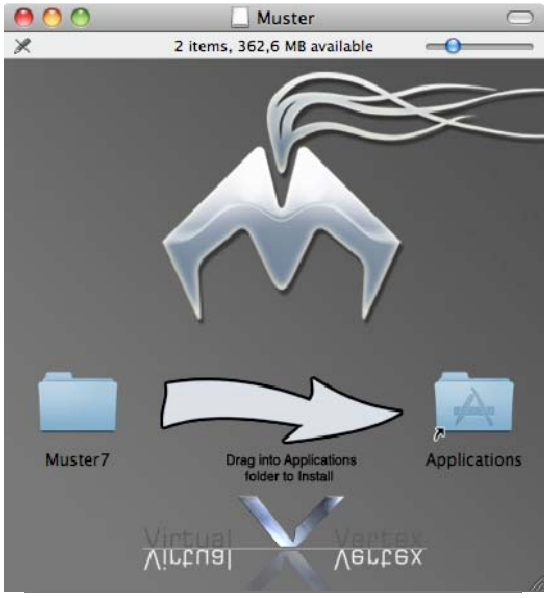
After configuring the user account the next step consists in performing a final check to verify that the Muster account is able to access your shared folders. From now on, we'll assume that we have created a shared folder, located on a file repository server called **MASTER** and that the server exports a share called **RENDERFARM**.

Logon a random machine where you've installed Muster and configured the account. Instead of your personal account, use the Muster account. If you've created a custom account, use it.

After logging in, try to open the shared folder (i.e. \\MASTER\RENDERFARM) , create some files inside it, try to rename them and finally try to delete them. If everything was successful, you have well configured well the user account and it's able to access your file server.

Apple MAC OS X installation

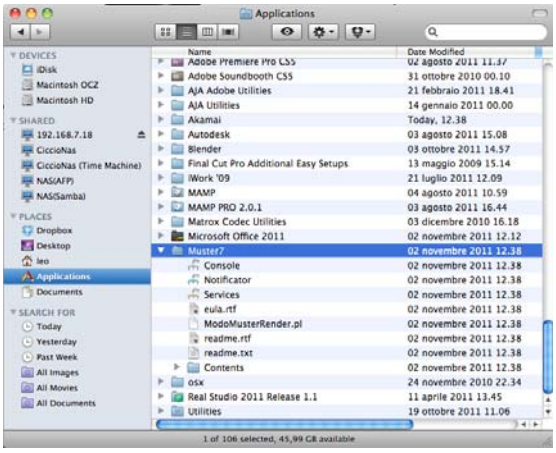
Muster comes on the MAC OS X platform as a self-mounting .dmg compressed image. Once you double-click the .dmg image, the following window will appear:



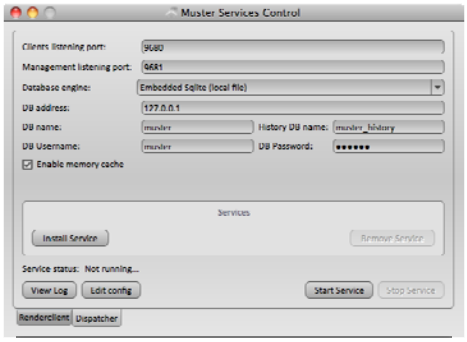
If you're not going to upgrade your existing installation, the only required step to install Muster n the MAC OS X platform is dragging the **Muster** folder inside your **Application** folder.

If you're going to upgrade an existing installation, please be sure that your existing services are stopped through the **Services** applet.

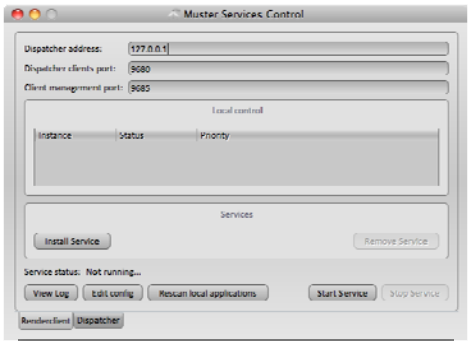
Once you copy the **Muster** folder inside the **Application** folder, locate it using the Finder:



then start the **Services** application that lets you start and stop the services, and install them as persistent system services:



From the **Services** applet, you can even configure some basic parameters before starting the services like the Dispatcher database and the network ports. If you're not going to install the **Dispatcher,** just move to the Render client tab.



To install the Render client persistently, just click the **Install Service** button.

You can tell Muster to scan the local workstation for installed batch renders just clicking the "**Rescan local applications"**. Remember that this process may be time consuming and will scan in know locations on your system drive or your entire file system if you choose "**perform a full scan on any local drive"**. You can always skip this step and configure the paths to the render engines later through the **Console**.

After the service installation and configuring the engines, check the IP address of your Dispatcher and then click **Start Service.**

The services control panel applet can be minimized. It will stay in the Finder bar and can be recalled on demand right clicking on the icon.

Linux installation

Muster comes on the linux platform as a gzip compressed tar file. Assuming you downloaded Muster on a temporary folder inside your home folder, open your shell and type:

```
tar –xvf Muster7.0.0.x32.linux.gz
```

This will explode the archive into your current directory. Now you can start the textual installer with the following command:

```
sudo ./install
```

The installer will prompt you for the modules you're interested in and copy the required files in the installation directory:

```
Muster 7 Copyright 2000-2009 Virtual Vertex
-------------------------------------------------
Welcome to the installation script for Muster 7
The script will attempt to install Muster on your local filesystem
Pay attention. The destination directory will be overwritten!
Where do you want to install Muster?[/usr/local/muster6]

Creating directory /usr/local/muster6

Do you want to install the Dispatcher service?[yes]
Do you want to install the Renderclient service and the client components?[yes]

Copying content to /usr/local/muster6...
```

If you choose to install the Dispatcher service, you'll be prompted for Dispatcher installation properties:

```
Do you want to configure the Dispatcher service?[y]

Do you want to start the Dispatcher engine on service startup?[y]
Do you want to enable the Dispatcher integrated web server?[y]
Which database engine you want to use for the Muster queue?[sqlite/mysql]
Please provide the filename for your local database[muster.db]
Please provide the filename for your local history database[muster_history.db]
Do you want to configure Dispatcher network ports?[no]
Writing Dispatcher configuration file to /usr/local/muster6/dispatcher.conf
```

If your choose to install the Render client service, you'll be prompted for Render client installation properties:

```
Do you want to configure the Renderclient service?[y]

How many instances do you want to spawn?[1]
Please insert the IP address of your Dispatcher service[127.0.0.1]
Please insert the Renderclient network port configured on your Dispatcher
service[8680]
Please insert the Renderclient network port used for management
connections[8685]
```

```
Do you want to let the client broadcast its presence on the network to allow
automatic discovery?[y]
If you want to protect incoming management connection with a password, type it
now[]
Do you want to start the Renderclient service in paused status?[y]
What selection priority do you want to give to the Renderclient?[1]
Where do you want to store the processes log files?[/usr/local/muster6/logs]

Writing Renderclient configuration file to /usr/local/muster6/rc.conf
```

After configuring the Render client properties, the installer will ask you to scan the local file system for installed batch renders. Remember that this process may be time consuming and will scan in know locations on your system drive or your entire file system if you choose "**perform a full scan on any local drive"**. You can always skip this step and configure the paths to the render engines later through the **Console**:

```
The Renderclient configuration file needs to be configured with the paths to
external processes.This may be done automatically by this installations script
but it may take a long time depending on your filesystem.

Do you want to scan your local filesystem for known applications and configure
the Renderclient templates?[y] n
```

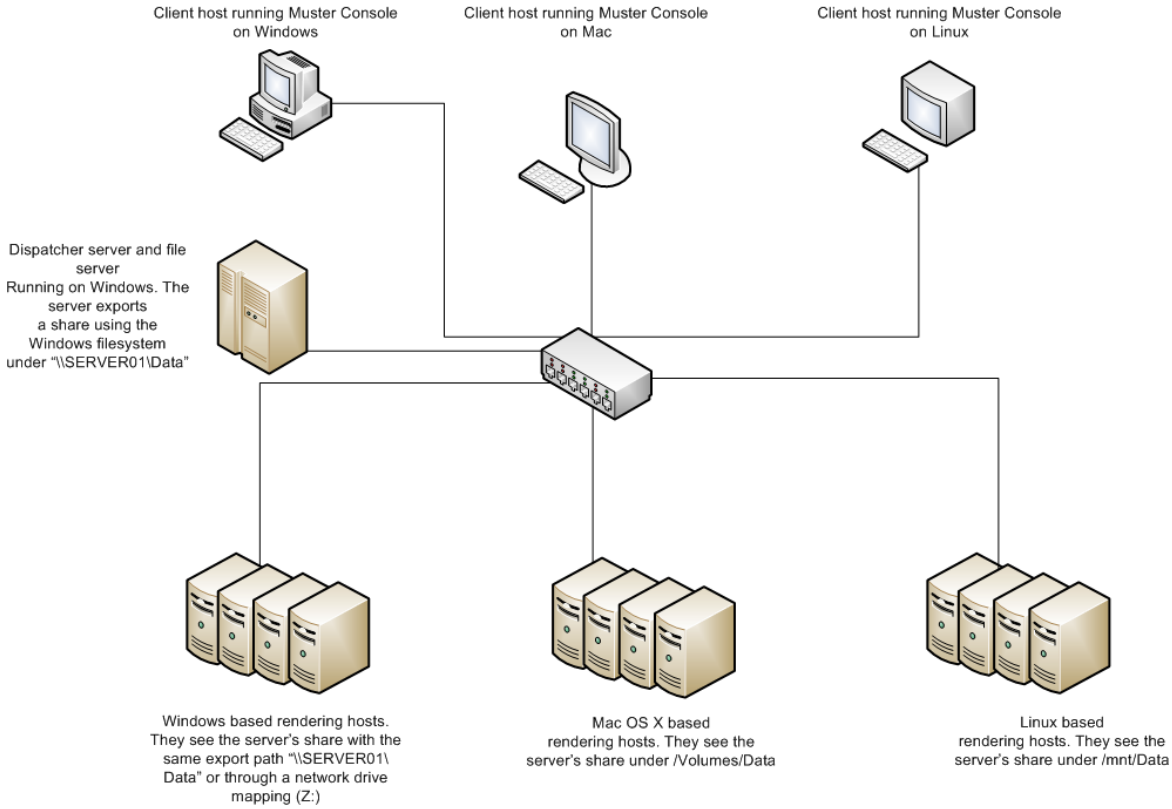
The last step will setup the required file permissions and copy if available, the init.d scripts for automatic services startup. At the time of this writing, we provide init.d scripts for Fedora, Suse and Debian distributions. Some distributions are direct forks of those ones, so there's a chance that our scripts will work in a different distribution.

```
Setting files permissions...
Do you want to install and configure init.d startup scripts for the installed
Muster daemons?[yes]

Copying ./scripts/fedora/muster6d to /etc/init.d/muster6d
Enabling runlevels...

Copying ./scripts/fedora/muster6rcd to /etc/init.d/muster6rcd
Enabling runlevels...

Installation completed. You can start Muster with /usr/local/muster6/dispatcher,
/usr/local/muster6/rc or invoking the muster6d and muster6rcd init.d scripts.

Muster Console GUI can be started with /usr/local/muster6/xConsole and the
Services applet with /usr/local/muster6/xServiceControl
```

Depending on the script availability and your choice, you may end with automatically started services, or you may need to start them from the command line. Just follow the paths and hints that the installer will provide at the end of the installation process.

Cross platform setup scenario

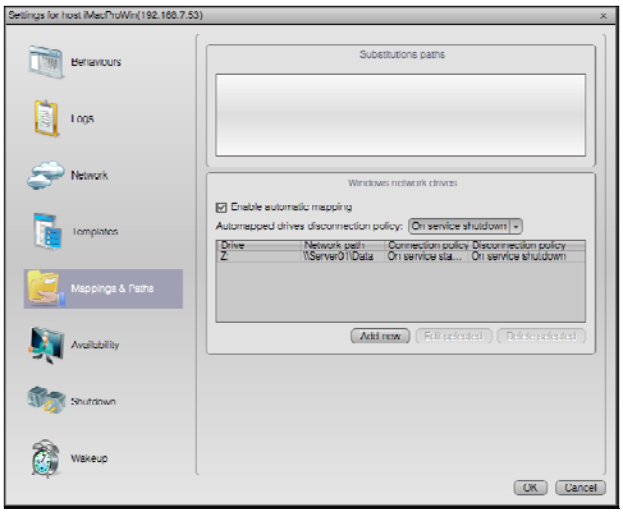
This section explains the steps required to setup a full cross platform environment. The following picture shows the setup of our *fake* render farm, **remember to swap the names of the shares with the ones matching your environment**:



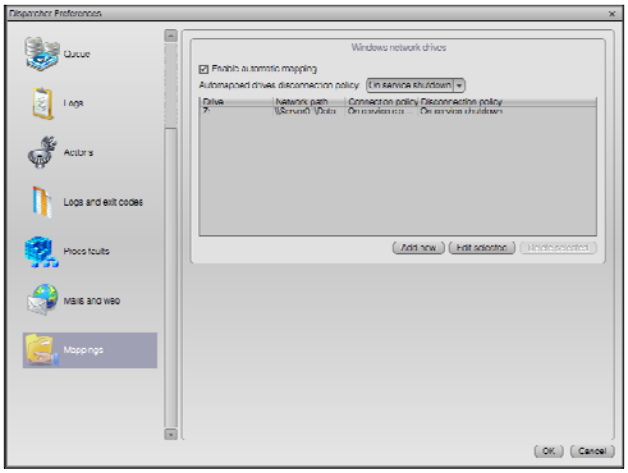
As you can see from the picture, we have a full set of mixed platforms, each one accessing a common shared folder hosted by a Windows server machine. In our example, the same machine also runs the Dispatcher server but the components may be on two distinct hosts.

The first step requires the mounting of the shared file system on each client. Beginning with **Windows**, there's very little to do, considering the Dispatcher is running on Windows too. If we want to support an additional path through a drive mapping, the Z drive must be configured on each client and on the Dispatcher. This is done **exclusively** through the Muster preferences of each module in the **Drive mappings** section. There's no relation between what you map through the interface while you're logged on the machine, and the network drives available to Muster. The Muster service lives in its own space and has the visibility over the shares created by a user.

The following picture shows how the preferences should be configured on the clients and on the dispatcher:



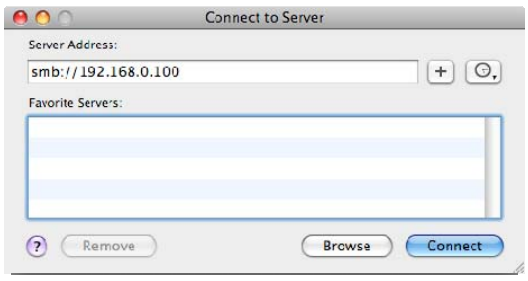
Client configuration



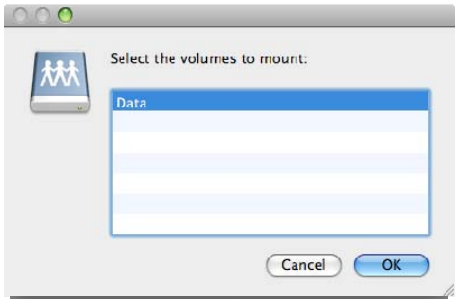
Dispatcher configuration

Remember that after each configuration, you need to restart the services to activate the changes.

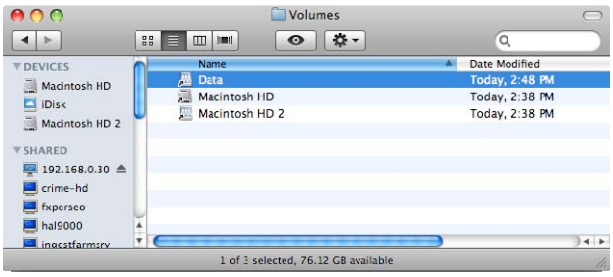
The second step requires the configuration of the Mac platforms. First, you need to mount the shares to make it visible on the Mac. Assuming 192.168.0.100 as the IP address of the network shares, the following pictures shows how to mount it using the Finder **Connect to Server** option:



Connect to the server



Select the share and mount it



Shares is now available in the /Volumes folder

As you can see from the last picture, the **Data** share is now available under /Volumes/Data.

If you're going to run Muster as a Mac service, you need to mount the volume in a different way. Considering Muster will run even with no users logged at all, if you restart your Mac, the share will be unmounted and won't be available until someone remount it. That's why you require a **static mount**, where the Mac will take care of mounting back the volume each time it performs the boot sequence. There're several ways to do that on a Mac platform, the most reliable we found until now is changing the /etc/auto_master file. This is a fast walk-trough:

To start with open up the Terminal application as an administrative user and then use sudo to create a bash shell:

```
sudo bash (enter)
```

You will be prompted to enter your administrator password at this point.

We will now create a file entitled **auto.smb** in the /etc/ directory to hold our server details

```
pico /etc/auto.smb (enter)
```

In this file enter the following line (add more lines for extra servers/shares)

```
$Sharename -fstype=smbfs ://$Username:$Password@$Server/$Share
```

Where:

$Sharename = the name you want to give the mount point
$Username = the user to connect to the server as
$Password = password of the user
$Server = the name of the server (dns/wins entry)
$Share = the name of the share on the server

That means, considering our scenario:

```
Data -fstype=smbfs ://$Username:$Password@192.168.0.100/Data
```

As this file stores the username and password to the server in plain text set the permissions of the file so that only the root user can read it.

```
chmod 600 /etc/auto.smb (enter)
```

Now edit the **/etc/auto_master** file and append the auto.smb record at the end of the file. The auto_master file controls all the automounts for the system, leave everything about this file alone except for the extra line at the end.

```
pico /etc/auto_master (enter)
```

```
#
# Automounter master map
#
+auto_master # Use directory service
/net -hosts -nobrowse,nosuid
/home auto_home -nobrowse
/Network/Servers -fstab
/-  -static
/Volumes auto.smb
```

This will tell the automounter to mount the shares defined in the /etc/auto.smb file under the /Volumes directory. You can force an automout update with: `automount -vc (enter)`

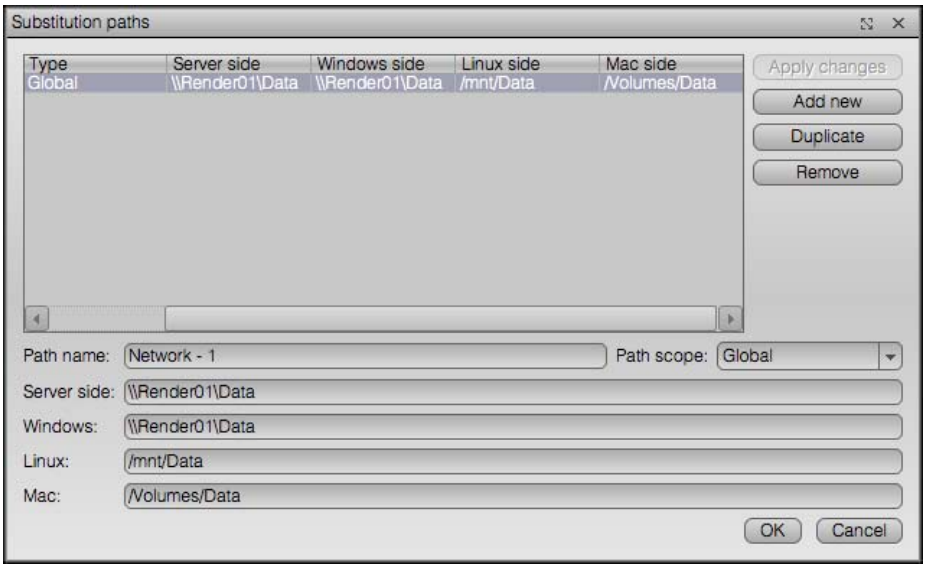
The next step requires a similar work on the linux platform. You can mount a share manually using the “mount –t cifs” command line syntax but we would like to do is have the share mounted statically. You can accomplish this task by editing the /etc/fstab file and adding this line:

```
//192.168.0.100/Data  /mnt/data smbfs
username=$Username,password=$Password 0 0
```

Remember to change the $Username and $Password placeholders with your network credentials.

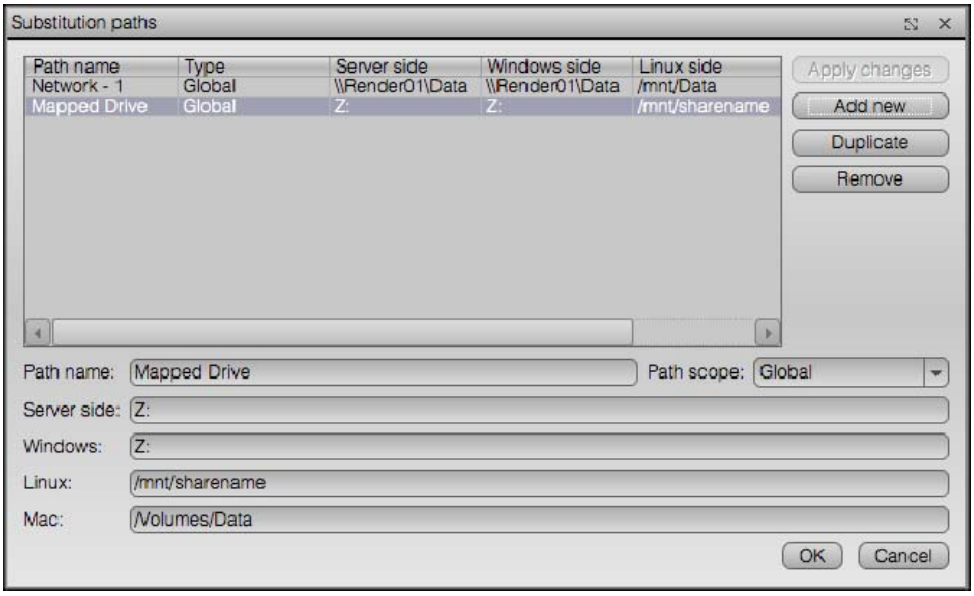
Ok, we almost done the work. We have the shares ready and mounted on each platform. The last step to perform requires the configuration of Muster to tell it how the shares are visible on each platform so it can exchange paths when required. This is done through the **Configure paths** voice

in the Management menu. Once you open the dialog, you have to create a path like the one shown in the next picture:



As you can see, we are telling Muster how paths are seen for each platform. The **Server side** field is a **Dispatcher side** path. Considering we're running the Dispatcher on Windows, the path matches with the Windows one. If we had installed the Dispatcher on a different platform, the field should have been configured according.

We almost done. Remember the Windows drive mapping ? If you want the freedom of using it as well as the direct network paths, you have to add an additional substitution rule:



That's all. Restart the Dispatcher service and start submitting and rendering cross platform!

Preparing a job for Network Rendering

The next step will be preparing a test scene for network rendering. You can use a scene built for this example following the instructions or you can adapt an existing scene taken from your works.

The most important concept to take care when launching networked jobs is **file referencing**.

When you link any external file in your scene, you should check carefully that it's linked in a relative way. A relative link basically means that the path refers to the file **assuming the project path prefix**. Let's make an example with Maya:

Open Maya and create a new project called **MusterTest**. For this example we'll assume that the project has been created inside **C:\MayaProjects**.

Create a test texture with your favourite painting software and save it as **MusterTexture.tga** on the root of your C drive.

Next, open Maya and create a NURBS Sphere, open the Hypershade and create a basic Phong shader. Create a new texture node on the color channel of the shader and select **MusterTexture.tga** on the root of your drive.

As you can see, after selecting the texture, Maya store the path to the file as an absolute path. This happened because the file is not inside the project structure but lives on an external path.

If we try to render this scene using Muster , the render won't be able to load the texture, unless you launch it from the workstation that generated the files and contains the files in their original position. You'll end with an incorrect result on the others nodes because they won't be able to load the files from their root drive.

You can solve the problem copying the textures inside the **sourceimages** folder of the Maya project or create a dedicated folder that must be inside the project structure like **C:\MayaProjects\MusterTest**.

At this time, if we delete the file node and create a new one, when selecting the file, Maya will link it as **sourceimages\MusterTexture.tga. THIS IS A RELATIVE PATH**!

Key concept: Check always that your textures are linked in a relative way. There are several scripts on the web that do exactly this. Some of them allows you to automatically move an out-of-the-project texture inside the project structure**. In Maya, our Connector script does this automatically**.

