MetaCode, Inc.

ScriptRunner™

Manual

Version 1.6

Table of Contents

1. Overview

ScriptRunner™ is an application built to extend the functionality of Cantemo Portal™. It is developed and owned by MetaCode, Inc., a new venture from the minds at Meta Media Creative Technologies.

ScriptRunner is a highly flexible and powerful tool, designed to subscribe to metadata changes just as Final Cut Server used to. By deploying external scripts, administrators can easily extend the core functionality of Portal and leverage tools they've used with legacy systems. This means you can migrate workflow enhancements along with your catalog data. The goal is to keep executables that fired from Final Cut Server relevant when switching to Portal, but of course ScriptRunner can be used to extend operations beyond a conventional Portal workflow and develop external customization.

To keep things familiar, ScriptRunner also borrows the concept of "triggers" and "responses" from Final Cut Server. To create a trigger, administrators define specific metadata conditions for Portal to test against. Whenever those conditions are met, ScriptRunner responds by running an executable and passing any relevant metadata to it. If required, multiple executables can be associated to a single metadata trigger. Administrators can order and reorder the executables to modify and refine the automation. ScriptRunner is designed to only fire if the previous operation has completed successfully.

For testing, monitoring and troubleshooting, the built-in ScriptRunner log provides administrators with reports for when each executable has been fired and the final state of each operation.

2. Installation

The ScriptRunner installation has been designed to be straightforward and painless. With a few steps you should have it up and running.

Note: Some of the actions require restarting services to complete the installation process.

2.1. System Requirements

ScriptRunner 1.6 requires Portal 1.6+.

2.2. Installing ScriptRunner

Please note to complete the installation you need to have administrator or sudo access to the system.

1. Copy the ScriptRunner-1.6.zip file you received to your Portal server.

2. Unpack the contents of the zip file:

```
$ sudo unzip ScriptRunner-1.6.zip
```

   The output reveals a number of files:

```
Archive:  ScriptRunner-1.6.zip
   creating: ScriptRunner-1.6/
  inflating: ScriptRunner-1.6/install.sh
  inflating: ScriptRunner-1.6/mmctscripts.rpm
   creating: ScriptRunner-1.6/sample_scripts/
  inflating: ScriptRunner-1.6/sample_scripts/md-output.bash
   creating: ScriptRunner-1.6/utilities/
  inflating: ScriptRunner-1.6/utilities/clear_sr_log.bash
  inflating: ScriptRunner-1.6/ScriptRunner-1.6-UserManual.pdf
```

3. Navigate to the ScriptRunner-1.6 directory.

Note: the install script will restart Portal and Vidispine services.