Even if we link our textures in a relative way, we must be sure that the rendering hosts will be able to access the entire project structure. If we leave it on the C drive of our workstation, it will be impossible for the hosts to access the project. So the next step requires to copy the entire **MusterTest** folder on our file repository, \\MASTER\RENDERFARM for this example.

Because we want to render an animation , animate the rotation of the ball across 10 frames and then save the scene as **test.mb** inside the **SCENES** folder. You are ready to start network rendering with Maya and Muster.

Those concepts applies for any rendering application. What we call a project may be called in a different way but the concept is always the same. Some applications are also able to automatically relocate the external references if they are inside the same or a child path of the job file.

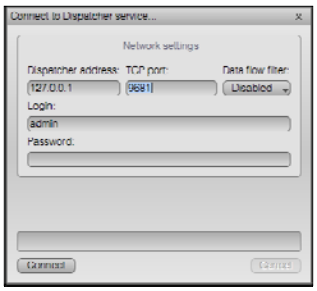
We strongly suggest to read the batch rendering documentation of the software you're going to use with Muster for further information.

Muster Console walk trough

Muster Console is the graphical interface to the Dispatcher Service. Each operation on the render farm and on the queue is performed through it.

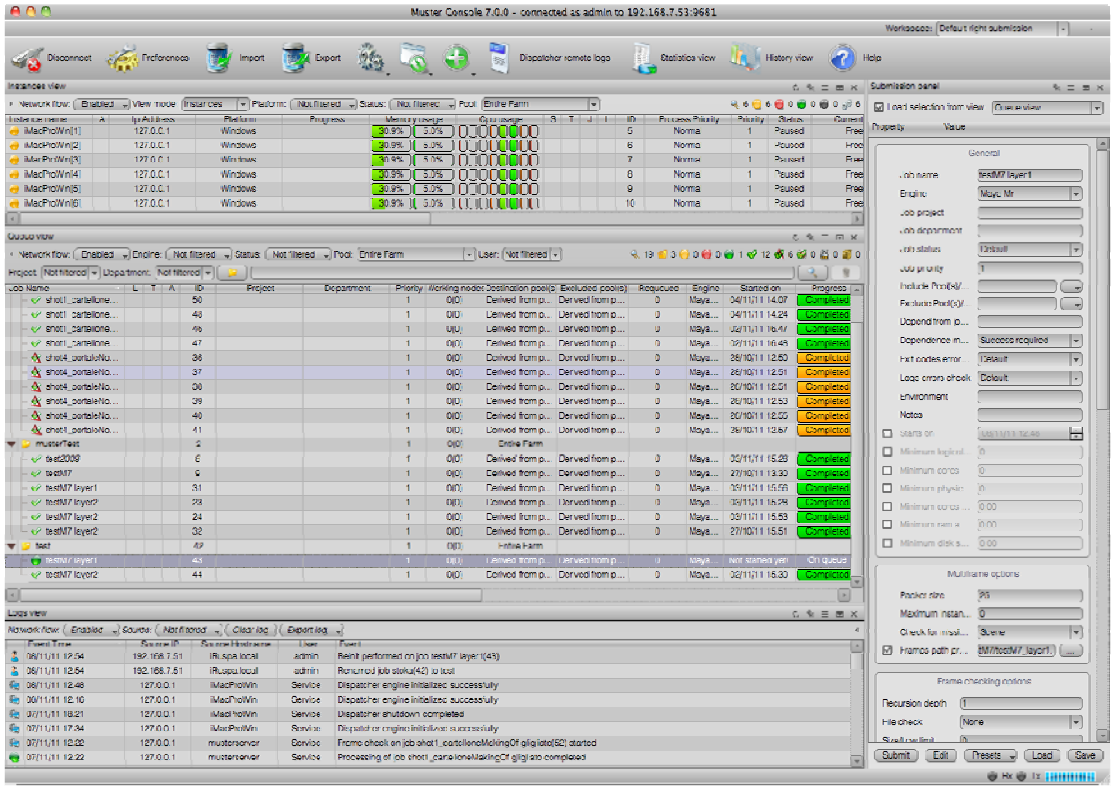
Launch the Console. This can be done from the Start menu in Windows, from the Application folder in OS X and from a command line in Linux. You should see something like this:

The first thing you need to do is to log in on the dispatcher service, so click on the first icon on the left side of the toolbar. A connection window will appear:



Insert your dispatcher service host name or IP address, put **admin** as username and leave the password field blank, then click on Connect.

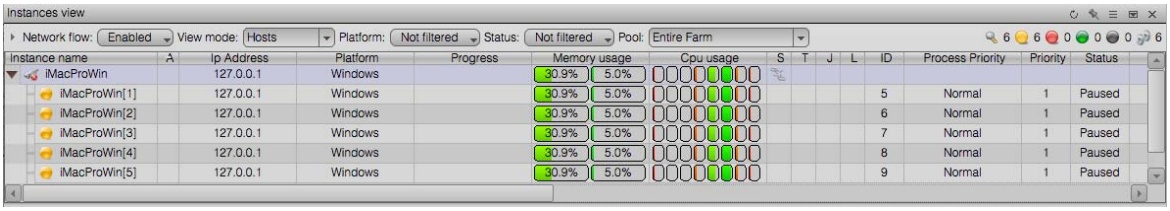
If the connection is successfully, you'll get the Console interface:



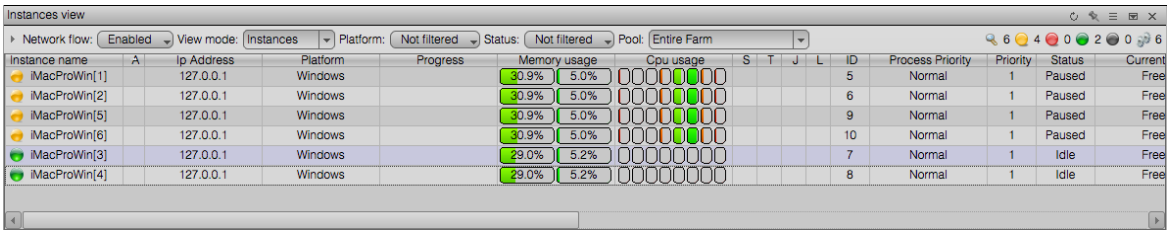
The Muster Console interface is made by four different view panels:

The **Instances/hosts** view: This is the upper view in the screenshot above. This is basically a list of your available and unavailable rendering instances and can be configured in **Instance** or **Host** view mode. Consider an host like a physical machine, while an instance is an **entity** inside Muster. One host is able to spawn multiple instances. In this way, you can have simultaneous batch renders running on the same host and use multi core and multi processors machines even the batch render doesn't support it. In Instance view mode, each instance has its own entry in the list, while in host view mode, instances are grouped under their host node. Operations that are pertinent to a single instance can be done in Instance view mode, while operations that are pertinent to the host , like starting and stopping services, are permitted in host view mode only.

This is an image of a typical view in host mode:



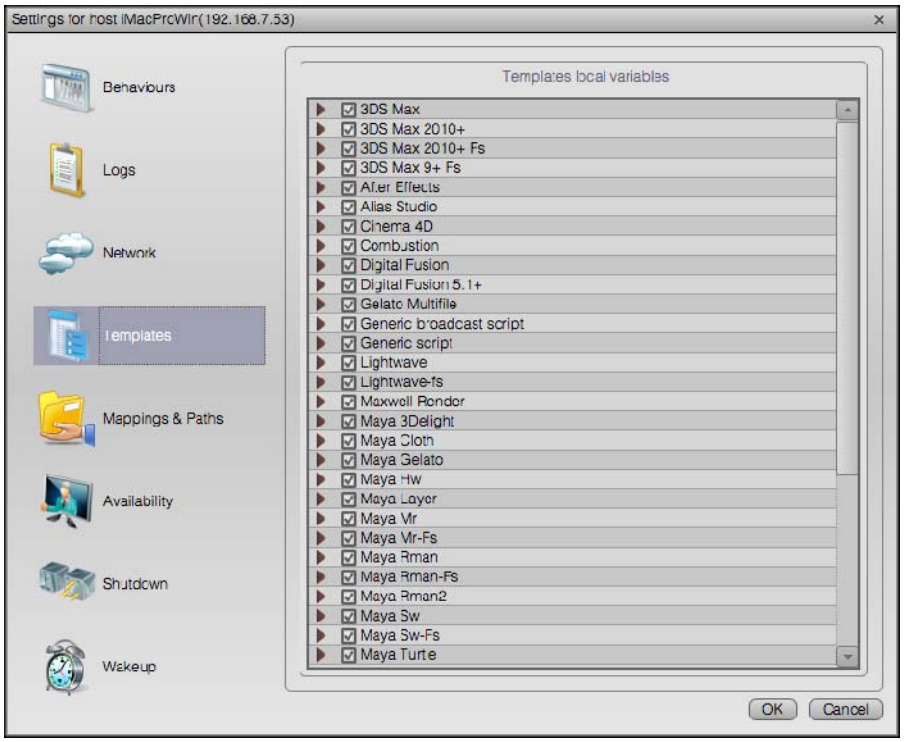
While this is an image of a typical view in instance mode:



The colored dot at the left of each instance shows the actual status of the machine. When the machine is idle and ready to render, it is green. When it's idle but paused, it's yellow. When it's rendering, it's red. After your first installation, you should have at least one instance with a green dot. If it's yellow, you may have configured the client to start paused. You'll change its status later.

The instances/hosts views can be filtered using the combo controls at the top of the view itself. You can also customize the columns by right-clicking on the view headers. **If you're working on a custom workspace (**see the workspaces section) **, each change you make in the view, will be stored persistently and left untouched at the next Console startup!** This allows you to build your very customized workspace with multiple views filtered in different ways!

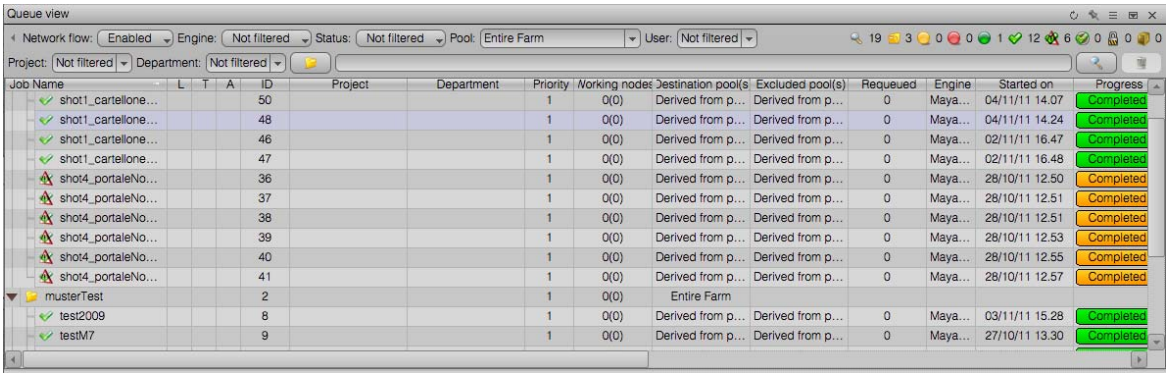
Before going further, we'll check that the hosts we are going to use have the correct paths configured. Just switch to **host view** mode, right click on a node and select **Configuration->Configure.** Once you get the configuration window, move to the **Engines** section, locate the Maya sw template we are going to use for this test, and verify that it's correctly pointing to the **Render.exe** file required to launch Maya batch renders. Also check that it points to the correct **Maya** version, the one that you used to produce the scene file and that's enabled with a checkmark near the engine name. Click on **Ok** and your configuration will be stored persistently on the host.



As said before, this configuration will apply to any instance belonging to the host and does not require a restart of the service. Your changes will be available immediately.

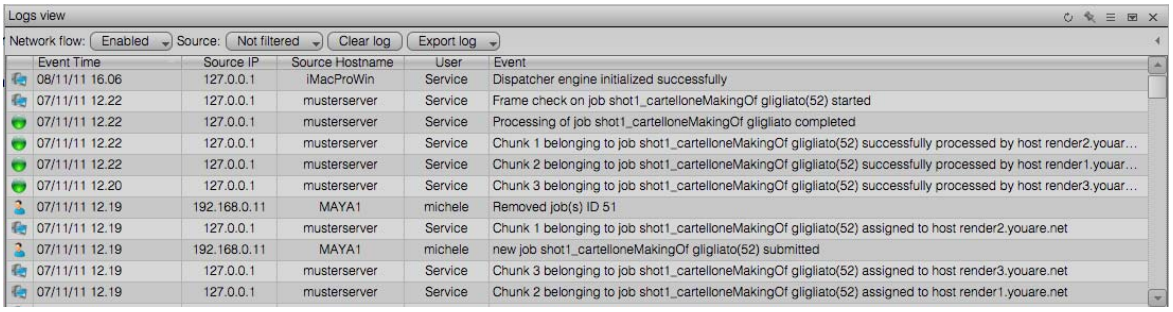
The next view, the one shown in the central pane contains the dispatcher job queue and it's called **Queue** view. Jobs can be organized inside folders so it's possible to have collapsed or expanded nodes. The icon on the left of the node shows the status of the node itself and in case of a folder, the overall status of the childs.

This is a typical queue view:



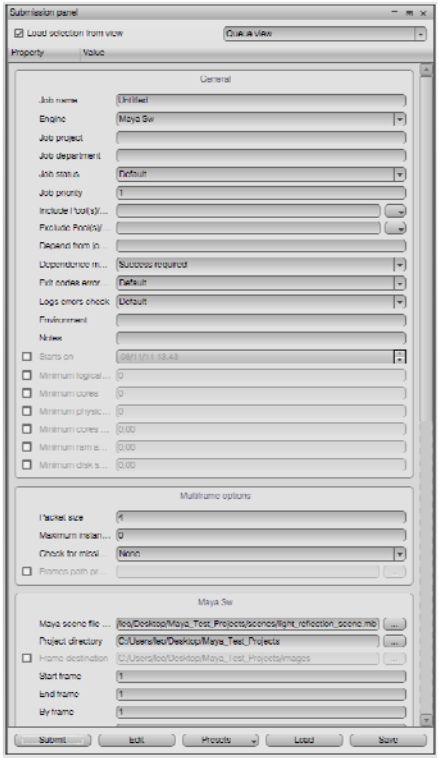
After your first installation, this view will be empty. You will need to submit some jobs to populate it.

The bottom pane contains the dispatcher internal log. It's a resume of the most important events happened on the Dispatcher service. This is a typical log view:



You should always take a look at the log during the rendering process. Every error reported from render clients or directly from the dispatcher will be logged here and is your primary error detection tool.

The last pane, the one on the right, is the **submission view.** From this window you can submit new jobs, inspect parameters related to jobs inside the queue, load and save presets. The Submission view is a dynamic property sheet. It will change contents depending on the currently selected render engine:



You can configure the submission panel to automatically load the jobs selected in a particular queue view. Just enable the **Load selection from view** option at the top of the submission dialog.

To explain its functionalities, we will submit our first job!

Before doing that, we have to check that the internal Dispatcher engine is running.

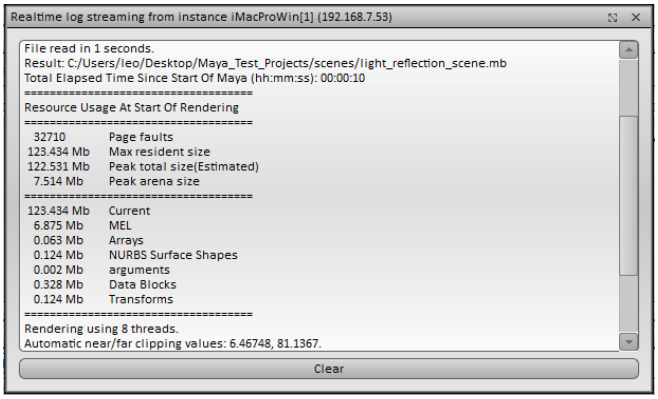
The Dispatcher engine is the abstract object made by its internal functionalities. By stopping the Dispatcher engine, you stop any kind of future activity on the Dispatcher. It won't stop any

render in progress but will prevent the submission of new ones. Stopping the Dispatcher may be useful in those situations where you need to configure something on the Dispatcher itself and you don't want any activity until you complete the configuration.

You can immediately check the activity of the Dispatcher by looking at the lower right corner of the Console. If you see a blue moving bar, the Dispatcher is active. If not, just start it by pressing the F10 key, or selecting the **Change Engine Status** menu entry in the **Management** menu.

Now check that our instances are available for rendering. If some of them have a yellow ball, it means they are paused. Just select them, right click on one and select **Resume**. They should change to the idle status and show a green ball.

As a last step, we want to exactly check what's outputted from the instances during their render, so right-click on the instance you're going to use, and select **Realtime Log Streaming -> Enable and Open view.** By selecting this menu item, Muster Console will open an output window that will dump the output from the processes spawned on the remote host:



We are ready to select our rendering engine from the submission view. In the general section, locate the **Engine** combo box, and select "Maya sw" as the render engine. This will tell Muster to render using the Maya software rendering.

Next, select the job file by clicking the button on the right of the Maya scene filename field, in the Maya Sw section of the submission panel (**notice that this section has been dynamically built if you had a different render engine selected)**.

Pick up the file from the network, you should have something like \\MASTER\RENDERFARM\MUSTERTEST\SCENES\TEST.MB inside the field. Muster will detect automatically the project path and will set it to \\MASTER\RENDERFARM\MUSTERTEST.

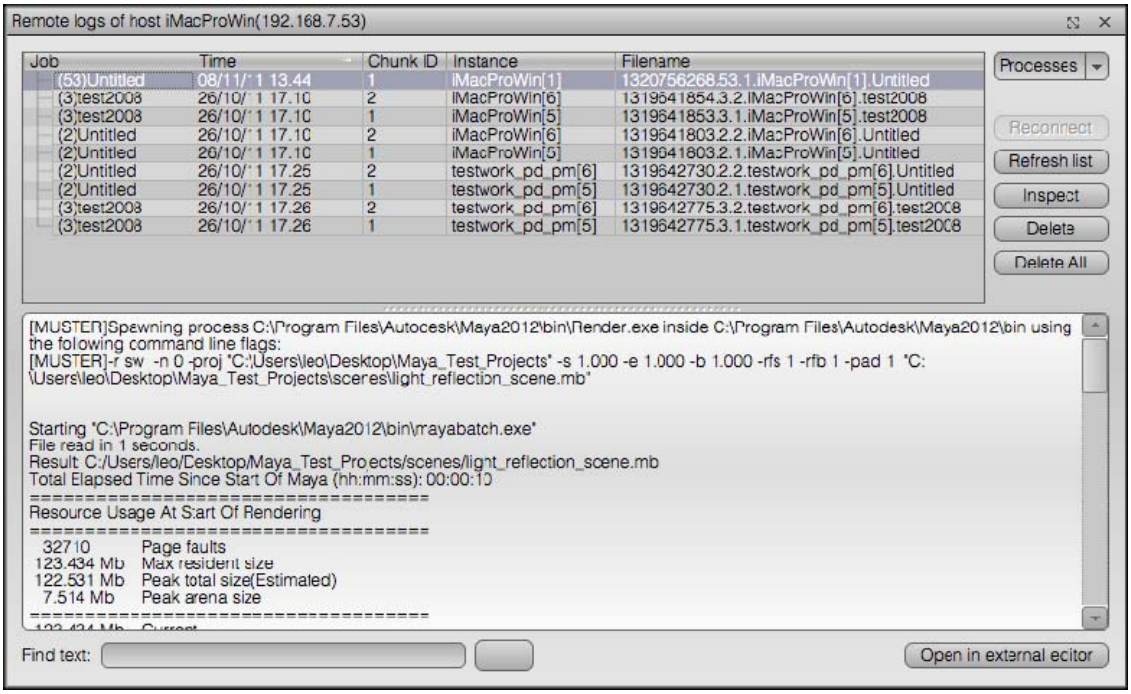
Set the starting frame to 1 and the ending frame to 10. Because we want to send the job to an high number of instances, set the packet size to 1.

The packet size value controls how many frames are assigned to a single instance for each render session (**a chunk**). Under production, a value of 4 is often a good compromise.

Click on submit and you should immediately see the job appear in the queue. If it has a yellow ball, just right click on it and select **Resume**.

Congratulations, you've sent your first render job. In a few minutes, you should get all your instances back to idle and the frames successfully rendered inside the images folder of your project.

During this example, you watched the output from the batch renders using the real time log feature. But what about checking the logs of completed packets ? No problem, just open the **Workstation Logs** menu item right clicking on your host and you'll get the Log's inspector:



The logs have the solution to your problem. Muster is not able to always understand what's going on during a batch rendering process. The output from the batch render always contains enough hints to understand why a particular job is failing!

You can start playing with the various Muster parameters or move to the reference section of each module to gain a deeper knowledge of the software.

If something went wrong, we suggest you to check carefully each section of this chapter and verify your steps. If you still can't get a valid result, check the Troubleshooting section at the end of this manual.

Muster Console Reference

Interface components

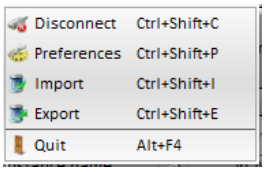
The following section will give a brief over the interface components provided by Muster Console, the complete real time graphical front end to the Muster Dispatcher Service.

The Muster Console window has a menu , a toolbar and five view types:

- The instances/host views
- The job queue views
- The log views
- The submission views

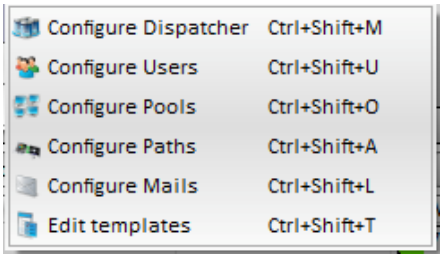
The Muster menus offer the same functionalities of the icons in the toolbar but divided in logical groups:

The file menu has the following options:



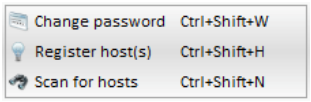
- **Connect/Disconnect –** Shows the connection dialog or terminates an active connection
- **Preferences –** Opens the Console preferences window
- **Import** - Imports a queue snapshot into the Dispatcher queue
- **Export** – Exports a snapshot of the current queue
- **Quit** – Exits from the Console application

The management menu has the following options:



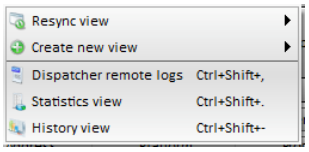
- **Configure Dispatcher** – Opens the Dispatcher configuration dialog that lets you change the Dispatcher server options
- **Configure Users** – Opens the Users configuration dialog to manage the Dispatcher server internal users database
- **Configure Pools -** Opens the Pools configuration dialog to manage the Pools database.
- **Configure Paths** – Opens the Substitution paths configuration dialog that lets you change the paths used to apply the substitutions between different hosts and platforms.
- **Configure Mails** – Opens the mail configuration dialog that lets you define distribution lists to be used with the Dispatcher mailing notification system
- **Edit templates** – Opens the templates editing dialog that lets you change and propagate the templates on the fly
- **Change Engine status** – Starts and stops the Dispatcher internal engine
- **Soft-restart Dispatcher** – Perform a soft restart of the Dispatcher service forcing a reload of the configuration and a reconnection with the slaves

The action menu has the following options:



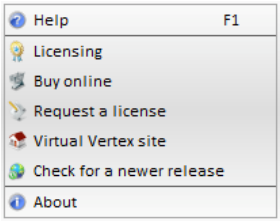
- **Change password** – Shows a dialog that lets you change the password for the currently logged user
- **Register host(s)** – Shows a dialog that lets you add a new host to the hosts/instances views
- **Scan for hosts** – Tells the Dispatcher to start a scan and find unregistered hosts in the current hosts/instances views

The view menu has the following options:



- **Resync view** – Reloads the selected views
- **Create new view** – Creates a new view and associate it to the current workspace
- **Dispatcher remote logs** – Opens a dialog that let you browse the logs on the Dispatcher
- **Statistics view** – Opens the statistics view dialog that lets you browse the Dispatcher statistics in real time and eventually export them
- **History view** – Opens the history view dialog that lets you browse the history of the Dispatcher activity and eventually, export and import the data.

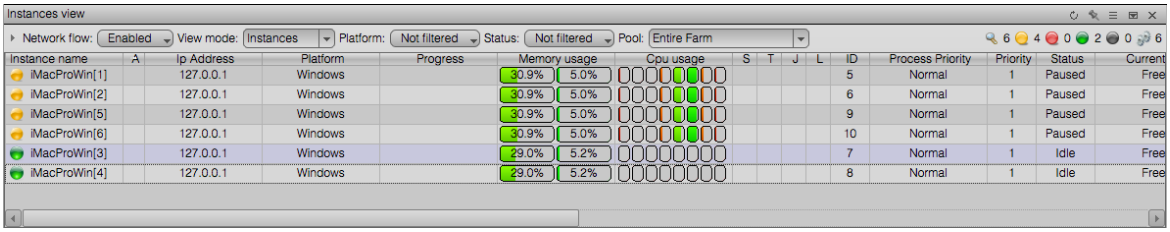
The help menu has the following options:



- **Help** – Launches Acrobat Reader and opens the user's manual
- **Licensing** – Opens a dialog that lets you change the current license
- **Buy online** – Connects to the Virtual Vertex store
- **Request a license** – Connects to the Virtual Vertex license requesting page
- **Virtual Vertex site** – Connects to the Virtual Vertex home page
- **Check for a newer release** – Check if a newer release is available online
- **About** - Shows the about screen

The instances/hosts view

The next figure shows a typical hosts/instances view in instances view mode. The view shows the connected and unconnected slaves instances, and their relative status will be immediately visible during their activity.



Each column of this view shows a specific property of the connected instance. The columns headers can also be used to sort the information. Just click on one of them and the view will be resorted. The default sorting is priority based.

The meaning of each column follows:

- **Instance name:** This is the name assigned to the machine. It can be the Netbios name on Windows or the name returned by gethostname() on Unix machines. When multiple instances are started on the client, an instance number is appended to the standard name. More information on instances are contained in the render client reference section. In addition to the host name, the icon on the left shows the current status of the machine. It can assume the following states in instance view mode:

   The instance is idle and waiting for jobs

   The instance is paused

   The instance is busy processing a job

   The instance is disconnected

   The instance is connected but unavailable by configuration rules

   The instance is connected but reserved by a logged user

  or the following states in hosts view mode on the host nodes:

   Console is actually directly connected to the host

   Console is not connected to the host

   Console is attempting a connection to the host

- **IP address:** This is the IP address of each machine. Multiple instances of the same client will share the column value
- **Platform:** Operative system running on the client
- **Memory usage:** This is the actual usage of RAM and swap file on the host
- **Cpu usage:** This is the actual usage of processors on the host
- **Procs:** This is the number of virtual processing units on the host. Hyper threading machines report multiple virtual units
- **S:** This is used in host view mode only and reports the status of the Render client service on the host. It can assume the following states:

   Service is up and running

   The Service status has not been queried yet

   Service is stopped

   The service is starting

   You do not have the rights to query the remote service

   Service is either uninstalled or an unknown error happened querying the service

- **T and J:** If you see a red cross  in this column, it means that one or more templates or jobs have been add to the client exclusion list. The exclusion list prevents further assignment of the job to the client to avoid infinite loops on faulty jobs
- **L:** Shows the status of the real time log. When it's enabled, you'll see a log icon  in this column
- **ID:** This is the internal identification number assigned by the Dispatcher to each host.
- **Process priority:** This is the scheduling priority for each process launched on the host.
- **Priority:** This is a priority number assigned to each instance. When idle, instances with the higher priority will be selected first in the job assignment logic.
- **Status:** Shows a textual status of the instance
- **Current job:** If busy, this field shows which job is currently assigned to the instance
- **Current chunk:** If busy with a multi frame job, this column shows the current job's chunk
- **Notes:** Shows notes assigned to the instance

- **Mac Address:** Shows the MAC address of the NIC of the node. If you have multiple NICs installed, you'll see multiple addresses separated by a pipe (|)
- **Command line:** If busy, shows the command line sent to start the current process

If you want to filter the view contents, you can use the options on the filtering bar:

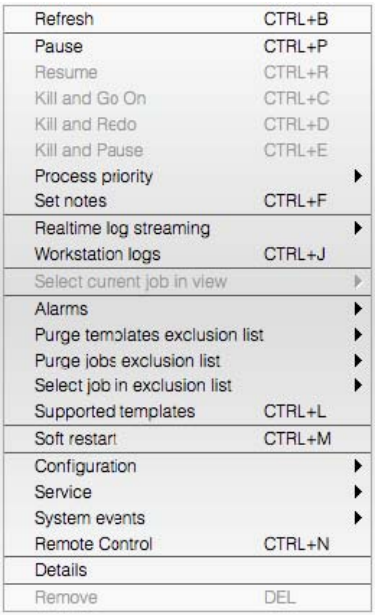
- **Network flow:** Enable or disable data flow to the view. If you disable the network flow, each host view will be disabled and you'll reduce the amount of network traffic between the Dispatcher and the Console. You should always disable a certain view network flow if you don't need the windows to be constantly updated
- **View mode:** Changes between hosts and instances view mode
- **Platform:** Filters the contents by looking the instance's platform
- **Status:** Filters the contents with the status value
- **Pool:** Filters the contents showing only instances belonging to a certain pool

Settings of the filtering bar will be stored persistently if you're working on a custom workspace. If you're working with the default, the settings will be reset on the next session. Check the workspaces section to learn more

Remember to disable the network flow on views you're not interested in. This will reduce the amount of traffic between Console and the Dispatcher!

Managing the hosts/instances

By right-clicking on a host or instance in the instances view, you get a popup menu that allows you to perform different actions. The same applies to groups if you select multiple items:



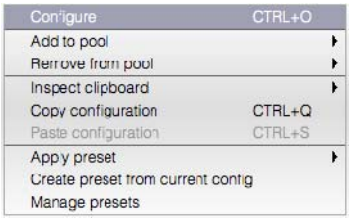
This is a brief explanation of each option:

- **Refresh:** Forces a refresh of the instance's status
- **Pause:** Pauses the instance
- **Resume:** Resumes a paused instance
- **Kill and go on:** Aborts a working instance, put the chunk back in the queue and move to the next or the same chunk, depending on clients availability and priority position
- **Kill and redo**: Aborts a working instance and restarts the chunk from the beginning
- **Kill and pause:** Aborts a working instance and pauses it
- **Process priority menu:** Sets the system process scheduling priority for the running process
- **Set notes:** Sets custom notes on the instance
- **Real time log streaming:** Enables or disables real time log streaming from the instance.
- **Workstation logs:** Accesses the workstation logs inspector. Through the inspector you can access logs produced by the instance, read and remove them

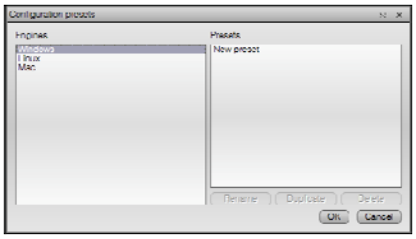
- **Alarms:** If you have an alarm running, you can reset the status from this menu
- **Purge templates exclusion list:** If you have some templates into the instance's exclusion list, you can see and remove them from the associated menu
- **Purge jobs exclusion list:** If you have some jobs into the instance's exclusion list, you can see and remove them from the associated menu
- **Select job in exclusion list:** Let you directly select a job fro the exclusion list
- **Supported templates:** Opens a dialog that shows the instance's supported templates (engines)
- **Soft restart:** Performs a soft restart reloading the configuration, disconnecting and reconnecting it again. Unlike **Reinit,** this is done on any instance sharing the same node
- **Configuration menu:** Access the configuration option for the node
- **Service:** Accesses services query options
- **System events:** Sends system events like shutdown or restart to the node
- **Remote control:** Attempts to take control of the node using Remote Desktop, ssh or a valid tool. This may be customized in the Console preferences dialog
- **Details:** Give a full resume of the host details like the OS, the system capabilities (RAM and CPU) and the free space on the hard drives
- **Remove:** Removes the instance from the list. Works only on disconnected instances

Before attempting to configure automatic wakeup for the hosts, try to shutdown and wakeup an host manually using the options in the System Events menu. Also check the documentation of your motherboard and its Bios to understand if your system actually supports wake up on lan (**Magic packet**)

If you select the Configure menu, you'll get the following additional popup:

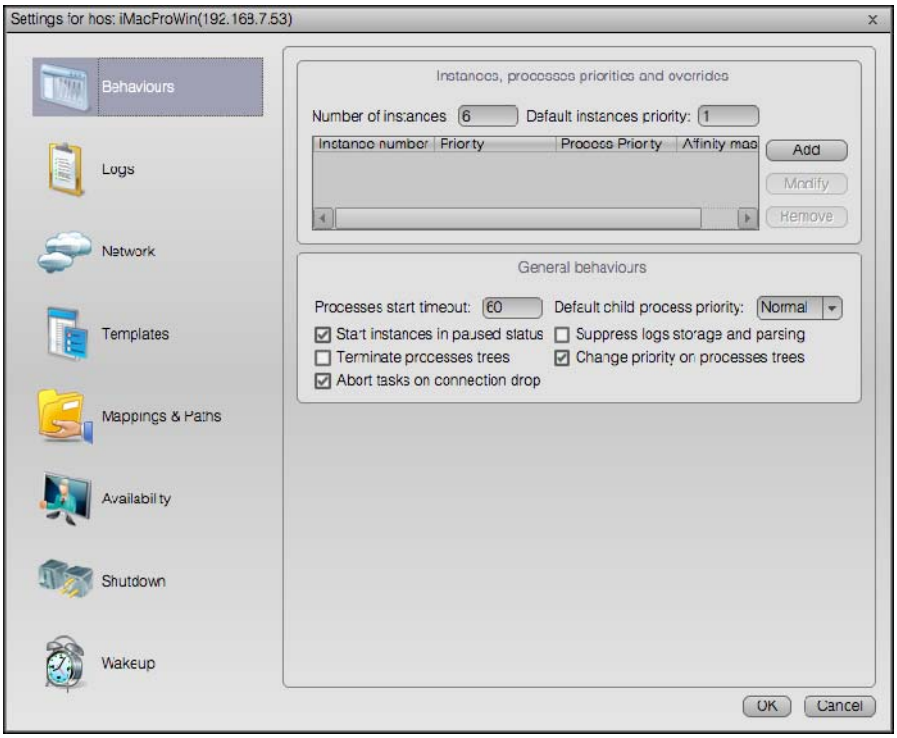


The menu let you add or remove an host on the fly from a specific pool as well as inspecting configurations, copy configurations and create configuration presets. You can create several presets for each platform, and manage them using the configuration presets dialog:



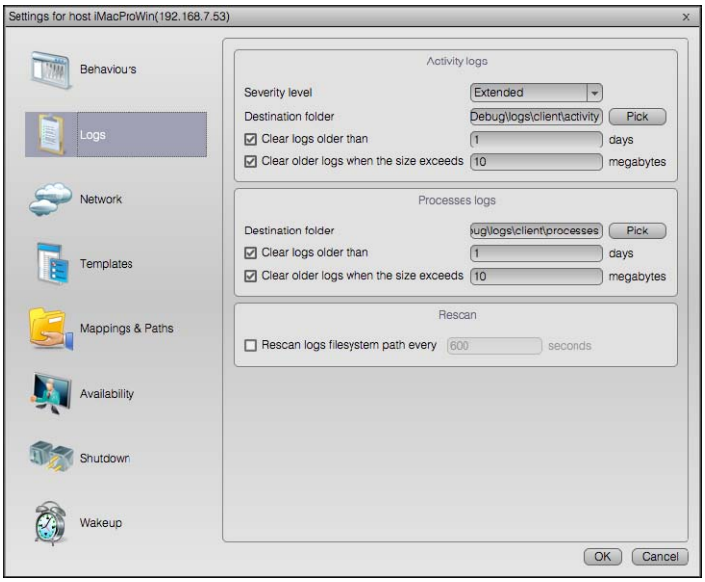
Configuring the hosts

Accessing the configuration menu clicking on **Configuration -> Configure**, the Console attempts a direct connection to the host to configure its behaviours. Once a successfully connection is made, you'll be prompted with the configuration dialog as follows:



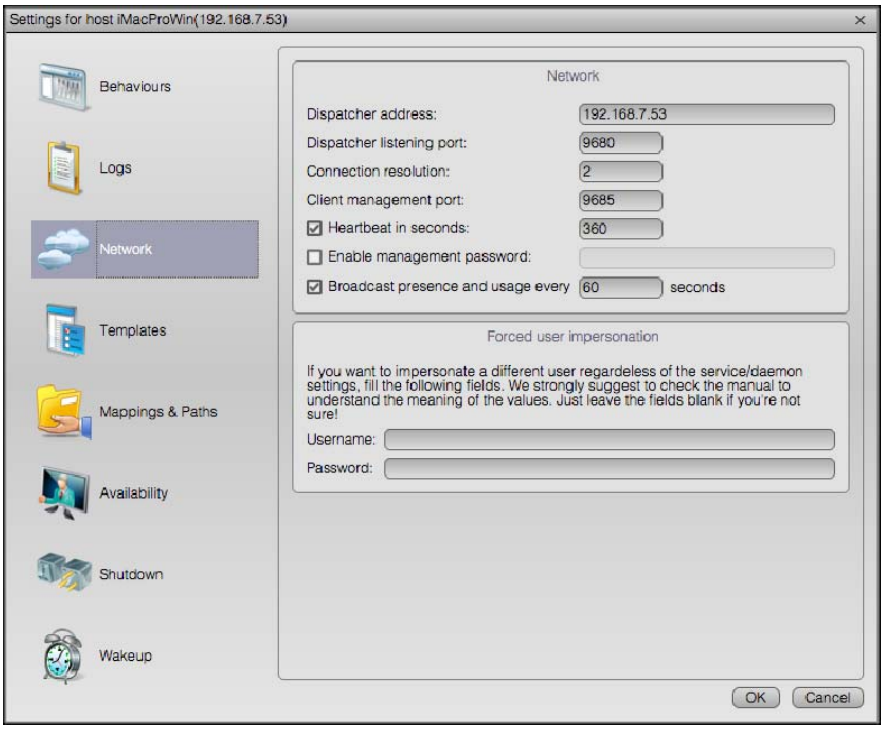
- **Number of instances:** Number of instances to spawn on service boot
- **Default instances priority:** The priority of the instances by default. Nodes with an higher priority will be allocated first. You can override the priority on an instance basis using panel below
- **Overrides window**: You can create several overrides on an instance basis as well as setting a specific process affinity mask for each instance
- **Default child process priority:** This defines the system default scheduling policy for each process spawned by the client. Values map to OS specific values
- **Process start timeout:** This defines a timeout value while Muster tries to catch the rendering process PID. If the timeout expires, the process is terminated and the chunk requeued and/or reported as failed
- **Start instances in paused status:** This starts the client in a paused status. Keep in mind that you should keep this feature disabled if you're going for a fault tolerance, automated render farm. Having this option enabled, will prevent client activity after a forced reboot until an administrator resumes it

- **Terminate processes tree:** When the client tries to terminate a running process, the kill command is sent to the entire process tree (if available)
- **Change priority on processes tree:** When the client tries to change the process priority, the command is sent to the entire process tree (if available)
- **Abort tasks on connection drop:** Tells Muster to abort any running process if the socket TCP/IP connection between the client and the Dispatcher drops
- **Suppress logs storage and parsing:** If you want to skip the error processing and the log parsing, enable this function. This should be used for debug purposes only



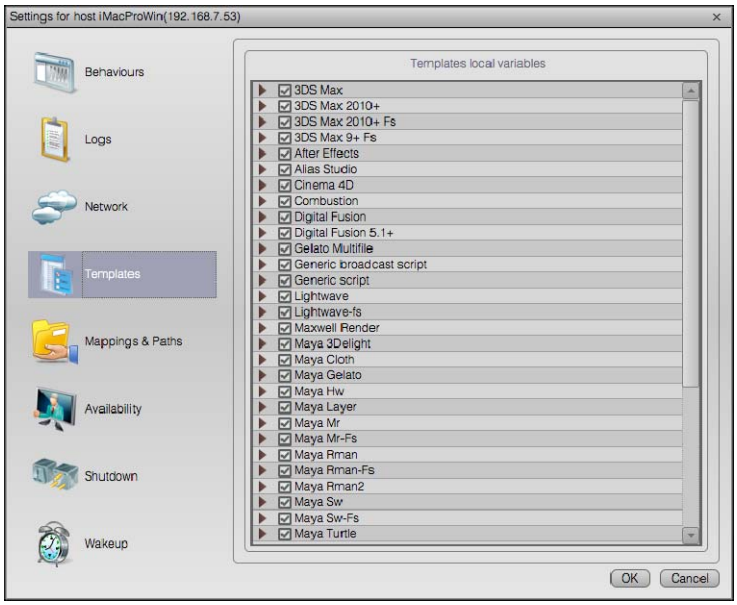
The logs section let you configure the storage path for the activity and processes logs. You can also configure the following parameters:

- **Severity level:** This applies to the activity logs only and set the logs severity filter
- **Destination folder:** Sets the local destination folder to be monitored by the logs engine
- **Clear logs older than:** Specifies if you want to automatically delete logs files older than a certain number of days
- **Clear older logs when the size exceeds:** Specifies if you want to automatically delete logs files when their size exceeds a certain amount of Megabytes
- **Rescan logs file system path every:** If you store the logs on a common location and you remove or change the files outside of Muster, you can tell each client to rescan the folder to have them in sync with the Workstation log's view

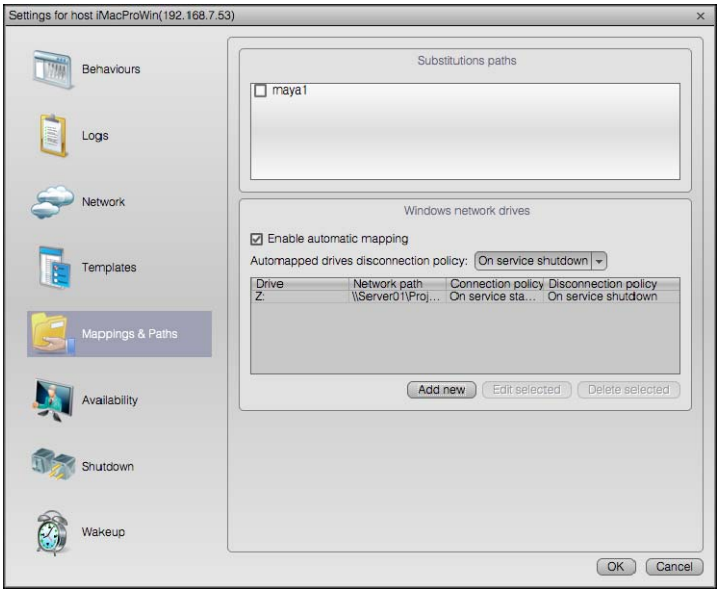


- **Dispatcher address:** The IP used to connect to the Dispatcher service
- **Dispatcher listening port:** The TCP/IP port where the Dispatcher is listening for incoming Render client connections
- **Connection resolution:** Attempts to reconnect after a disconnection when the amount of seconds is passed
- **Client management port:** The TCP/IP port where the Render client are listening for management connections incoming from Consoles
- **Heartbeat in seconds:** Send a pulse to the Dispatcher to let it know the client is alive. Depending on this setting, and the one in the Dispatcher global preferences, a client may be disconnected and flagged as offline if there's no activity for a certain amount of time
- **Enable management password:** If you want to block incoming management connection on a client to prevent unauthorized changes, just put a password in this field and future connections will ask it to the users
- **Broadcast presence:** If you want the client to be auto discoverable by the Dispatcher on the network, flag this option. After you successfully configure your render farm, you should disable this option to avoid packet storming on your network

You can also specify a different username and password pair to be used when launching external processes. Unless you've a particular reason , you should leave those fields blank and relay on the configuration of the Services on Windows and the startup scripts on Unix.



The Engines section of the client configuration dialog let you configure variables required by each batch render template. The values are template specific but there's always a variable pointing to the batch render executable. You can change this value to use different versions of your software and you can check/uncheck the checkbox near the template name to enable or disable the support of that particular engine.

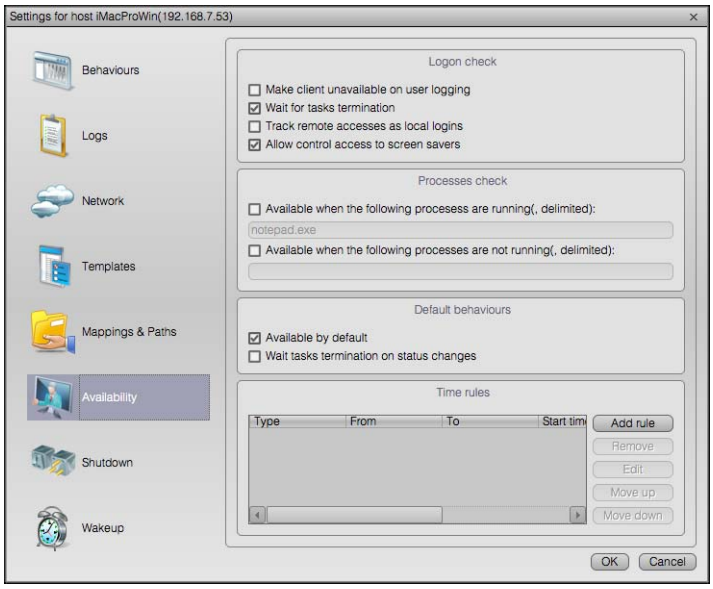


This Window let you specify one or more substitution paths to be used when dealing with the client. If you're configuring a Windows client, you can setup **static drive mappings** too. Having a drive mapped in your interface does not propagate the setting to Muster. As specified in the

beginning, Muster lives in its own user address space. That means you've to tell it the drives to map.

By changing the way the client maps the drives let you keep under control the amount of connections to your file server limiting the amount of client licenses required. You can also activate the automatic drive mapping for Windows that embeds and automap a network share within each job.

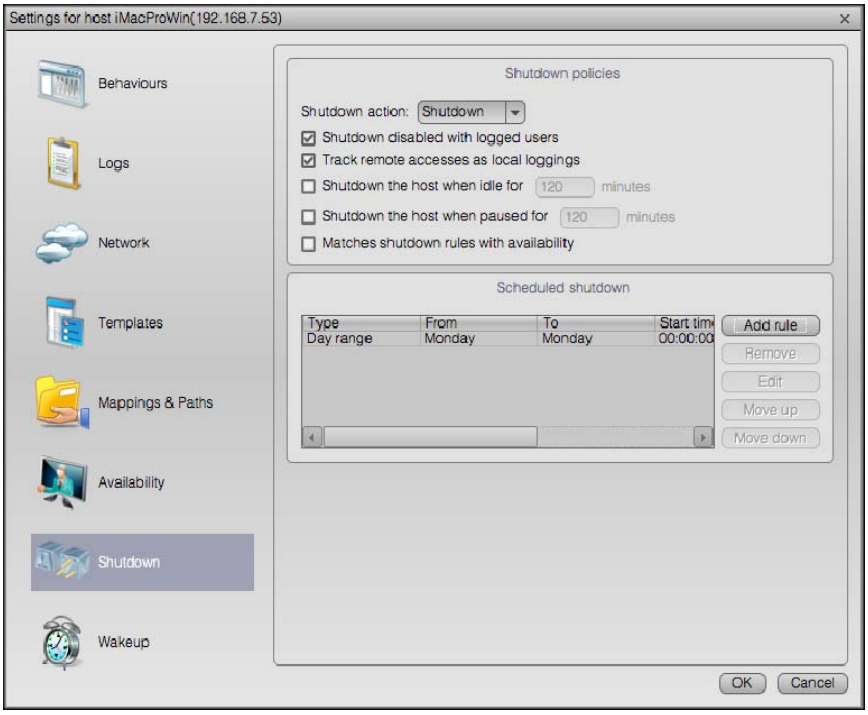
Drive mappings information is embedded in the jobs only when submitting them from a Windows workstation and picking up the file from the drive map itself. There's no way to embed a drive mapping information when submitting the job from Linux or MAC OS X. In a mixed OS environment, you should relay on static drive mappings configuration and disable this function.



The availability rules define when a particular client is available for rendering. You can choose its default availability by checking or unchecking the **Available by default,** tell the client if it has to abort the running process or wait its termination when its availability change by checking or unchecking the **Wait tasks termination on status change** and configure specific time lapses by clicking the **Add rule** button.

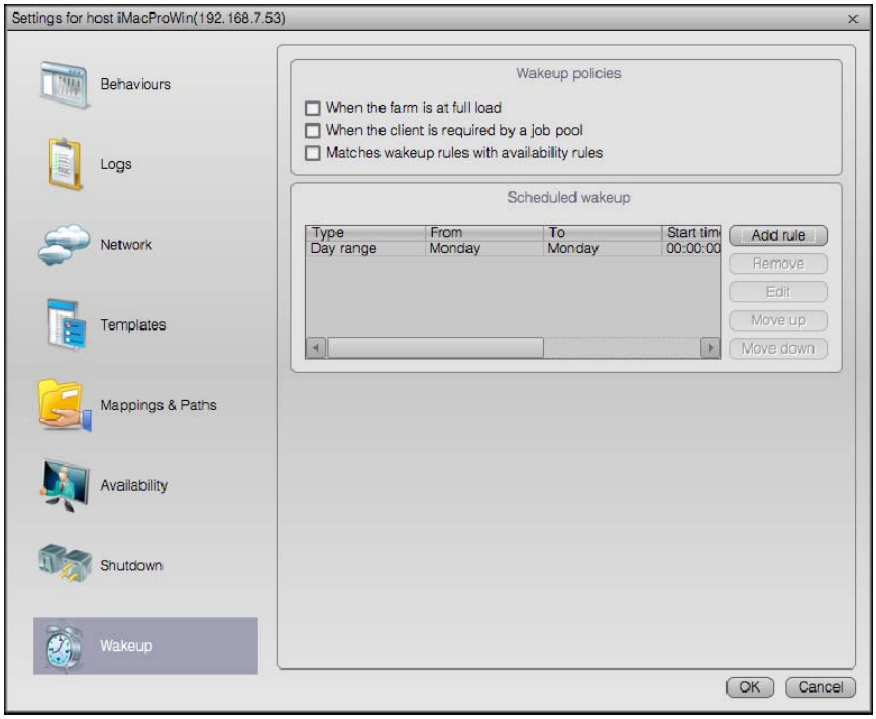
You can also tell the client to be available or not available depending on the presence or the absence of a particular process. This is very useful to let Muster co exists within other software that requires full control of the host.

Keep in mind that rules are always evaluated with other configurations like being available while a user is logged or during the screen saver activity. To make a client eligible for being available, the entire set of rules must be satisfied.



The shutdown section let you configure rules to automatically shutdown an host. Using the wake up feature, you can setup your render farm to shutdown on idle timing and wake up on demand when the full power is required. This is a great feature to reduce costs.

- **Shutdown action:** You can choose if the shutdown action is effectively a full host shutdown, a sleep action, or a restart. This is particular useful on Mac OS X, where there's no way to wake up a powered off Mac. You can put it into sleep instead. During this phase, the wakeup function through the LAN will work
- **Shutdown disabled with logged users:** This tells Muster to avoid a shutdown process if someone is logged on the workstation
- **Track remote accesses as local loggings:** If someone is logged though a remote connection, Muster considers them as local loggings and prevents a shutdown if enabled in the previous option
- **Shutdown the host when idle for:** Tells Muster to shutdown the host after a certain amount of minutes of idle status
- **Shutdown the host when paused for:** Tells Muster to shutdown the host after a certain amount of minutes of paused status
- **Scheduled shutdown:** Defines custom rules to shutdown the host
- **Matches shutdown rules with availability rules:** Apply the action only if the availability rules are also valid

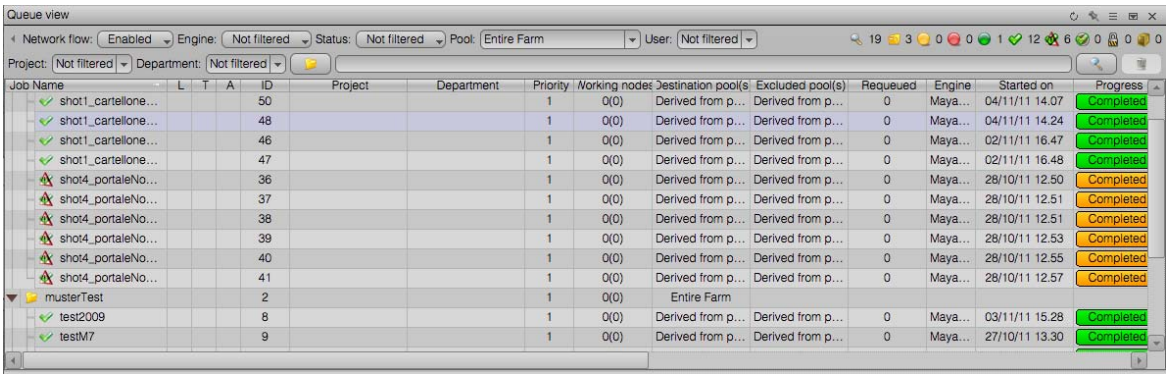


The wake up dialog let you configure specific rules to wake up an host through the **Magic Packet (formerly Wake up on Lan).** Apart from specific timings you can check the following options:

- **When the farm is at full load:** Wakeup the client when there are no idle hosts on the farm
- **When the client is required by a job pool:** Wakeup the client when it is required by the job's pool
- **Matches wakeup rules with availability rules:** Apply the action only if the availability rules are also valid

The queue view

The next figure shows the jobs queue view. As you can see, jobs can be arranged in a hierarchical way. **By dragging and dropping jobs**, you can parent them to a new folder. To move them upward and downward in the queue you must change their priority.



Each column of this view shows a specific property of the job. The columns headers can also be used to sort the information. Just click on one of them and the view will be resorted. The default sorting is priority based.

As you can see each folder can contain multiple jobs. For viewing purposes they can be collapsed or expanded by clicking on the arrow on the left of the job status icon.

Jobs are arranged on a priority basis. This means that the jobs with the **higher** priority will be the first to be sent to the available instances.

When a job is a child of a folder it inherits some properties from its parent. In the specific, if the parent folder has a certain priority, even if the job has an higher one, it will be started only when the parent priority matches the current queue status. The priority of the job is valid when compared to the jobs belonging to the same parent. The same concept applies to the job destination pool. If it's set to target the entire farm but its parent specifies a specific pool, the job will override its setting and will be sent to the pool specified by the parent folder. The Console will show this with a pool name called "Derived from parent". If you select a different pool for the job, the derivation will be overridden.

An explanation of each column follows:

- **Job name:** This specifies the job name as well as its current status. In a similar fashion to the clients queue, the icon on the left of the job name can assume 10 different states:

   The folder is currently paused. The setting propagates to the childs

   The folder is active

   The job is on the queue and ready to start

 The job is paused. Until you resume it, it will never start

 The job is in progress

 The job is in progress but reported some kind of warnings

 The job has been completed but reported some kind of warnings

 The job is in progress but reported some kind of errors

 The job has been completed but reported some kind of errors

 The job has been completed successfully

 The job is locked. No operations are allowed on it until it's unlocked

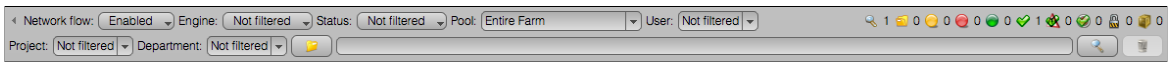
 The job is archived. It won't be visible until you change the view's filters

 The job is running a pre/post job action or the image assembler

- **L:** Shows a lock when the job is locked
- **T:** The job has a timed action (like pause or resume at a certain time)
- **ID:** The ID is an internal progressive number assigned to each job by the dispatcher. You should take care about this number when setting job's dependencies
- **Project:** Specify a custom string related to the job project
- **Department:** Specify a custom string related to the job department
- **Priority:** This is the priority for each job. The priority decides where a job is located inside the queue
- **Destination Pools:** This field shows the destination pool sfor the job.
- **Excluded Pools:** This field shows the excluded pools for the job.
- **Requeued:** At the end of each job, Muster can optionally check for missing frames. In case of a failure, one of more fractions of the job may be requeued. The requeued jobs will increase this counter. You can limit how many times a job or its fractions are requeued to avoid infinite loops for a faulty job.
- **Engine:** This field shows the targeted engine for the relative job
- **Started on:** For a started job, this field shows the exact time when the job has started

- **Progress:** For an in progress job, the field shows in percentage, the overall progress of the job. The progress does not include the in-rendering packets but it's computed only on completed packets. Depending on your settings, the progress bar may show the status for each chunk in a visual way
- **Completed on:** For a completed job, this field shows the exact time when the job has been completed. In case of an in progress job, the field shows the estimated ending time
- **Working nodes:** The field shows how many instances actually are used to process the job. An instance can include one or more processors belonging to a particular render client. This depends on render client configuration
- **Submission time:** The field shows the time when the job was submitted
- **Belonging to:** The field shows the name of the user that submitted the job
- **Starting frame:** The field shows the starting frame of the job where applicable
- **Ending frame:** The field shows the ending frame of the job where applicable
- **Total frames:** Shows the total number of jobs frames or the number of the slices (where applicable)
- **Filename:** The field shows the primary filename of the job (where applicable)
- **Notes:** Show additional notes for the job

If you want to filter the view contents, you can use the options on the filtering bar:



- **Network flow:** Enable or disable the flow of data to the view. If you disable the network flow, each host view will be disabled and you'll reduce the amount of network traffic between the Dispatcher and the Console. You should always disable a certain view network flow if you don't need the data shown to be updated
- **Engine:** Filters the contents by showing only jobs for certain templates
- **Status:** Filter the contents by showing only jobs with a certain status
- **User:** Filter the contents by showing jobs belonging to a certain user

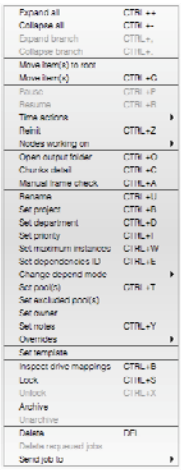
At the end of the filter bar, there are two buttons. The first one lets you create a new folder, while the second one lets you filter the contents of the view by searching a particular string in the jobs names.

Settings of the filtering bar will be stored persistently if you're working on a custom workspace. If you're working on the default, the settings will be reset on the next session. Check the workspaces section to learn more

Remember to disable the network flow on views you're not interested it. This will reduce the amount of traffic between Console and the Dispatcher!

Managing the jobs

By right-clicking on a job in the queue view (or making a multiple selection), you get a popup menu that lets you take control of one or multiple jobs:

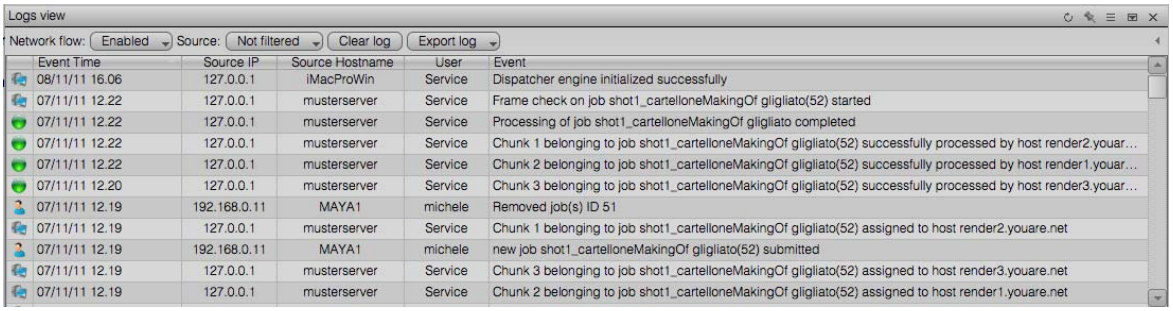


This is a brief explanation of each option:

- **Expand all:** Expands the entire queue tree
- **Collapse all:** Collapses the entire queue tree
- **Expand branch:** Expands the entire tree selected branch
- **Collapse branch:** Collapses the entire tree selected branch
- **Move item(s) to root:** Moves the selected item(s) to the root level
- **Move item(s):** Moves the selected item(s) inside the selected folder
- **Pause:** Pauses a job. A paused job won't be sent to clients for processing.
- **Resume:** Resumes a paused job
- **Time actions:** Pause or resume a job at a certain time
- **Reinit:** Reset the job to its initial state. If you have instances working on the job, the processes will be terminated
- **Nodes working on:** Perform actions on the instances currently working on the job
- **Open output folder:** If the job specifies an output folder, the command opens the folder to view the files
- **Chunks details:** Opens the chunks detail window

- **Manual frame check:** Opens the manual frame check dialog
- **Rename:** Renames the job
- **Set project:** Sets the unique job project string. This may be used later for filtering and arranging the jobs
- **Set department:** Sets the unique job's department string. This may be used later for filtering and arranging the jobs
- **Set priority:** Sets the job priority. Jobs with an higher priority will be processed first. Jobs that share the same priority are sorted by the ID number. Smaller IDs will be sent first
- **Set maximum instances:** Set the maximum number of instances allowed to work simultaneous on the job
- **Set dependencies IDs:** Set a list of jobs dependancies. If a job depends from another job, it will wait until it completes before being eligible for processing
- **Change depend mode:** Change the result mode required by the dependancies
- **Set pool(s):** Change the destination pools of the job
- **Set excluded pool(s):** Change the excluded pools of the job
- **Set owner:** Change the job owner
- **Set notes:** Set the custom notes on the job
- **Overrides:** Let you change the jobs behaviours overrides
- **Set template:** Change the job current template
- **Inspect drive mappings:** Let you check, if available, the drive mappings embedded in the job
- **Lock:** Locks the job. Any operation is forbidden until the job is unlocked
- **Unlock:** Unlocks a locked job
- **Delete:** Removes the job from the queue
- **Delete requeued jobs:** Removes the jobs requeued from the selected one
- **Send job to:** Send job properties to one of the available submission views of your workspace

The log view



The log pane is the most important Muster window. Messages reported by render clients or status/error messages reported by the Dispatcher are displayed here.

You should pay attention to its output to be able to track error and/or render failures.

The log shows the time the event has occurred, the text of the event, the user that thrown that event (**Service** refers to Dispatcher Service), the machine originating the event and its relative IP. The lines have different colours according to the kind of the event:

The log entry is an error. This may come from an instance activity or a dispatcher notice.

The log entry is a successfully event. This may come from an instance activity or a dispatcher notice.

The event comes from an user action

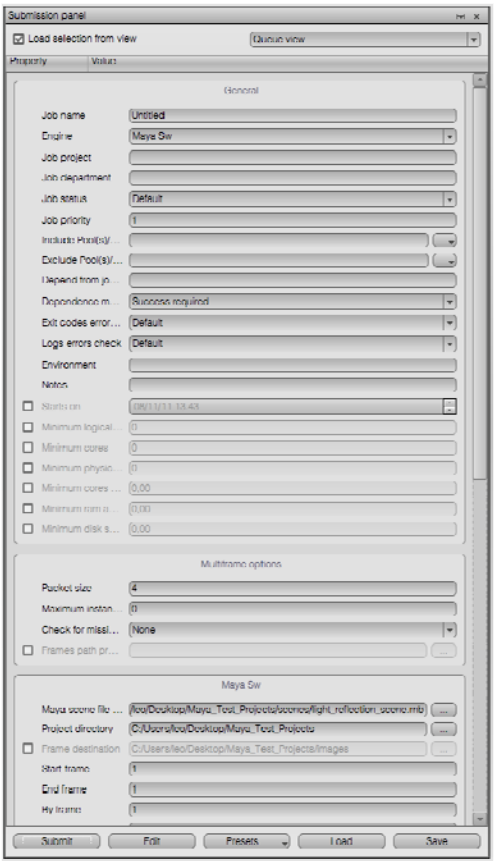
The event comes from a system action (Dispatcher)

If you want to filter the view contents, you can use the options on the filtering bar:

- **Network flow:** Enable or disable the data flow to the view. If you disable the network flow, each host view will be disabled and you'll reduce the amount of network traffic between the Dispatcher and the Console. You should always disable a certain view network flow if you don't need windows to be constantly updated
- **Source:** Filters the contents showing logs entries originating from a certain entity

You can clear the current log content by clicking the **Clear log** button. If you clear the log, it will be done on the Dispatcher side meaning that each Console will get its log cleared.

The submission view



The Submission view allow you to check and edit existing jobs properties and submit new jobs.

The job properties list is dynamic, and changes according to the selected job and its properties template.

At the top of the view, a combo box lets you choose which view automatically fills the list with the selection.

The buttons at bottom of the view allows to submit a new job, edit an existing one, save and load presets stored in XML based files, or access the presets manager that stores easily recallable settings from a persistent list.

The properties on the submission view depends on the selected engine but they share some common ones explained below:

General section

- **Job name:** Specifies the name for your new job or the ones you're editing
- **Engine:** This drop-down combo box shows the available render engines. By selecting one of them the submission view will update its fields reflecting the engine template properties
- **Job project:** Specifies the project of the job (custom string)
- **Job department:** Specifies the department of the job (custom string)
- **Job status:** Overrides the default status in the queue
- **Job priority:** Specifies the priority level for the job. Jobs with higher values will be processed first
- **Include Pool(s)/Hosts:** Specify one or more render pool to use for the render, you can also specify single hosts
- **Exclude Pool(s)/Hosts:** Specify one or more render pool to exclude from the render, you can also specify single hosts
- **Depend from job ID(s):** If you want to prevent the job processing until one or more jobs are completed, write their IDs there (comma separated)
- **Dependence mode:** Specify what kind of results Muster should expect from dependent jobs to allow the job processing
- **Notes:** You can put any kind of notes inside this field
- **Starts on:** Specify a starting time for the job
- **Minimum logical units:** Specify a minimum amount of virtual processors an host must have to be eligible to process the job
- **Minimum cores:** Specify a minimum amount of physical cores an host must have to be eligible to process the job
- **Minimum physical cpus:** Specify a minimum amount of physical processors an host must have to be eligible to process the job
- **Minimum cores speed:** Specify a minimum speed in GHz an host must have to be eligible to process the job
- **Minimum ram amount:** Specify a minimum amount of Ram, in Megabytes, an host must have to be eligible to process the job

- **Minimum disk space:** Specify a minimum amount of free space on the physical disks, in Megabytes, an host must have to be eligible to process the job

Multiframe options section

This section is available only for multi frame jobs. It contains the following fields:

- **Packet size:** This field specifies the size in frames for each chunk sent to a client. You should carefully balance this value carefully depending on your scene frame range and the processing power of your clients.
- **Maximum instances:** This field specifies the maximum number of render instances used by the job. A value of 0 instructs the job to use all the available instances.
- **Check for missing frames:** This field activates the automatic missing frames feature available on the Dispatcher. At the end of the job processing, the Dispatcher will scan for missing frames and requeue them automatically. Additional information can be found in the Missing frames section
- **Image name prefix:** If you use the missing frames features, you need to specify here the expected filename prefix as well as the frames path

Image slicing options section

This section is available only for single frame jobs. It contains the following fields:

- **Frame number:** Put here the frame to render as a single sliced image
- **Number of slices:** This value specifies the number of separate slices to create (basically it means how many machines will render the frame). At the end of the rendering, the slices will be reassembled by the Dispatcher
- **Image width:** Specify the width of the final image
- **Image height:** Specify the height of the final image
- **Aspect ratio:** Specify the aspect ratio of the final image. This value applies only to certain engines
- **Anti alias overlap:** The anti aliasing filters may work incorrectly on the image borders. To provide slices that may produce a valid result, you may need to increase the amount of pixels rendered on the border of the slices. Increasing this value will generate images that are a bit larger but correctly reassembled. Additional pixels are rendered on the slices but the final image will reflect the expected size.
- **Maximum instances:** This field specifies the maximum number of render instances used by the job. A value of 0 instructs the job to use any available instances

Broadcast options section

This section is available only for broadcast jobs. It contains the following fields:

- **Packet type:** Single instance/Every instance, by choosing this field you can tell Muster to send the job to every running instance, or only one process for each IP address.

Single host options section

This section is available only for single host jobs. It contains the following fields:

- **Delegated host**: You can specify the instance name or IP address of the host to be used.

Overrides

- **Process valid exit codes:** Overrides the expected valid exit codes (comma separated)
- **Process warning exit codes:** Overrides the expected warning exit codes (comma separated)
- **Process error exit codes:** Overrides the expected error exit codes (comma separated)
- **Valid process log's texts:** Overrides the expected string or regular expression to match valid log's texts
- **Warning process log's texts:** Overrides the expected string or regular expression to match warning log's texts
- **Error process log's texts:** Overrides the expected string or regular expression to match error log's texts
- **Chunks timeout:** Overrides the job chunks timeout
- **Chunks maximum requeue:** Overrides the maximum number of times a chunk can be requeued
- **Mail address:** Overrides the destination email address to send notifications
- **Override job notifications:** By checking this field, you can override the default notification event related to jobs notifications
- **Override chunks notifications:** By checking this field, you can override the default notification event related to chunks notifications

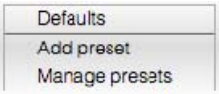
Actions

- **Action:** Specifies the executable to launch as a pre or post job/chunk action

- **Check return code:** Tells Muster to check for the action return code and abort processing according
- **Override timeout:** Overrides the default timeout for the action

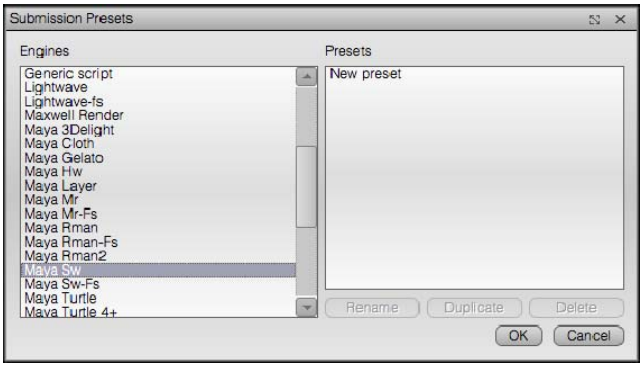
Managing submission presets

The submission panel lets you store and manage properties presets. Preset are retrievable through a pop-up menu and are stored on a template basis:



If you click the **Add preset** menu entry, the current values of the submission dialog are stored as a new preset and you're prompted for the preset's name.

By clicking the **Manage presets** menu entry, you can Rename, Delete or Duplicate existing presets.

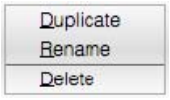


Presets are always stored on a template basis, there's no way to duplicate a preset done for a different template than the one it belongs to

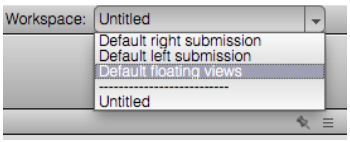
Managing workspaces and views

Muster Console supports customizable workspaces. A workspace contains a set of views and their settings, and stores them persistently across different sessions.

You can access the workspaces functions by clicking the icon next to the workspaces selector combo box in the right corner of the menu bar:



By using the menu entries, you can **Duplicate**, **Rename** and **Delete** the current workspace. If you want to create a new workspace, you must start duplicating current; this will inherit your current settings and will create the new workspace. Once you create a new workspace, it will be available in the workspaces list:

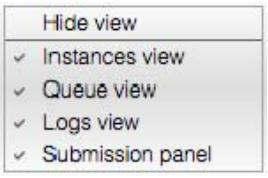


Muster Console supports three default workspaces. You cannot Delete or Rename them, but you can Duplicate them to originate new workspaces.

The settings of the views like their columns ordering and filtering are stored with the workspace. If you're working on the default, you'll lose your settings each time you exit from the Console. Always create your own workspace if you want to keep your settings!

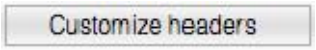
From the view menu, you can create a new view selecting the **Create new view** submenu. Once you create a new view, you can manage the state of it using the buttons at the top of it:

The buttons let you rename, hide or destroy a view. Remember that once you hide a view using the central button, you can recall it using the views popup menu, right clicking on a view header or on the empty workspace area of Console (if no view is visible). This is a typical views popup menu you may get:

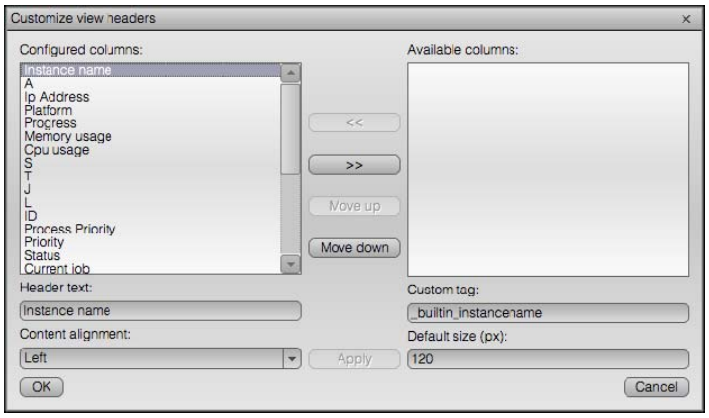


Customizing a view

If you want to customize the columns of a view, you can right click on its headers and invoke the popup menu:



If you click the **Customize headers** entry, you'll get the following dialog:



The customize view headers window lets you choose exactly which column you want to show, its position in the list, its content alignment and the default column size.

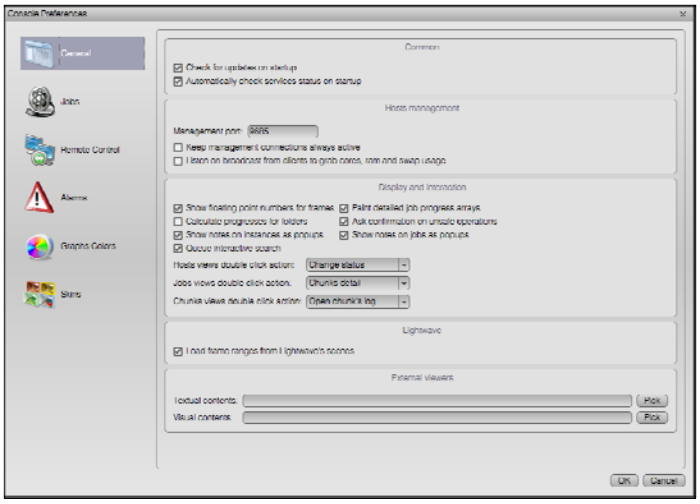
Customizing the view's columns and the view's filters is a good way to have multiple views with different contents available in the workspace.

If you're working on a custom workspace, the settings changed by this window will be stored persistently with the view

Muster Console preferences

The Muster Console can be configured selecting *Preferences* from the menu or clicking the relative button on the toolbar.

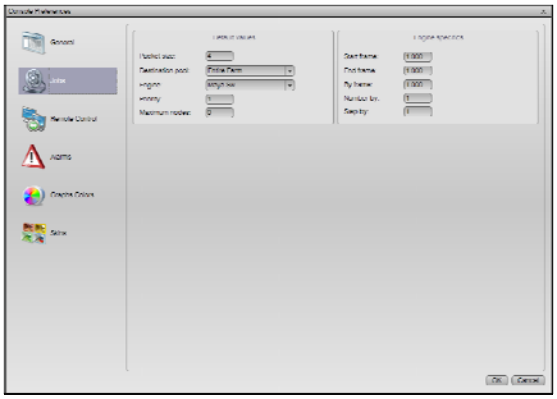
The general section has the following options:



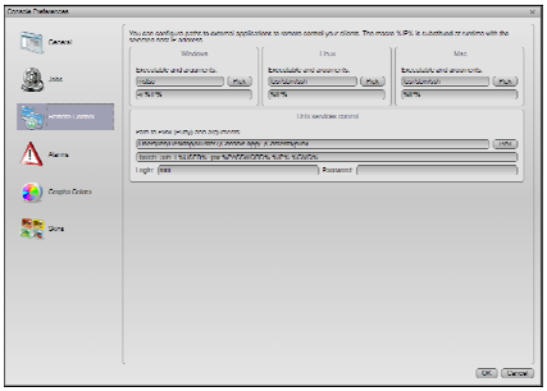
- **Check for updates on startup:** Tells the Console application to connect to the Virtual Vertex site checking for updates each time you start it
- **Automatically check services status on startup:** When active, the Console will attempt to query the service status of any host available in the hosts views. This may be network and resource consuming, so depending on the size of your farm, you may need to disable this and check the services statuses manually when required
- **Management port:** This configures the management port used to connect to the Dispatcher service
- **Keep management connections always active:** When you configure a client, a direct connection is established to get its properties and to send the configuration back. If you check this option, the connection is persistent
- **Listen on broadcast from clients to grab Cpu and Ram usage:** If status broadcast is enabled on a render client , console checks for this traffic and updates the usage counters according
- **Show floating point numbers for frames:** If you're going to work with floating point frames numbers, you may need to activate this option for a correct feedback of the frames numbers
- **Paint detailed job progress arrays**: Paints a progress bar that reflects the status of each chunk. This may be resource intensive, so depending on the size of your queue, and the number of the jobs, you may need to deactivate this option

- **Calculate progresses for folders:** Calculates the folders progresses by averaging the progress of the childs
- **Ask confirmation on unsafe operations**: If activated, when you make an unsafe operation, like deleting a job in the queue, Muster Console will show a confirmation dialog
- **Show notes on instances as popups:** If active, when you hover with your mouse on an instance icon you'll see a popup with the host's notes
- **Show notes on jobs as popups:** If active, when you hover with your mouse on a job icon you'll see a popup with the job's notes
- **Queue interactive search:** When changed the search filter into the queue, Muster recalculates the result on each keystroke, this may be cpu intensive when dealing with big queues
- **Hosts views double click action:** Choose an action to perform when you double click on a hosts/instance
- **Jobs view double click action:** Choose an action to perform when you double click on a job
- **Chunks views double click action:** Choose an action to perform when you double click on a chunk
- **Load frame range from Lightwave's scenes:** If you're working with Newtek Lightwave, you can tell Console to load the frame range from the selected .LWS file when you pick one from the submission view.
- **Textual contents:** Defines an external viewer used to open textual contents
- **Visual contents:** Defines an external viewer used to open image files

The jobs section lets you define several default values for the submission views:



The remote control section lets you define how to remote control an host when you select the **Remote Control** function from the popup menu:

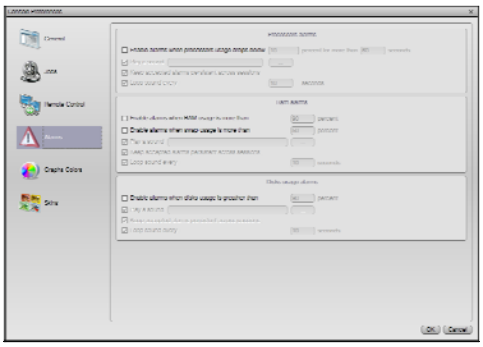


The default values on Windows use **Remote Desktop** to access Windows hosts, and **Putty** to access Unix hosts through an **ssh** connection while Unix's Console uses **Vnc** or **Ssh**. You should configure the command line according to your environment and your preferred software.

Particular attention should be paid to the **Unix services control.**
While Windows is able to query remote services using its built-in API, we rely on a modified version of **Plink** (part of the **Putty** suite) to access Unix hosts through an SSH connection and check the Services remote status. You should not change the command line preconfigured unless you've particular needs. Just be sure to specify a root username and password in the preferences and open the **SSH** port on the remote hosts.

While you can use the Windows API and Plink from a Windows hosts running Console to query any platform service status, there's no way to query a Windows service status from a Linux or Mac OS X platform!

The alarms section let you define local alarms thresholds on the hosts



The graphs colours section lets you define the maximum amount of statistics hold in memory by the Console and the colours used by the various graphs:

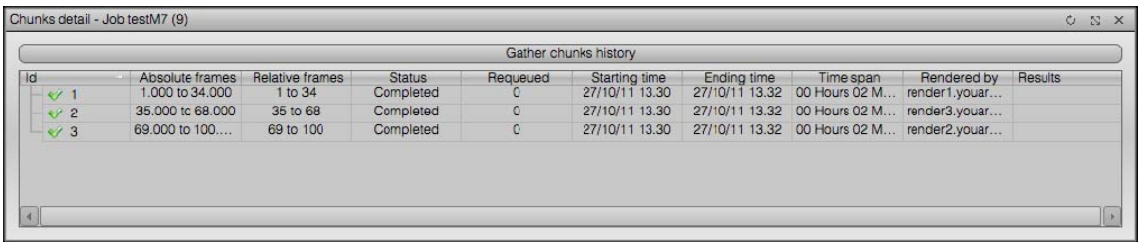
The skin section let you modify or define new skins to be used by Console. The syntax of the skinning language is based on the QT CSS cascading style sheets. For further information on how to modify the skins, please reefer to the QT4 CSS style documentation available on the QT website at qt.nokia.com

If you need to reset the Console preferences or store them for future reference, they can be found into the registry for Windows (**HKEY_CURRENT_USER\Software\Virtual Vertex\Muster\6\Console)** and inside your home folder (**.Muster**) on Linux and Mac OS X

The Chunks detail

The Chunks detail is a very useful summary window that lets you manage packets (called chunks) that build a job. On multi frame and image slicing jobs, each chunk is a logical representation of a processing phase. On broadcast jobs, each chunk is dedicated to an instance or an host. Single host jobs do not have chunks.

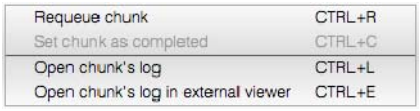
This is a typical view of the chunks detail window:



Each packet is numbered with its internal ID and reports the following information:

- **Absolute Frames:** Those are the effective frames of your sequence chunk. They may not reflect the number appended on the frame name.
- **Relative Frames:** Those are the frames numbers that will be generated. They may not reflect the absolute frames but will match with the final files. The first packet reflects the value of the starting frame value of a job.
- **Status:** This field reports the current status of the chunk. It can assume 4 different state and reflects the colour of the icon on the left of the chunk's ID. Possible states are: On hold, In progress ,completed or completed with warnings.
- **Requeued:** When a chunk is re-queued using the automatic missing frames feature, this field will be increased each time the process occurs.
- **Starting time:** This field shows the starting time when a chunk has been submitted to a client. It remains blank for a "On hold" packet.
- **Ending time:** This field shows the ending time for a completed chunk.
- **Time span:** This field shows the effective amount of time taken by a completed chunk for its processing.
- **Rendered by:** This field shows the name of the instance that has rendered the chunk. If chunks are flagged as completed by an user, the field shows "Forced by console".
- **Results:** If a packet completed with some kind of error, it will be reported inside this field otherwise the field will stay blank.

To manage chunks , right click on them and you'll get the following pop-up menu:

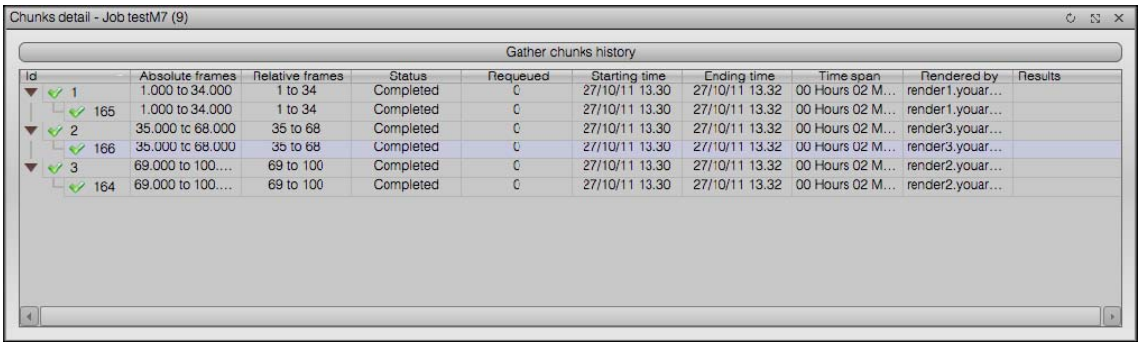


Operations permitted by this menu are simply the requeuing of a completed chunk or the setting of the chunk in a completed status. When forcing a chunk to a different status, render clients that are currently rendering the chunk may receive an abort message.

If you select the Open chunk's log item, the logs of the hosts that processed the chunk is opened, and the related file selected automatically.

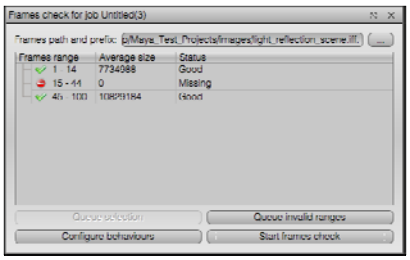
A special type of chunk is **the Image assembler** one. This is a chunk that's always found at the end of a single frame job. It's included inside the chunk view only for consistency but actually rendered by the Dispatcher service when the previous packets have been completed. Its purpose is to assemble slices created by the instances.

The chunks view reports just the chunks of the current job status, if you want to compare the entire lifecycle of the job, you can click the "Gather chunks history" button and have a tree based view with the results of each chunks in the history. This let you also check why a particular chunks has failed and also recover the logs of the failed or requeued chunks.



The frames check window

The missing frames window will pop up when you right click on a job and select "**Manual frame check**":



The dialog lists the job frames grouped in a logical way. Each missing or good sequence of consecutive frames is reported with a summary of the average size in bytes. The frame parser can be configured to check only the existence of the files or to check the file size.

To start the frame check, just press the "**Start frames check**" button. Remember that, if you have not specified a frames path and prefix during job submission, you may need to specify them in the frames check dialog. The prefix of the file must be include the path and any character that precedes the number that identifies the actual frame number.

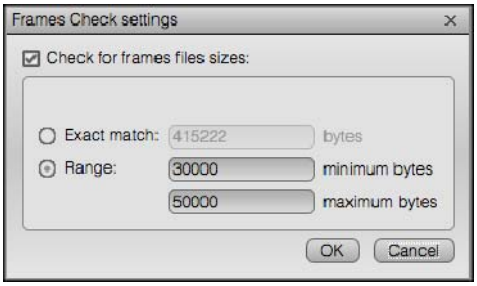
For example , if your final renders are saved inside the folder `\\Myserver\myrenders` and have a name similar to this:

```
Scene1_0001.tga

Scene1_0002.tga

Scene1_0003.tga
```

the prefix you've to supply is `\\Myserver\myrenders\Scene1_`

After running the frames check, you can automatically queue the wrong packets pressing the "**Queue wrong packets**" button or just queue the list's selection using the "**Queue selection**" button.

If you want to configure the parser, select **Configure behaviours**:

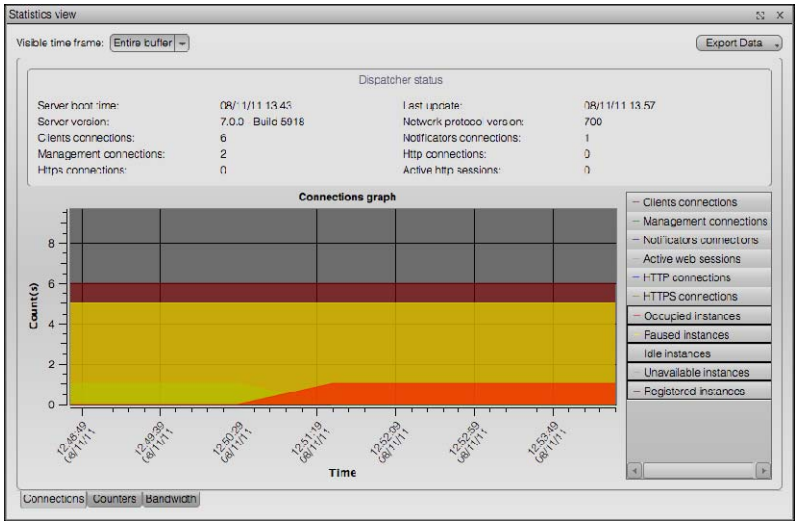


The parser can also check the files sizes between a minimum and a maximum value. This is particular useful if you are rendering frames in a compressed file format where file sizes can change between each frame.

If you disable the size parser, Muster will check only the frame existence.

Browsing the network statistics

Each time you open the Console, it receives some statistics from the Dispatcher on a regular basis. You can access such statistics through the **Statistics view** menu entry in the **View** menu:

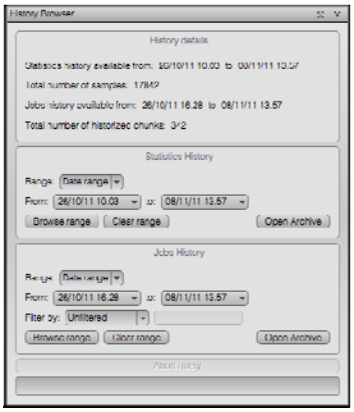


You can use the button menu in the upper right corner of the window to export the current data. You can then reload the data using the **History view** explained later.

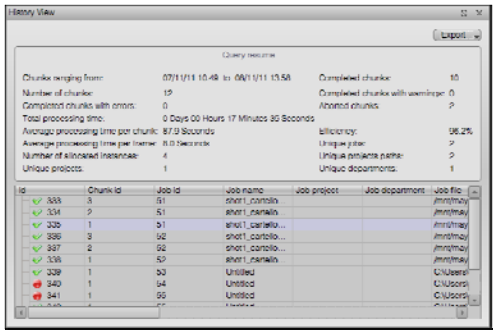
Managing the history

The history view can be used to retrieve the processing and the usage history of the Dispatcher itself, store queries and reload saved archives.

You can access the history browser by selecting **History view** from the **View** menu:



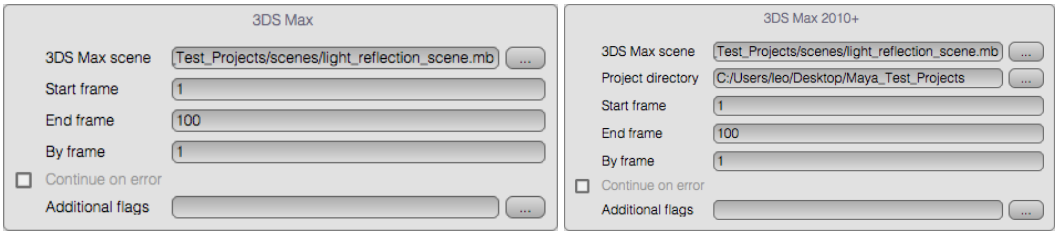
If you query the jobs history you'll get a resume of the selected period, you can then export the data in a proprietary format, or to well known formats like PDF.



Engine specific submission view options

The following section explains parameters available in the submission dialog for each supported render engine:

3D Studio Max (3DS Max)



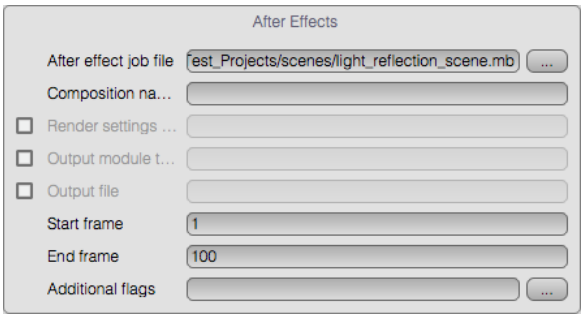
You have two versions of the Max template, one for Max versions from 6.0 to 9.0, the other for versions 2010+.

- 3DS Max Scene: The job file to render
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Continue on error: Tells Max to continue if it encounters an error during the batch render
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

3D Studio Max Frame slicing (3DS Max 9+ Fs)

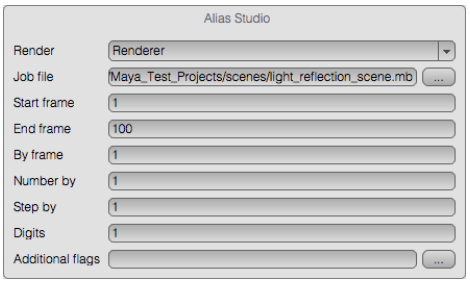
- 3DS Max Scene: The job file to render
- Frame destination: The full path where you want to store the slices and the assembled image
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

After Effects



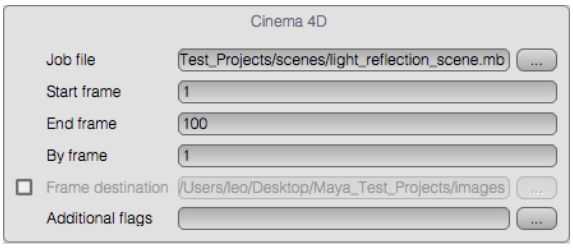
- After effect job file: The job file to render
- Composition name: Specifies the composition to render
- Render settings template: If you want to override the render settings template, write the name of the template to be used
- Output module template: If you want to override the output module settings, write the name of the module to be used
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Alias Studio



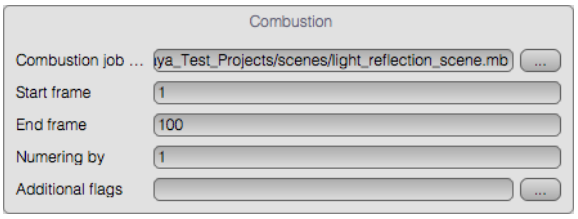
- Render: Specifies the Alias Studio renderer to use
- Job file: The job file to render
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Number by: Specifies the number to use to start numbering the frames sequence
- Step by: Step for the final frames numbering
- Digits: Number of digits to use for the frame numbering
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Cinema 4D (C4D)



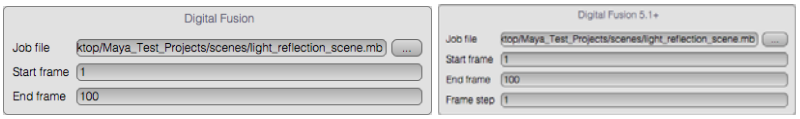
- Job file: The job file to render
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Frame destination: Overrides the destination for the rendered images
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Combustion



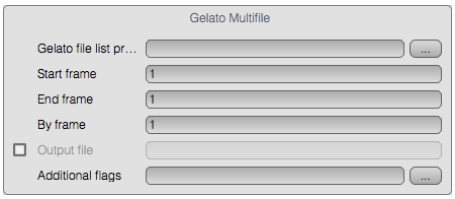
- Combustion job file: The job file to render
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- Numbering by: Specifies the number to use to start numbering the frames sequence
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Digital Fusion



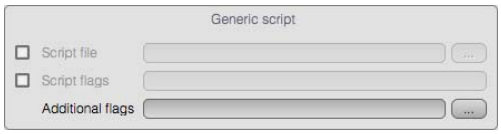
- Job file: The job file to render
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame

NVidia Gelato (Gelato Multifile)



- Gelato file list prefix: The prefix (full path and filename up to the numbering portion) to render
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Output file: Overrides the destination for the rendered images
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

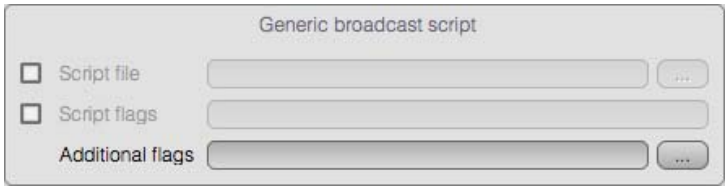
Generic script



The first available host will execute the generic script and it's a one shot script. You can use an host name inside the destination pool to target the script to a specific host.

- Script file: The script file to be executed. This will be executed through CMD.exe on Windows and through /bin/sh from Mac and linux
- Script flags: Flags to be passed to the script
- Additional flags: Additional flags to be passed to the script

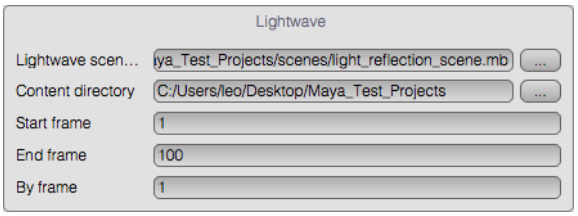
Generic broadcast script



Every host registered inside Muster will execute the generic broadcast script. You can filter the number of hosts involved by specifying a destination pool.

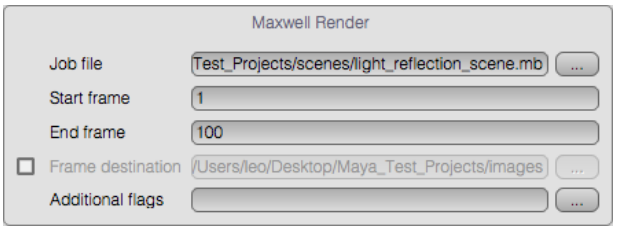
- Script file: The script file to be executed. This will be executed through CMD.exe on Windows and through /bin/sh from Mac and linux
- Script flags: Flags to be passed to the script
- Additional flags: Additional flags to be passed to the script

Newtek Lightwave



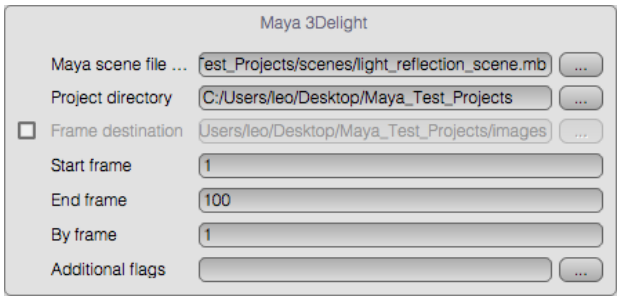
- Lightwave scene file name: The job file to render
- Content directory: Full path to the root directory of your project
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence

Maxwell Render



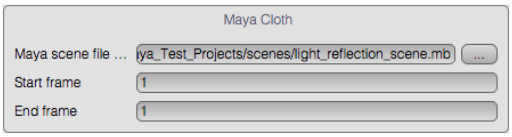
- Job file: The job file to render
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- Frames destination: Overrides the destination for the rendered images
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Maya 3Delight



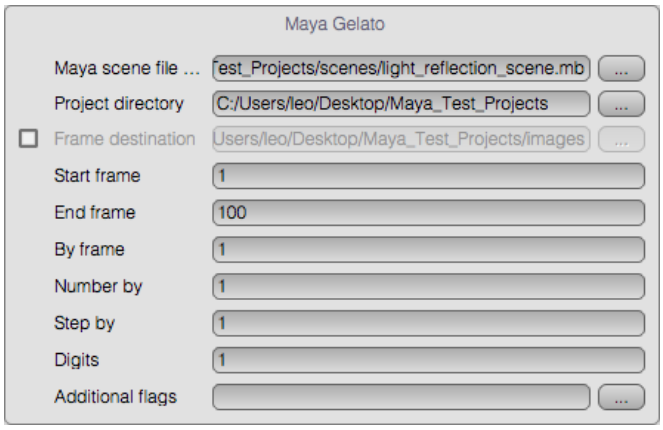
- Maya scene file name: The job file to render
- Project directory: Full path to the root directory of your project
- Frame destination: Overrides the destination path for the rendered images
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Maya Cloth



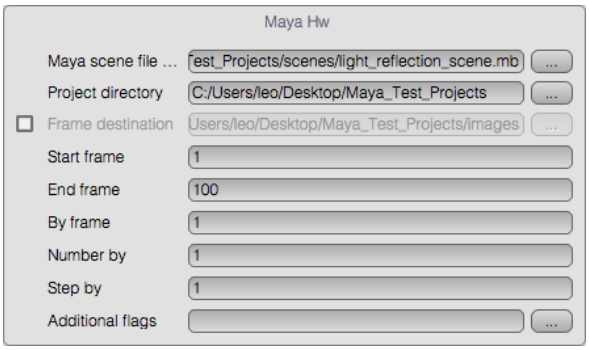
- Maya scene file name: The job file to render
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame

Maya Gelato



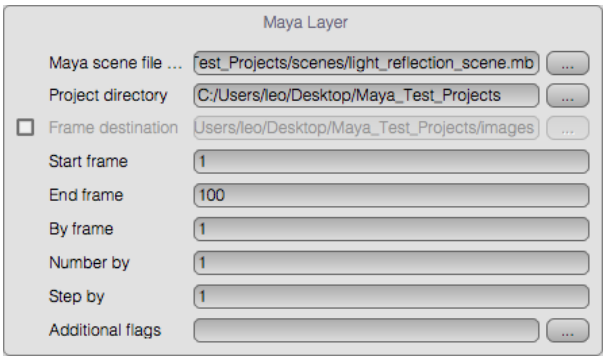
- Maya scene file name: The job file to render
- Project directory: Full path to the root directory of your project
- Frame destination: Overrides the destination path for the rendered images
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Number by: Specifies the number to use to start numbering the frames sequence
- Step by: Step for the final frames numbering
- Digits: Number of digits to use for the frame numbering
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Maya Hardware



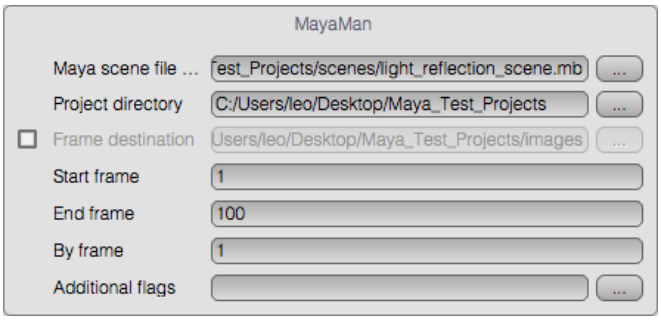
- Maya scene file name: The job file to render
- Project directory: Full path to the root directory of your project
- Frame destination: Overrides the destination path for the rendered images
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Number by: Specifies the number to use to start numbering the frames sequence
- Step by: Step for the final frames numbering
- Digits: Number of digits to use for the frame numbering
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Maya Layer



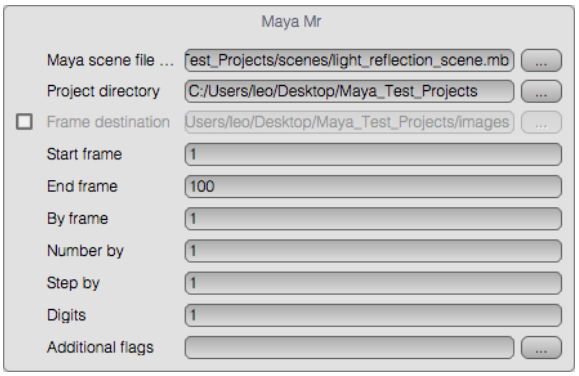
- Maya scene file name: The job file to render
- Project directory: Full path to the root directory of your project
- Frame destination: Overrides the destination for the rendered images
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Number by: Specifies the number to use to start numbering the frames sequence
- Step by: Step for the final frames numbering
- Digits: Number of digits to use for the frame numbering
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

MayaMan



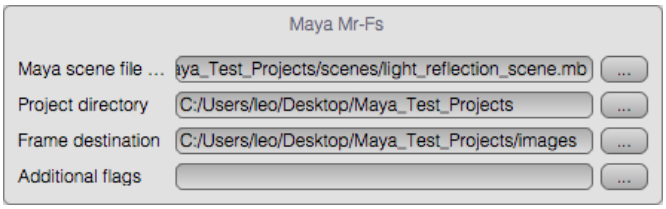
- Maya scene file name: The job file to render
- Project directory: Full path to the root directory of your project
- Frame destination: Overrides the destination for the rendered images
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Maya Mental Ray (Maya Mr)



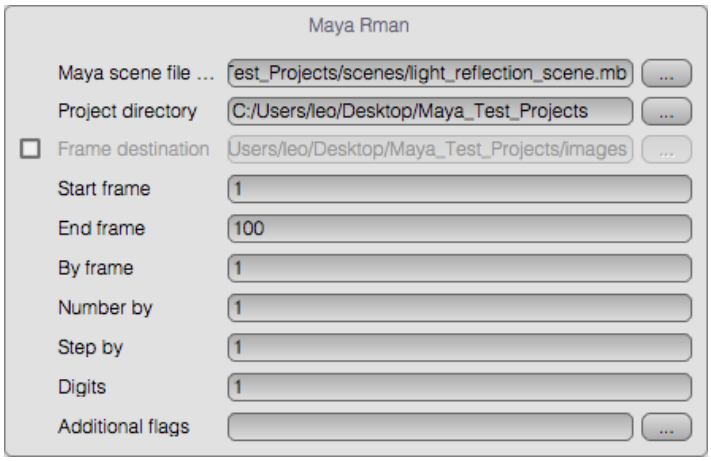
- Maya scene file name: The job file to render
- Project directory: Full path to the root directory of your project
- Frame destination: Overrides the destination for the rendered images
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Number by: Specifies the number to use to start numbering the frames sequence
- Step by: Step for the final frames numbering
- Digits: Number of digits to use for the frames numbering
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Maya Mental Ray Frame slicing(Maya Mr-Fs)



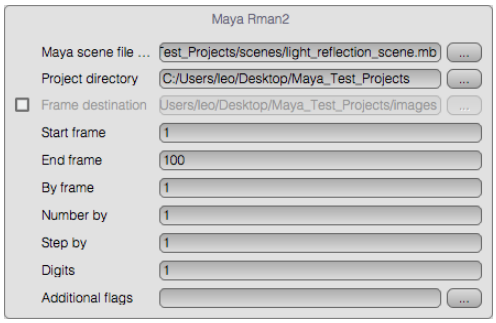
- Maya scene file name: The job file to render
- Project directory: Full path to the root directory of your project
- Frame destination: The full path where you want to store the slices and the assembled image
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Maya Renderman (Maya Rman)



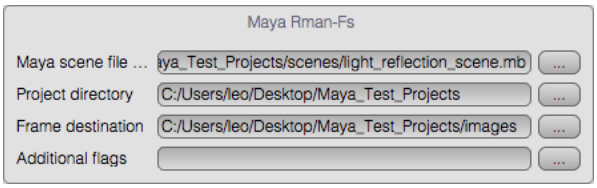
- Maya scene file name: The job file to render
- Project directory: Full path to the root directory of your project
- Frame destination: Overrides the destination for the rendered images
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Number by: Specifies the number to use to start numbering the frames sequence
- Step by: Step for the final frames numbering
- Digits: Number of digits to use for the frame numbering
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Maya Renderman 2.0 or greather (Maya Rman2)



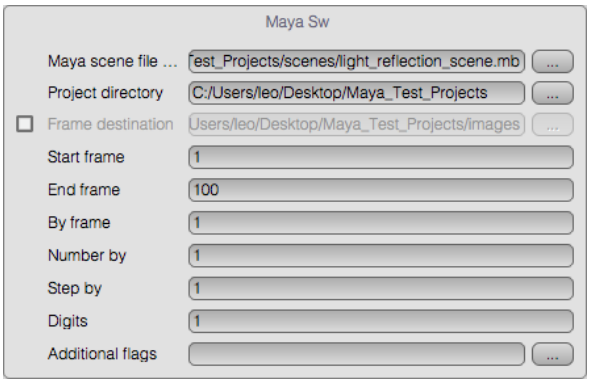
- Maya scene file name: The job file to render
- Project directory: Full path to the root directory of your project
- Frame destination: Overrides the destination for the rendered images
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Number by: Specifies the number to use to start numbering the frames sequence
- Step by: Step for the final frames numbering
- Digits: Number of digits to use for the frame numbering
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Maya Renderman Frame slicing (Maya Rman-FS)



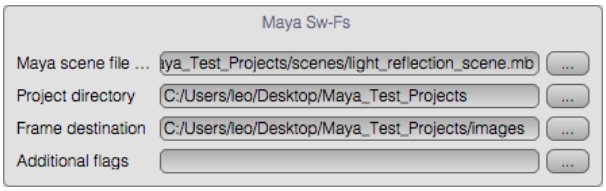
- Maya scene file name: The job file to render
- Project directory: Full path to the root directory of your project
- Frame destination: The full path where you want to store the slices and the assembled image
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Maya Software Render (Maya Sw)



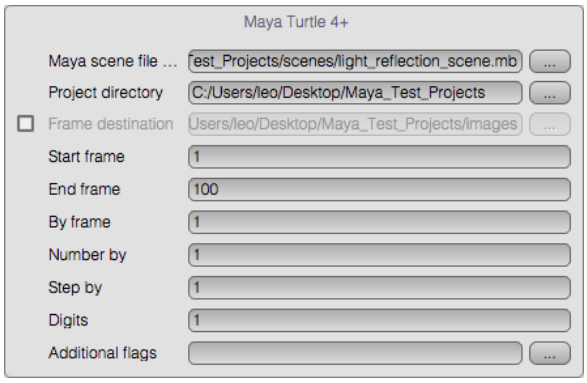
- Maya scene file name: The job file to render
- Project directory: Full path to the root directory of your project
- Frame destination: Overrides the destination for the rendered images
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Number by: Specifies the number to use to start numbering the frames sequence
- Step by: Step for the final frames numbering
- Digits: Number of digits to use for the frame numbering
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Maya Software Render image slicing (Maya Sw-Fs)



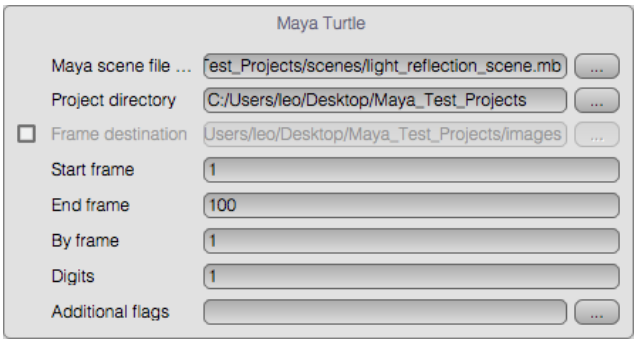
- Maya scene file name: The job file to render
- Project directory: Full path to the root directory of your project
- Frame destination: The full path where you want to store the slices and the assembled image
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Maya Turtle v. 4+ (Maya Turtle 4+)



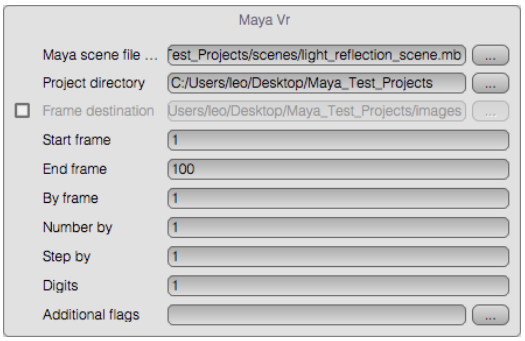
- Maya scene file name: The job file to render
- Project directory: Full path to the root directory of your project
- Frame destination: Overrides the destination for the rendered images
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Number by: Specifies the number to use to start numbering the frames sequence
- Step by: Step for the final frames numbering
- Digits: Number of digits to use for the frame numbering
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Maya Turtle up to v.3 (Maya Turtle)



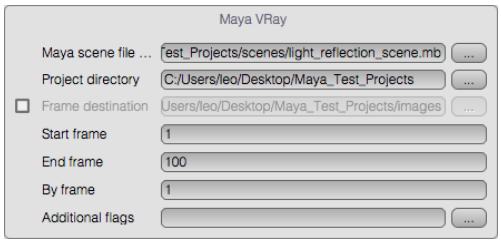
- Maya scene file name: The job file to render
- Project directory: Full path to the root directory of your project
- Frame destination: Overrides the destination for the rendered images
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Digits: Number of digits to use for the frame numbering
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Maya Vector Render (Maya Vr)



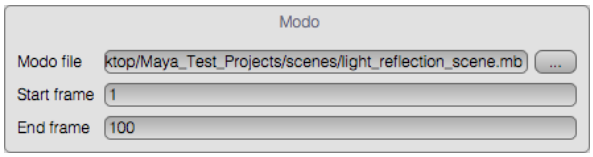
- Maya scene file name: The job file to render
- Project directory: Full path to the root directory of your project
- Frame destination: Overrides the destination for the rendered images
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Number by: Specifies the number to use to start numbering the frames sequence
- Step by: Step for the final frames numbering
- Digits: Number of digits to use for the frame numbering
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Maya VRay



- Maya scene file name: The job file to render
- Project directory: Full path to the root directory of your project
- Frame destination: Overrides the destination for the rendered images
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

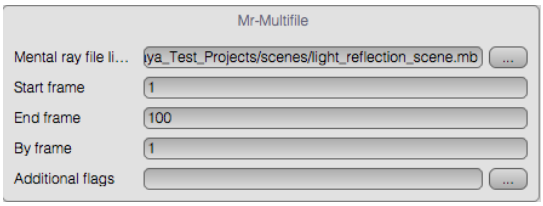
Modo



- Modo file: The job file to render
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame

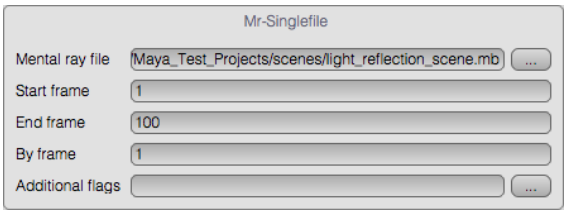
To successfully configure Modo to render through Muster, you need to copy on every Modo installation across the clients, the file **ModoMusterRender.pl** inside the scripts folder of each modo installation.

Mentalray Standalone multiple files (Mr-Multifile)



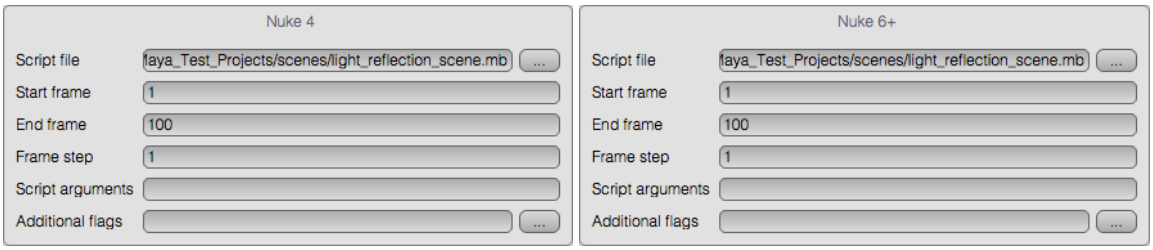
- Mental ray file list prefix: The prefix (full path and filename up to the numbering portion) to render
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Mentalray Standalone single file (Mr-Singlefile)



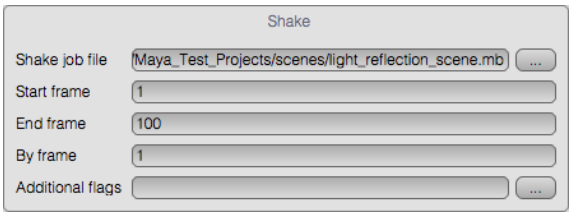
- Mental ray file: The .mi file to render
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Nuke 4-5 or 6+



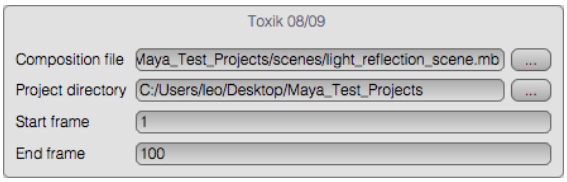
- Script file: The job file to render
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Script arguments: Arguments to pass to the script
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Shake



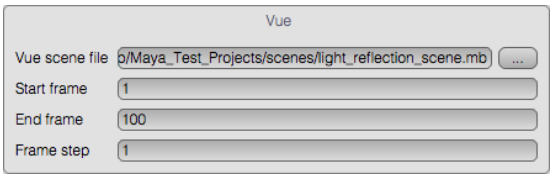
- Shake job file: The job file to render
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Toxik 2008/2009



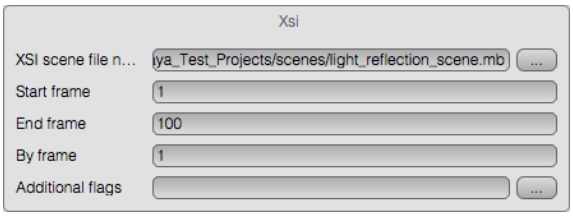
- Composition file: The job file to render
- Project directory: Full path to the root directory of your project
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame

Vue



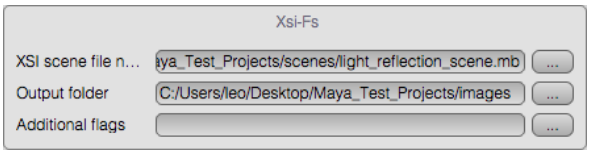
- Vue scene file: The job file to render
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- Frame step: Frames step in the final sequence

Xsi



- XSI scene file name: The job file to render
- Start frame: The final sequence starting frame
- End frame: The final sequence ending frame
- By frame: Frames step in the final sequence
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Xsi frame slicing (Xsi-Fs)



- XSI scene file name: The job file to render
- Output folder: The full path where you want to store the slices and the assembled image
- Additional flags: This field can be filled with additional flags to pass to the batch render command line

Managing Lightwave jobs

If you're going to render with Lightwave, some additional setup is required:

To find the plug-ins referenced by a job, you've to create a custom lw3ext.cfg file where plug-ins entries are located on a shared drive (You can find this file in the root folder of your user account). In this way, every slave will find the plug-ins and you've also the ability to start the lwsn.exe from a remote drive without installing Lightwave on the farm.

This is a sample **lw3ext.cfg** file:

```
{ Entry
Class "AnimLoaderHandler"
Name "AVI(.avi)"
Module "c:\\LightWave\\programs\\plugins\\Input-Output\\AVI.p"
}
{ Entry
Class "AnimLoaderHandler"
Name "QuickTime"
Module "c:\\LightWave\\programs\\plugins\\Input-Output\\QTTools.p"
}
{ Entry
Class "AnimSaverHandler"
Name "4XStoryboard"
Module "c:\\LightWave\\programs\\plugins\\Input-
Output\\VideoTap.p"
InfoTag 409 "4X Storyboard"
}
{ Entry
Class "AnimSaverHandler"
Name "AVI(.avi)"
Module "c:\\LightWave\\programs\\plugins\\Input-Output\\AVI.p"
}
```

This configuration will not work for Muster (or will work if you copy all your plug-ins in the same folder for your entire farm). It must be modified in this way (assuming that Z is your network folder) and that you've copied your copy of Lightwave in Z:\RenderFarm\Lightwave:

```
{ Entry
Class "AnimLoaderHandler"
Name "AVI(.avi)"
```

```
Module "Z:\RenderFarm\\LightWave\\programs\\plugins\\Input-
Output\\AVI.p"
}
{ Entry
Class "AnimLoaderHandler"
Name "QuickTime"
Module "Z:\RenderFarm\\LightWave\\programs\\plugins\\Input-
Output\\QTTools.p"
}
{ Entry
Class "AnimSaverHandler"
Name "4XStoryboard"
Module "Z:\RenderFarm\\LightWave\\programs\\plugins\\Input-
Output\\VideoTap.p"
InfoTag 409 "4X Storyboard"
}
{ Entry
Class "AnimSaverHandler"
Name "AVI(.avi)"
Module "Z:\RenderFarm\\LightWave\\programs\\plugins\\Input-
Output\\AVI.p"
}
```

Save the file always as lw3ext.cfg and put it in a shared folder (i.e. Z:\RenderFarm) and specify this folder in the Lightwave section of the templates settings on each client.

Dispatcher service Reference

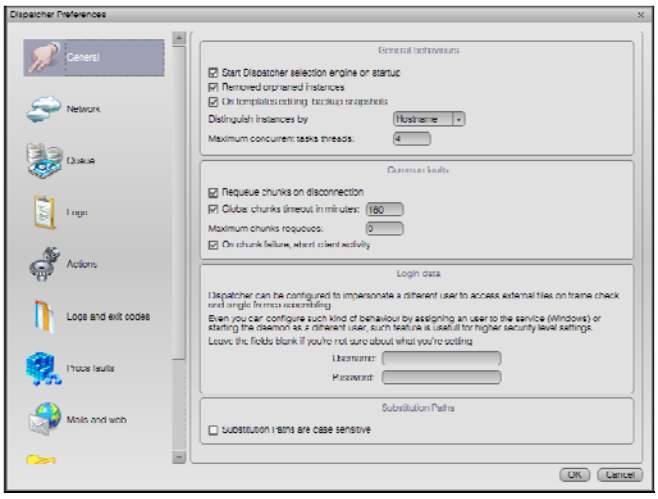
The configuration of the Dispatcher service is done through the Console. You can configure the following items/functionalities:

- **Dispatcher preferences**
- **Render pools**
- **Unix substitution paths**
- **Muster users database**
- **Muster mailing lists**

Dispatcher preferences

The dispatcher preferences window lets you configure every behaviour of the Dispatcher service:

The following window shows the General properties:



- **Start Dispatcher selection engine on startup:** Tells Muster to automatically start the Dispatcher selection engine after the boot sequence
- **Remove orphaned instances:** When an instance connects, Muster checks the total amount of instances for that node, and remove previously registered ones, if present.

- **On templates editing, backup snapshots:** When you change the templates on the Dispatcher service, an entire snapshot of the current status is backed up on a dedicated folder into the templates folder

- **Distinguish instances by:** This tells the Dispatcher what parameter to use to distinguish the render client instances. If you change from IP address to host name (i.e.) you're allowed to use a DHCP to assign the addresses to the render farm

- **Maximum concurrent tasks threads:** Defines the maximum amount of tasks (Image assembling, pre/post job actions) runnable by the Dispatcher

- **Requeue chunks on disconnection:** If an instance is disconnected and flagged offline, tells Muster to requeue the chunk assigned to that particular instance

- **Global chunks timeout in minutes:** Defines a global amount of time in minutes to use as a timeout value for chunks processing

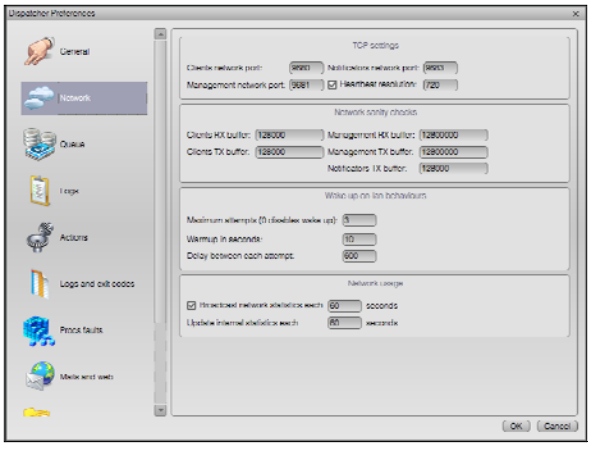
- **Maximum chunks requeue:** Defines a global maximum amount of times a chunk can be requeued

- **On chunk failure, abort client activity:** If the timeout for a chunk expires, Muster tries to abort the client activity that may be locked

- **Substitution Paths are case sensitive:** Defines the sensitivity of the substitution paths case

If you want to impersonate a different user during Dispatcher file access operations like Image slicing assembling, you can force it here filling the Username and password fields.

The following window shows the Network properties:

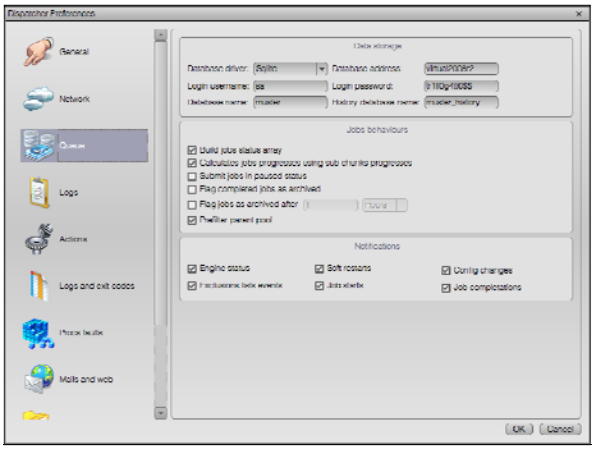


- **Clients network port:** Defines the TCP/IP network port used to listen for incoming instances connections

- **Management network port: :** Defines the TCP/IP network port used to listen for incoming management connections

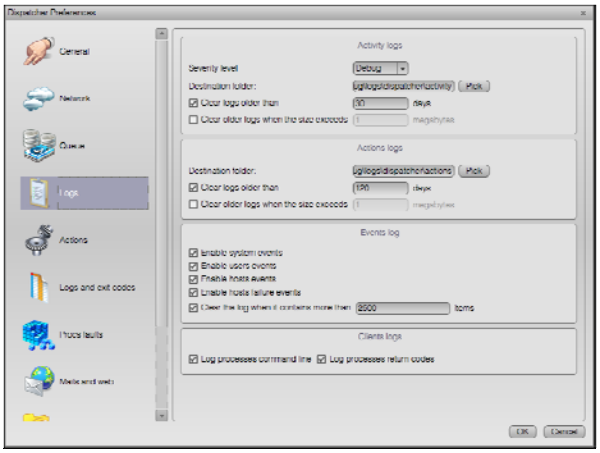
- **Notificators network port:** Defines the TCP/IP network port used to listen for incoming notificators connections
- **Heartbeat resolution:** Defines an interval to be used to send pulses to the clients to check they are still alive
- **Clients RX buffer:** Defines the amount of bytes storable in the network buffers
- **Clients TX buffer:** Defines the amount of bytes storable in the network buffers
- **Management RX buffer:** Defines the amount of bytes storable in the network buffers
- **Management TX buffer:** Defines the amount of bytes storable in the network buffers
- **Notificators TX buffer:** Defines the amount of bytes storable in the network buffers
- **Maximum attempts:** Defines the maximum number of attempts the Dispatcher does to wake up an host using the **Magic Packet** technology. A value of 0 disables the wake up globally
- **Warm up in seconds:** To give time on a full reboot , you can specify a warm up in seconds required before starting the Wake up logic
- **Delay between each attempt:** Defines the delay between each wakeup attempt
- **Update internal statistics each:** Defines an interval to use to update and store the internal status of the Dispatcher.
- **Broadcast network statistics each:** Defines an interval to use to broadcast the internal status of the Dispatcher to Consoles.

The following window shows the Queue properties:



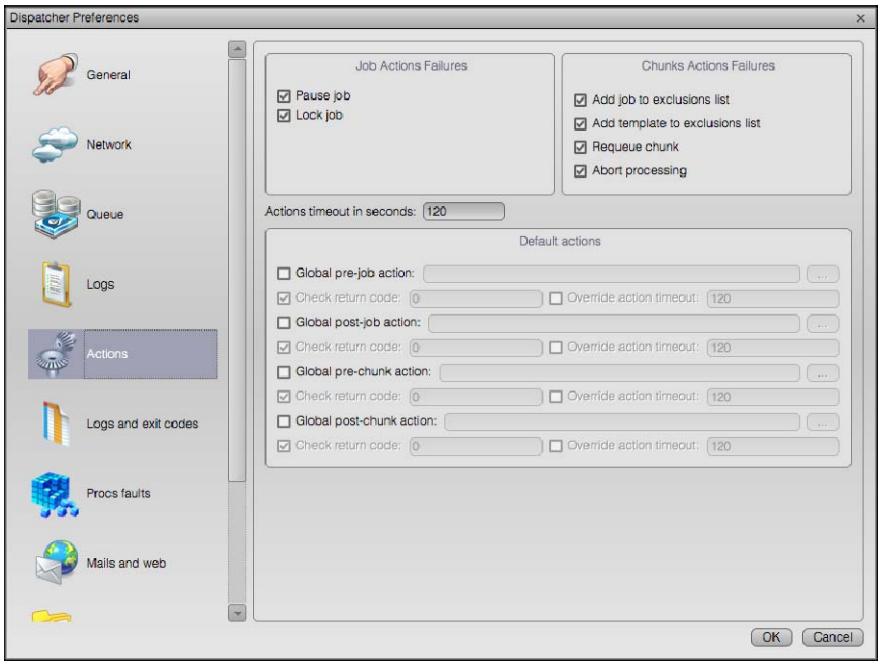
- **Database driver:** Choose the driver to use to access the Muster database (Sqlite, Mysql or Sql server). You're allowed to use Sql server only on a Windows based Dispatcher.
- **Database address:** The IP address of the database if applicable
- **Login username:** The login name to access the database
- **Login password:** The password to access the database
- **Database name:** The name of the Muster database
- **History database name:** The name of the Muster history database
- **Enable in-process cache:** Enable the in-memory caching of the database data improving performances but increasing memory usage
- **Build jobs status array:** Builds an array of statuses for each job to display the status of each chunk in the progress bar
- **Submit jobs in paused status:** New jobs are submitted paused
- **Flag completed jobs as archived:** If active, jobs are automatically flagged as archived at the end of the render
- **Flag jobs as archived after:** Specifies an interval for the automatic archive feature
- **Prefilter parent pool:** If activated, a folder that specifies a destination pool will filter the hosts available to childs jobs, regardless of the settings of the pool of the childs jobs, that may apply an additional filter
- **Notifications:** Defines which kind of notification you want to be sent to Notificators

The following window shows the logs properties:



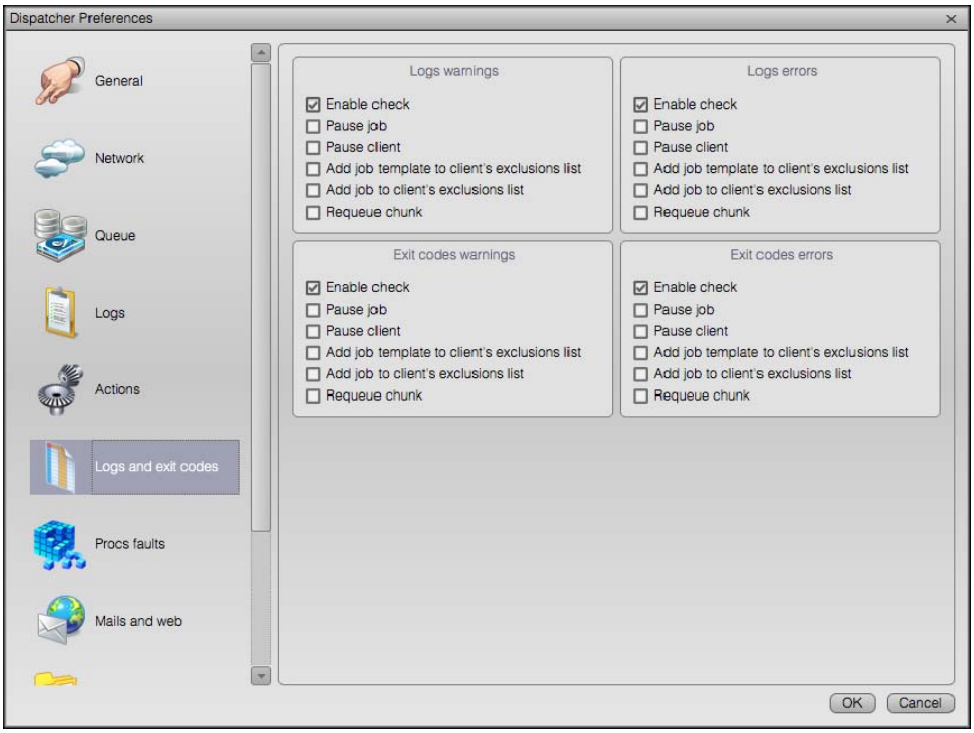
- **Severity level:** Specifies the severity level for the activity logs
- **Destination folder:** Specifies a destination path for the logs
- **Clear logs older than:** Tells the Dispatcher to automatically clear the logs after a certain amount of days
- **Clear older logs when the size exceeds:** Tells the Dispatcher to automatically clear the logs when their cumulative size exceeds a certain amount of Megabytes
- **Enable system events:** Tells Muster to log events related to the Dispatcher activity
- **Enable users events:** Tells Muster to log events related to users actions
- **Enable hosts events:** Tells Muster to log events related to hosts activity
- **Enable hosts failure events:** Tells Muster to log events related to hosts failures
- **Clear the log on:** Automatically clear the logs when it reaches the specified amount of entries
- **Send the log on e-mail at:** Sends a dump of the log on the mail specified in the mail configuration section when it reaches the specified amount of entries
- **Log processes command line:** Adds the command line used to start the process at the top of each log file
- **Log processes return codes:** Adds the process exit code at the bottom of each log file

The following window shows the Actions properties:



- **Job actions failures Pause job:** Tells Muster to pause a job if it encounters a failure during an action execution
- **Job actions failure Lock job:** Tells Muster to lock a job if it encounters a failure during an action execution
- **Chunks actions failures Add job to exclusions list:** Tells Muster to put a job in the client exclusions lists if it fails a chunk's action
- **Chunks actions failures Add template to exclusions list:** Tells Muster to put a job template in the client exclusions lists if it fails a chunk's action
- **Chunks actions failures Requeue chunk:** Tells Muster to requeue a chunk if It fails the pre post chunk action
- **Chunks actions failure Abort processing:** Tells Muster to abort a chunk processing if it fails the pre chunk action
- **Actions timeout in seconds:** Defines a global timeout value for actions
- **Global pre/post job/chunk actions:** Defines an executable to be launched as a pre/post job or chunk action
- **Check return code:** Tells Muster to check the return code of the action
- **Override action timeout:** Overrides the default action timeout

The following window shows the Logs faults properties:

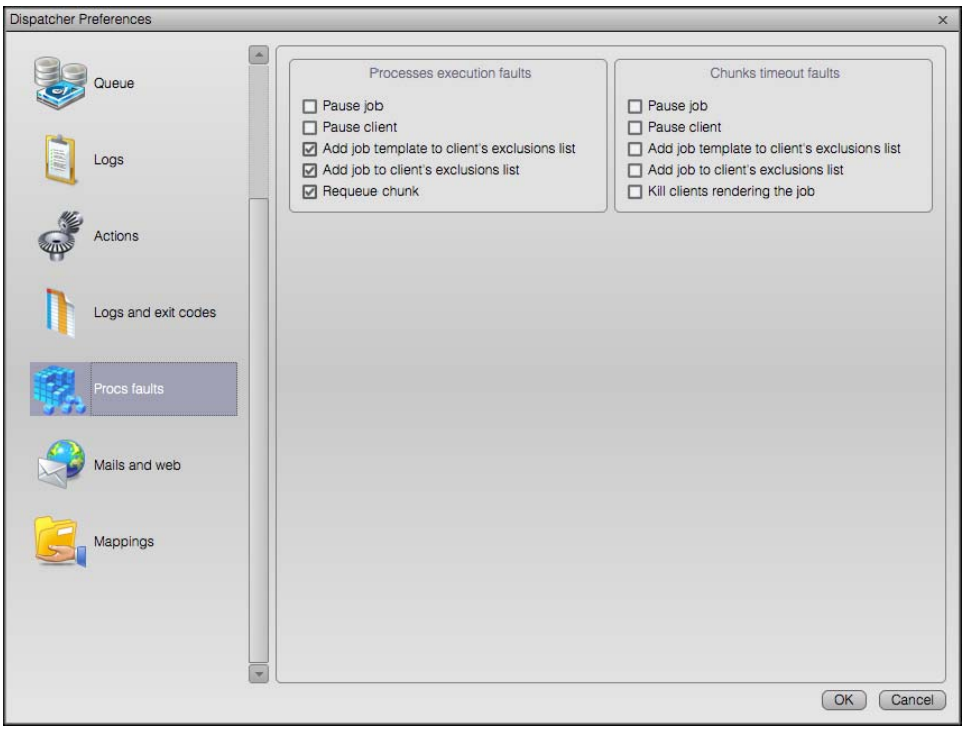


- **Logs warnings enable check:** Tells Muster to enable checking of warnings inside the logs produced by a process. Searched keywords or contents are specified in each template
- **Logs warning Pause job:** Tells Muster to pause a job if it finds a warning in the log
- **Logs warning Pause client:** Tells Muster to pause a client if it finds a warning in the log
- **Logs warning Add job template to client's exclusions list:** Tells Muster to add the job template to the client's exclusions list if it finds a warning in the log
- **Logs warning Add job to client's exclusions list :** Tells Muster to add the job to the client's exclusions list if it finds a warning in the log
- **Logs warning Requeue chunk**: Tells Muster to requeue a chunk if an error is found in its log
- **Logs errors enable check:** Tells Muster to enable checking of errors inside the logs produced by a process. Searched keywords or contents are specified in each template
- **Logs errors Pause job:** Tells Muster to pause a job if it finds an error in the log
- **Logs errors Pause client:** Tells Muster to pause a client if it finds an error in the log
- **Logs errors Add job template to client's exclusions list:** Tells Muster to add the job template to the client's exclusions list if it finds an error in the log

- **Logs errors Add job to client's exclusions list :** Tells Muster to add the job to the client's exclusions list if it finds an error in the log

- **Exit code warnings enable check:** Tells Muster to enable checking of processes warning return codes. Return codes considered warnings are specified in each template
- **Exit code warning Pause job:** Tells Muster to pause a job if it finds a warning in the exit code
- **Exit code warning Pause client:** Tells Muster to pause a client if it finds a warning in the exit code
- **Exit code warning Add job template to client's exclusions list:** Tells Muster to add the job template to the client's exclusions list if it finds a warning in the exit code
- **Exit code warning Add job to client's exclusions list :** Tells Muster to add the job to the client's exclusions list if it finds a warning in the exit code
- **Exit code warning Requeue chunk**: Tells Muster to requeue a chunk if a warning is found in its exit code
- **Exit code errors enable check:** Tells Muster to enable checking of processes error return codes. Return codes considered errors are specified in each template
- **Exit code errors Pause job:** Tells Muster to pause a job if it finds an error in the exit code
- **Exit code errors Pause client:** Tells Muster to pause a client if it finds an error in the exit code
- **Exit code errors Add job template to client's exclusions list:** Tells Muster to add the job template to the client's exclusions list if it finds an error in the exit code
- **Exit code errors Add job to client's exclusions list :** Tells Muster to add the job to the client's exclusions list if it finds an error in the exit code
- **Exit code errors Requeue chunk**: Tells Muster to requeue a chunk if an error is found in its exit code
- **Logs errors Requeue chunk**: Tells Muster to requeue a chunk if an error is found in its log

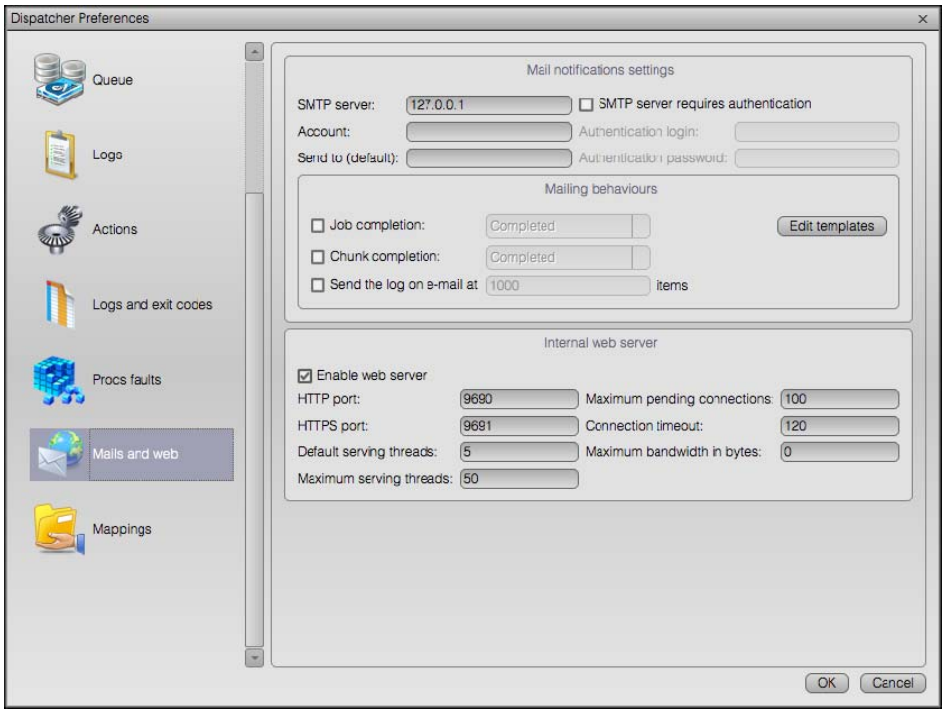
The following window shows the processes execution faults and chunks timeout properties:



- **Process execution faults enable check:** Tells Muster to enable checking of faults during processes executions
- **Process execution faults Pause job:** Tells Muster to pause a job if it reports a fault during processes executions
- **Process execution faults Pause client:** Tells Muster to pause a client if it reports a fault during processes executions
- **Process execution faults Add job template to client's exclusions list:** Tells Muster to add the job template to the client's exclusions list if it reports a fault during processes executions
- **Process execution faults Add job to client's exclusions list:** Tells Muster to add the job to the client's exclusions list if it reports a fault during processes executions
- **Process execution faults Requeue chunk**: Tells Muster to requeue a chunk if it reports a fault during processes executions
- **Chunks timeout faults enable check**: Tells Muster to enable checking of timeouts of chunks
- **Chunks timeout faults Pause client**: Tells Muster to pause a client if a chunk expires
- **Chunks timeout faults Add job template to client's exclusions list**: Tells Muster to add the job template to the client's exclusions list if a chunk expires

- **Chunks timeout faults Add job to client's exclusions list**: Tells Muster to add the job to the client's exclusions list if a chunk expires
- **Chunks timeout faults Kill clients rendering the job:** Tells Muster to kill any rendering activity if a chunk expires

The following window shows the mailing and the web server properties:

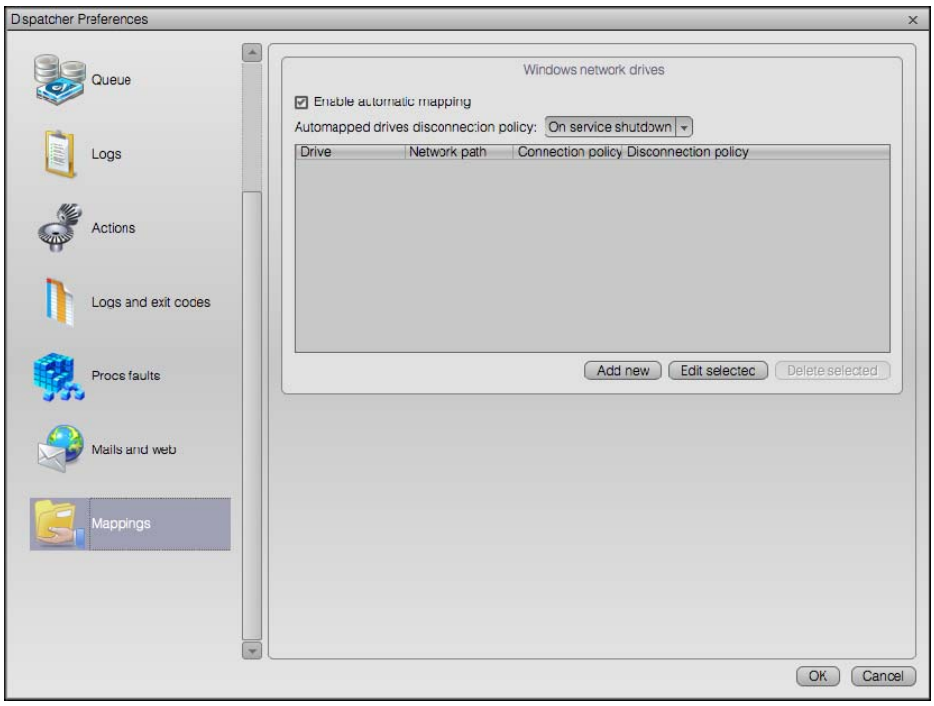


- **SMTP server:** Specifies the SMTP server to use when sending e-mails
- **SMTP server requires authentication:** If your SMTP requires authentication, check this box
- **Account:** The mail account to use when sending e-mails in the form mail@domain
- **Authentication login:** If you need authentication, put your login here
- **Authentication password:** If you need authentication, put your password here
- **Send to (default) :** Default destination address for notifications
- **Mailing behaviors Job completation:** Enables notifications for a particular job event
- **Mailing behaviors Chunk completation:** Enables notifications for a particular chunk event
- **Enable web server:** Enables the integrated web server binding it to the listening ports
- **HTTP port:** Port to listen for incoming HTTP connections

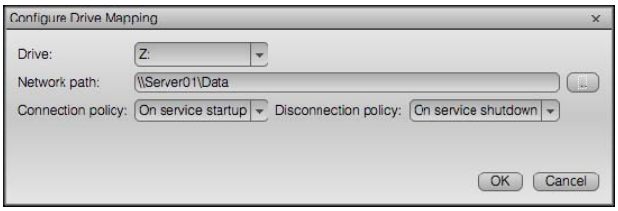
- **HTTPS port:** Port to listen for incoming HTTPS connections (SSL encrypted)
- **Default serving threads:** Number of threads that listen for incoming connection by default
- **Maximum serving threads:** Maximum number of concurrent serving threads allocable by the web server
- **Maximum pending connections:** Maximum number of pending connections in the queue of the web server
- **Connection timeout:** Global timeout for each web server connection
- **Maximum bandwidth in bytes:** Maximum bandwidth allocable by the web server for each connection

By clicking the Edit templates button, you can modify the templates used to generate the mails for each notification:

The following window shows the drive mappings properties:

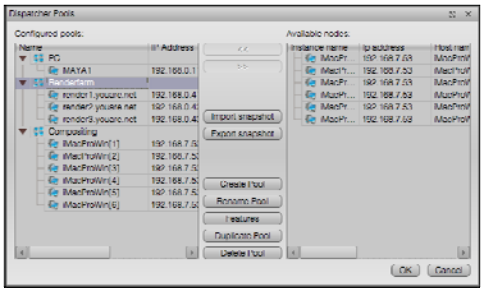


If your Dispatcher is Windows based and you want to statically map some network drives, just configure them in this section by using the drive mapping editing window:



You can tell the Dispatcher to mount or unmount the drives at specific times. I.e. by telling the Dispatcher to unmount the drives after the job completion, you can keep under control the total amount of connection to your file server and limiting the client licenses required.

Render pools



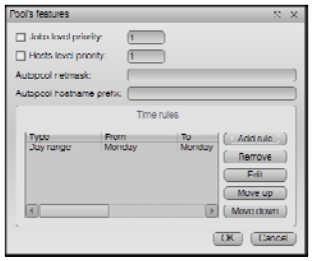
A Render pool is a logical group of instances. By defining render pools, you can tell Muster to use a specific subset of instances/hosts to render a particular job, limit the usage of the resources for a particular user, and wake up some instances when they are effectively required by configuring the wake up on Lan feature.

You can use the Dispatcher Pools dialog to create new pool , duplicate them , and assign available instances to existing pools.

The list on the right shows you the available instances. As soon as you click on a pool in the left view, the instances still not part of that pool will be shown and you'll be able to assign them using the left arrow button.

If you want to remove an instance from a pool just click on it and click the right arrow button. This will remove the instance and put it back in the availability list.

You can configure pools on the fly while the Dispatcher engine is active. Existing jobs will automatically inherit the new settings.



Pools can also have special features like automatic pool assignment based on the host name and / or the IP addresses. You can also set a queue-level priority (to give an additional priority value to the jobs depending on the pool they target) or an instance-level priority (to define an additional priority to instances belonging to a pool).

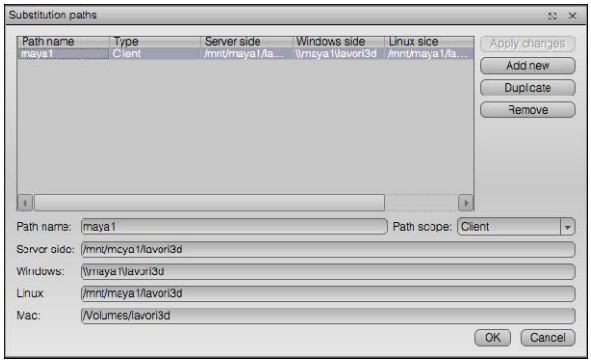
Substitution paths

Substitution paths engine is a fundamental tools for cross-platform rendering. As you probably know, the way the system manages file paths is completely different between Windows and Unix based systems like Linux and Mac OS X.

When you submit a job, you tell Muster where the file for this job is, where is its project and where you want to store the files. You embed this information using the path coding of the platform you're running the Console on.

But what happens when the job is submitted to an instance, or the Dispatcher requires access to the file contents (i.e. Image slicing assembling) ? Muster uses the substitution paths configuration. It exchanges back slashes with forward slashes where required, and transform the paths according with the rules you configure.

This is an example of the Substitution paths dialog:



No matter if you have a longer or shorter path, just put the path prefix to be exchanged and Muster will do the work.

You can also configure a path exchange exclusively for one host (think about an host mounting the shares in a different way or on a different drive) or for one particular user (think about a user connecting from home with a total different mounting scheme).

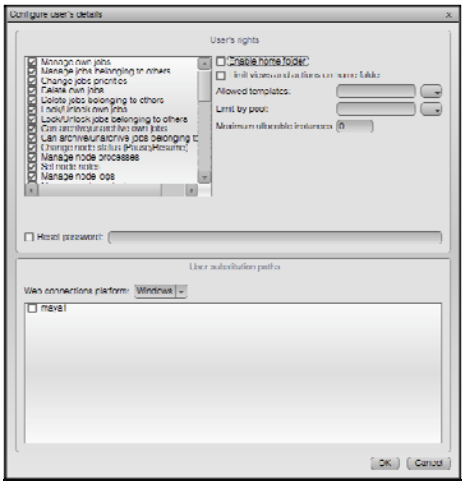
Even paths substitution may be case insensitive (depending on the settings in the Dispatcher preferences), always be sure to specify the paths with their correct case. Linux and Mac OS X are sensitive to cases, so even the exchange of the path may happen correctly, you may still encounter issues if the case doesn't match the target file system.

Users management

Muster has its own users database. The following window is used to manage trusted users:

Once you add an user or you want to configure a new one, click on the **Rights button**.

This is the user rights configuration window:



- **User rights:** This list defines exactly what rights the user has
- **Enable home folder:** Tells Muster to create a folder as an home for the user
- **Limit views and actions on home folder:** Each action of the user is limited inside his home folder. The root of the queue is hidden to him and he cannot see jobs belonging to other users unless an Administrator moves them inside the user's home folder.
- **Allowed templates**: Limits the usage of specific engines
- **Limit by pool:** Limits the usage only to instances belonging to specific pool
- **Maximum allocable instances:** Defines the maximum number of concurrent instances allocable by an user
- **Reset password:** Resets the user's password

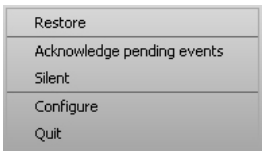
- **Web connections platform:** Defines the platform to use for paths substitution when the user connects to the web server
- **User substitution paths:** Defines user's custom substitution paths

Muster Notificator Reference

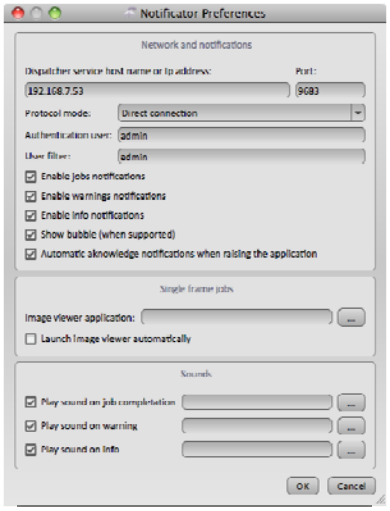
Muster Notificator is a small application that listens to Dispatcher service and pops up reporting several different events. It can even open an image viewer to display single frame renderings once they are available.

Muster Notificator runs in the taskbar or in the upper bar of the Finder on Mac OS X.

By right-clicking the Notificator icon in the task bar, you can access a pop-up menu that lets you configure the Notificator , acknowledge pending event, flag it as silent (no balloons popup) or close it.



The next picture shows the window that appears when clicking the configure menu item:



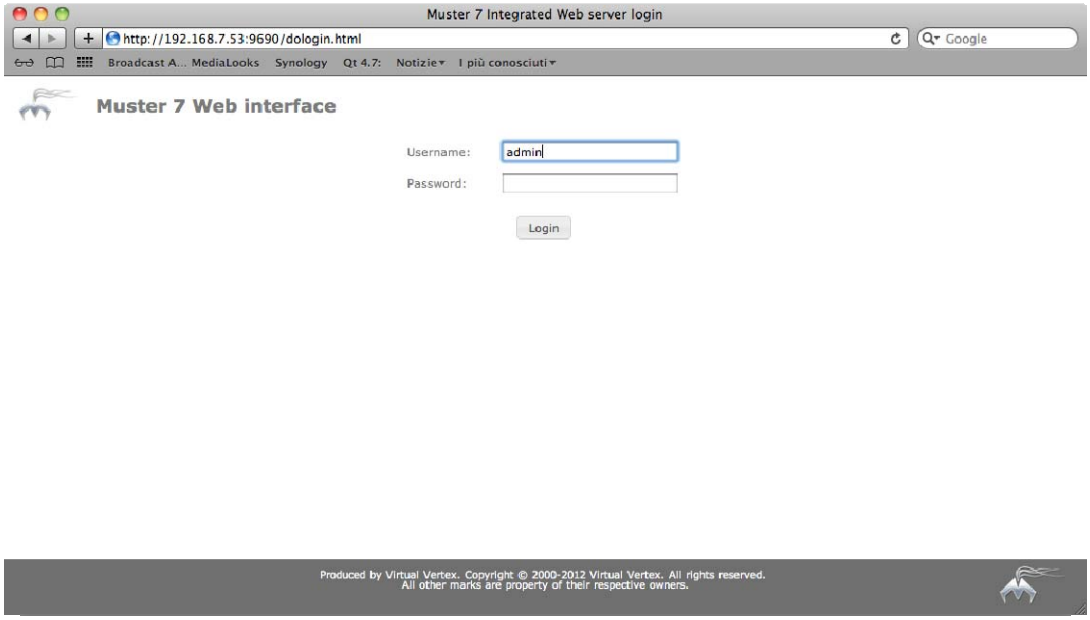
This is an explanation of each parameter:

- **Dispatcher Server:** This is the IP address or name of the Dispatcher service.
- **Port:** This is the port where the Dispatcher listens for notifications connections.
- **Protocol mode:** You can configure the Notificator to directly connect to the Dispatcher service or listen for broadcast messages on a UDP port. If you are going to connect an high number of Notificator clients, using UDP broadcast will reduce your network usage

for messages delivery. Keep in mind however that, UDP broadcast cannot know where the message is delivered, so the substitution path system won't work in UDP mode.

- **Authentication user**: To perform custom substitution paths, an authentication user name in needed
- **User filter:** If you want to discriminate messages on a user basis, you'll need to put your own username in this field.
- **Enable jobs notifications:** Tells the notificator to report jobs notifications
- **Enable warnings notifications:** Tells the notificator to report general warning notifications
- **Enable info notifications:** Tells the notificator to report informative notifications
- **Show bubble:** Shows a bubble in the system tray (where supported) when a new notification is available
- **Automatic acknowledge notifications when raising the application:** Tells the notificator to automatically flag notifications has acknowledged when you raise the application
- **Image viewer application:** Defines an image viewer to view rendered frames
- **Launch image viewer automatically:** Tells the notificator to automatically launch the image viewer when a new single frame has been assembled by the Dispatcher
- **Play sound on job completation:** Plays a sound when a job is completed
- **Play sound on warning:** Plays a sound when a warning event is available
- **Play sound on info:** Plays a sound when an informative event is available

Using the integrated web server

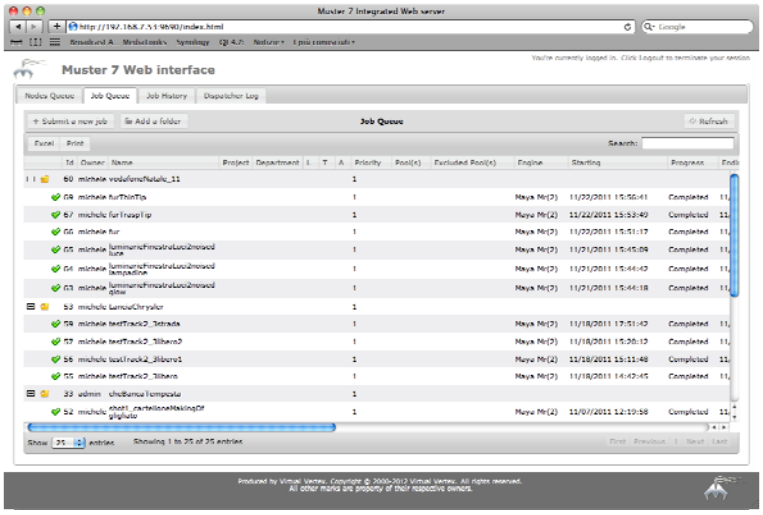


Muster integrates a fully featured internal Web server. You can navigate to the web server interface by opening your browser and then following the links:

http://localhost:9690 (for unsecure connections)

https://localhost:9691 (for secure connections)

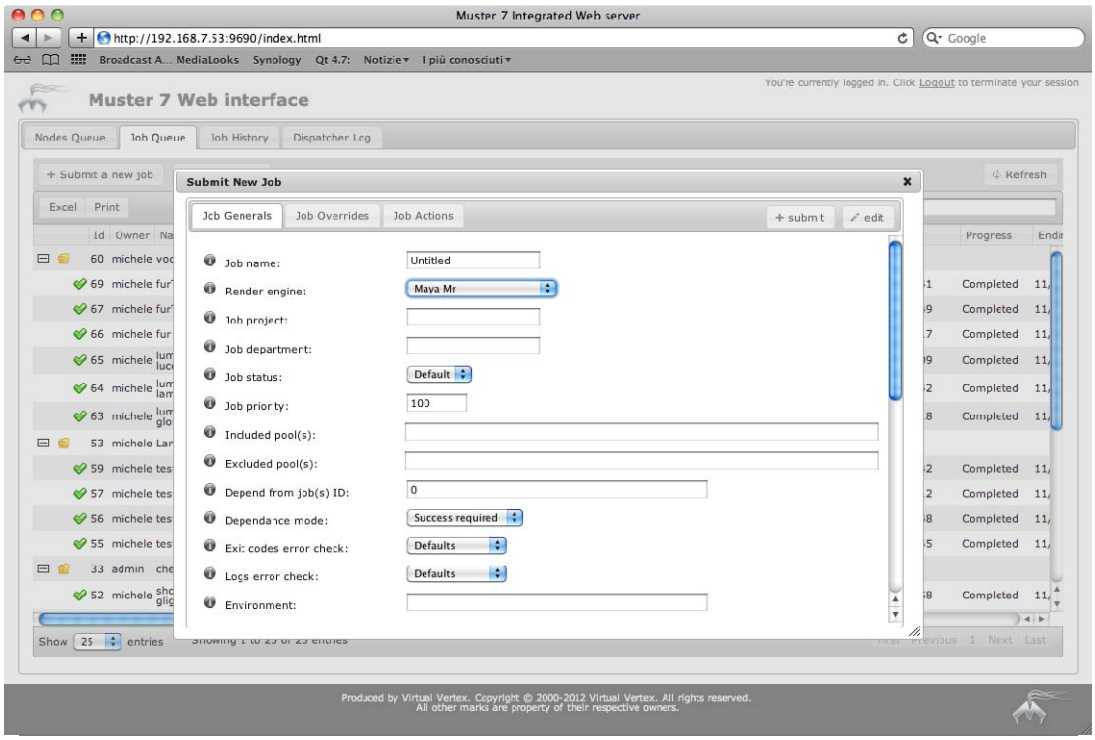
where localhost can be changed with the IP address of your Dispatcher host. After login using your account data, you'll get a similar screen:



The web interface works in a similar fashion to muster explorer. Just right click on the folders/jobs or the hosts to access a popup menu and change job values.

HTTP communications are stateless. This means that you won't get any kind of real time visual feedback like the ones you get using Muster Console. Even some panes are automatically refreshed when managing the jobs, the host queue may require some time to acknowledge the response from the Dispatcher and the host itself. This means you may need to manually refresh the host queue to reflect the changes.

Through the web server, you can even submit or modify jobs with a submission dialog similar to the Explorer one:



Using the web server, you can even query the Dispatcher internal history. You can perform some basic queries and then export the results in an Excel file for further processing.

The web interface of Muster 7 is fully compatible with touch based tablets like the iPad making it the best tool for on site monitoring.

Using Mrtool command line utility

Mrtool is a command line utility used to query and submit jobs to the dispatcher service. It can be used to write custom submission scripts and proprietary plug-ins.It also acts as a general command line interface to the Dispatcher service.

To obtain a list of available commands just type: **Mrtool –h** on the command line.

The Maya connector plug-in uses Mrtool to submit batch job. An in deep analysis of the mel script can be a good starting point for writing custom submission scripts.

The next lines show some examples of Mrtool usage, we strongly suggest you to read the Mrtool section on the reference manual for a full listing of the flags:

Querying the Dispatcher service (on local host, using user admin without any password):

Mrtool –s 127.0.0.1 –q a –u admin

Querying the Dispatcher service jobs:

Mrtool –s 127.0.0.1 –q j –u admin

Pausing a job:

Mrtool –s 127.0.0.1 –u admin –jobp JOBID

Pausing an instance:

Mrtool –s 127.0.0.1-u admin –cpu INSTANCEID

Submitting a Maya job on the queue:

Mrtool –s 127.0.0.1 –u admin –b –e 1 –n Myjob –f "\\Ser\Myprojects\scenes\test.mb*" –proj "\\Ser\Myprojects" -sf 1 -ef 100 –bf 1*

Querying the jobs queue with a custom formatted output with a custom job attribute:

Mrtool –s 127.0.0.1 –u admin –q j –jf id,file,project,MYCUSTOMATTR

Maya Connector plug-in

The Maya connector plug-in is installed automatically by the Muster Installer on windows platforms. For Unix platforms, you need to manually install it by copying the .mel files, the icons and the shelves to the relative Maya directories.

After installing, you must follow those steps:

- Define a new environmental variable named MUSTER that points to the location of the installed Mrtool executable. You can define the variable system-wide of directly inside the maya.env file making it available to Maya only.
- load the shelves, simply execute the Mel command:

```
loadNewShelf "shelf_Muster.mel"
```

- Then source the .mel file:

```
source "MusterConnector.mel"
```

After this step, you can popup the submission dialog clicking on the relative icons in the Muster shelves.

This is a little explanation about the four Connector icons:

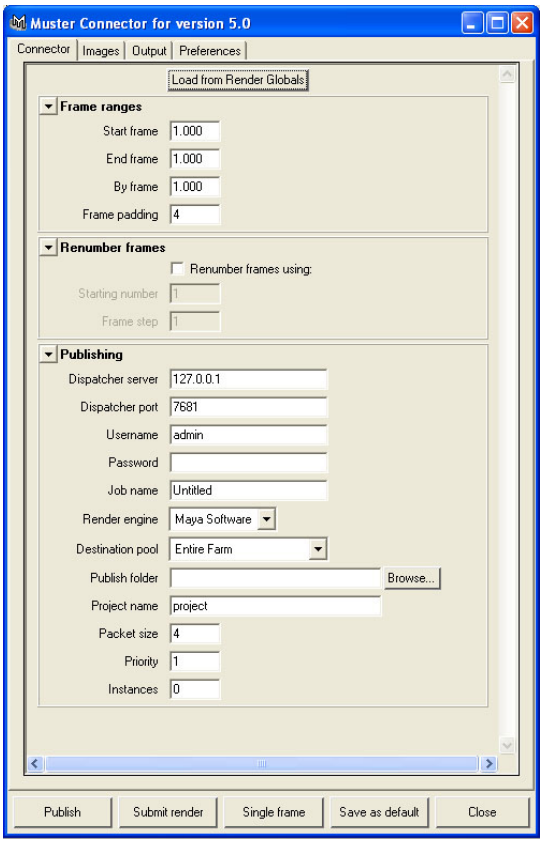
 Starts a single frame render

 Starts a multi-frame render

 Opens the Connector dialog

 Opens the Muster Explorer

By opening the connection dialog, you can configure basic settings as well as perform the basic functions supported by the add on: Texture relocation and batch submission.



The texture relocation feature checks all the textures linked in your current Maya scene (file textures and attached image plane images) and then relocate them in the current project directory. If an image is found outside the regular sourceimage folder, It's copied back to the original position.

After the texture relocation, the scene is saved in the scenes directory of the current project.

When you want to send the scene to your Muster Dispatcher Service, you can click either on Render current frame or on Submit Job. If you send the entire job, the frame range will be taken from the Connector dialog as well as all the other Muster parameters. If you send a single frame, resolution will be taken from the current render global settings.

When sending a job, the connector automatically performs texture relocation and copies all the project structure in the network path specified in the "Network path where you want to publish the project" field. The job is then submitted using the network location path.

General guidelines for distributed rendering

Distributed rendering requires some additional setup to be accomplished successfully. Rules on preparing the jobs change on a software basis and it also depends on the production workflow. By the way, we can provide some general rules to follow, if applicable:

- **Check files external references:** Jobs often requires external files. When you pick up those files during the contents generation, you often pick them up from your local file system. In opposition, a distributed rendering requires contents to be available on the network. You should always pick up the external references from a network path, or consolidate them with your **project** structure, and then move the entire structure on the network

- You should also consider **baking data** performed randomly like Global illumination photons, dynamic caches, particles caching and simila. By linking a cache object, you'll be sure that any host will perform a valid frame that matches with the frames rendered across the farm. Also check the product documentation, there're certain technologies that cannot be rendered on different hosts

- If you experience strange results or general problems when rendering through Muster, we suggest you **to test the batch render directly from the command line**. Keep in mind that batch renders sometimes may have different behaviours than a rendering from the main software interface; you should always refer to the software documentation and check for differences/hints about using the batch rendering

Configuring your network

Distributed renderings relay on the network for any kind of data transmission. That's a good reason to consider your network physical layer as the primary bottleneck in the entire rendering process.

As a network administrator, you can follow several steps to ensure your network is working at its maximum capabilities and modify your settings and topology to increase performances.

This section gives some hints that may apply in a common environment. Every network environment has its own characteristics that should be analyzed according.

- **Priority 1: Your file repository:** As explained in the tutorial, files that are going to be processed by multiple hosts must have visibility over the network in a common way. Building a shared file repository is the first step you've to take into account. There're several ways to accomplish it, you can choose either a NAS or a server depending on your budget limits; just make sure that you do not impose any concurrent connection limit, like it may happens when sharing files on a Windows XP system. The Windows XP system isn't designed to act as a file server and the default Microsoft policy is to limit the concurrent connections to a maximum of 5. This is a common pitfall of new users that are not aware of such kind of limit and get back strange results, process hangings and similar. A good solution could be moving to a Linux system using the Samba service and/or install a Windows server operating system with enough client licenses. Again this strongly depends on your budget and your existent network topology.

- **Priority 2: Connections to the dispatcher server:** The Dispatcher service should be considered as something different than the file repository. Even the service can be installed on the same machine, it performs a different task. The purpose of the Dispatcher is controlling the remote hosts using low bandwidth messages. When applicable, the Dispatcher server should have a privileged connection to the rendering hosts or at least, separate the corporate network traffic. An optimal setup could be made by several routers routing different packets across different network interface cards on both the server and the clients. Having an high number of clients talking with the Dispatcher for both controlling and I/O messages could increase the latency of the transmission itself, wasting precious seconds of unused calculation waiting for incoming commands.

- **Priority 3: Cross-platform:** When using different operating systems, you should take into account the issues deriving from shares mount points. The substitution paths tool in Muster helps you exchanging the paths but a good planning of your shares is something you should take into account in the initial farm design.

Troubleshooting

If you're stuck and not able to render successfully with Muster, please take a look at the following common issues. If this doesn't help, take a look at the Virtual Vertex forum or drop a line to vvsupport@vvertex.com !

- **Error 211 on Maya, Input/output error or scene loading error**: This is the most common issue. Muster as explained, runs as a service. If you didn't assign an user to the service or the user has not enough privileges to access the network shares, your render will stop with a file read error. This may also happen when your file repository has not enough client access licenses (i.e. Windows XP with more than 5 hosts)

- **Objects are black and/or textures are missing**: You've linked your textures in a relative way on your local host. The slaves on the network don't have the same texture on their locate drive and are not able to read the data. Move your textures on the network and/or re link them on the shader

- **Red dot under J or T column**: The job failed. When it encounters a **critical error** Muster prevents further processing and puts the job in the client exclusion list (job based or template based depending on the error). The behavior is normal and avoids an endless loop in the farm. You can fine tune the sensitivity of the system in the Dispatcher preferences. I.e. a warning in the log may be normal and you may not want to lock the render for a warning, so just disable it in the Dispatcher preferences

- **Jobs are not starting**: Your Dispatcher selection engine may be stopped. Check that the blue bar is moving. Check also that the engine you're using is actually supported by the idle hosts (**right click and selected supported templates**), that you're not targeting a pool with busy hosts and that you've not configured limits on the users preferences (templates and pools limits)

- **Jobs are never completed, chunks are always re queued after a certain amount of time**: Muster has a global timeout value for the chunks. If your job is so slow to render you may hit the global timeout and never get it completed. Just tune the global packets timeout in the Dispatcher preferences, or override the timeout in the job's properties

- **Muster complains about a missing executable**: During the setup phase, Muster may fail to detect installed packages, you may need to manually configure each client with the correct paths to the render engines

- **Muster complains about a missing license**: Certain packages requires at least a batch render license. Check that you own such license and you've configured each node in the way described in the software producer manual

System requirements

Muster Dispatcher Service version 7.X
Windows XP sp 2 / SERVER 2003 sp 1 / VISTA / 7 Mac OS X 10.3.9 or greather Linux Kernel 2.4 or greather 1024 MB RAM 200 Free Mbytes on System Disk TCP/IP compatible network card

Muster console version 7.X
Windows XP / SERVER 2003 / VISTA / 7 Mac OS X 10.3.9 or greather Linux Kernel 2.4 or greather 1024 MB RAM 100 Free Mbytes on System Disk TCP/IP compatible network card

Muster Render client for Windows version 6.X
Windows XP / SERVER 2003 / VISTA / 7 Mac OS X 10.3.9 or greather Linux Kernel 2.4 or greather 1024 MB RAM 100 Free Mbytes on System Disk TCP/IP compatible network card

System requirements are related to Muster components only. The render clients must meet the system requirements published by the producers of the batch rendering you're going to use. You'll also need, depending on the licensing scheme, a valid batch render license for each node, unless the software provides batch render licenses for free.