4. Run the install script:

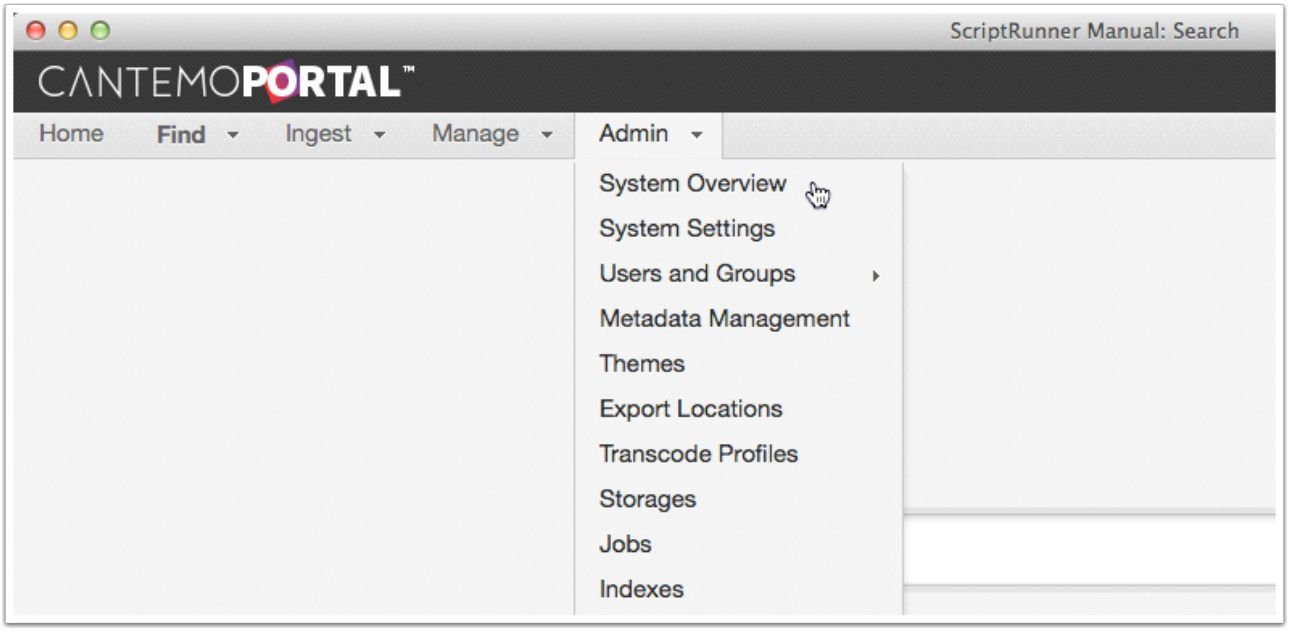
```
$ sudo ./install.sh
```

The installer is quite verbose, which makes it easy to track the progress of the installation and check what is being modified on your system.

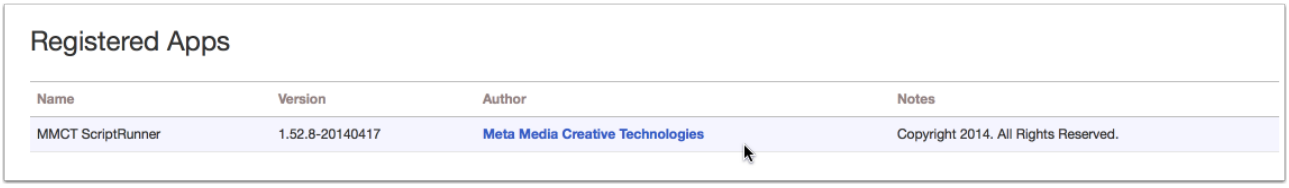
Note: The successful restart of both Portal and memcached completes the install process.

5. Open a web browser and log in to Portal as an administrator user.

6. Click on the Admin menu heading and select the System Overview option.



Once installed correctly ScriptRunner appears listed as a Registered App on the System Overview page.

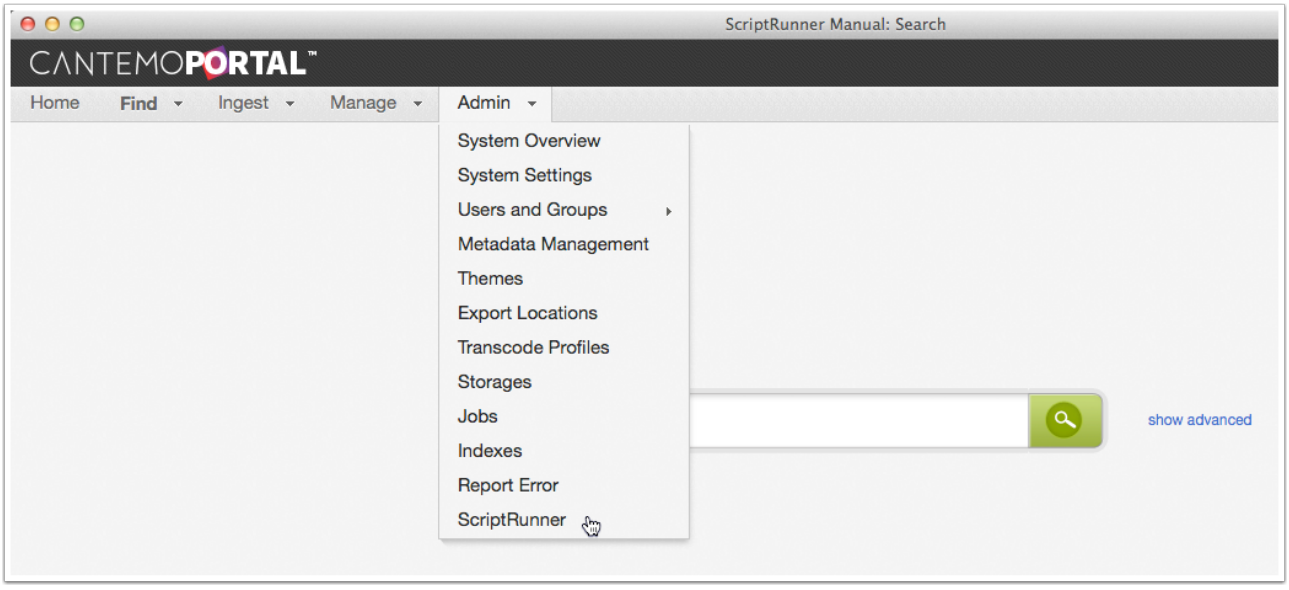


2.3. Installing a License Key

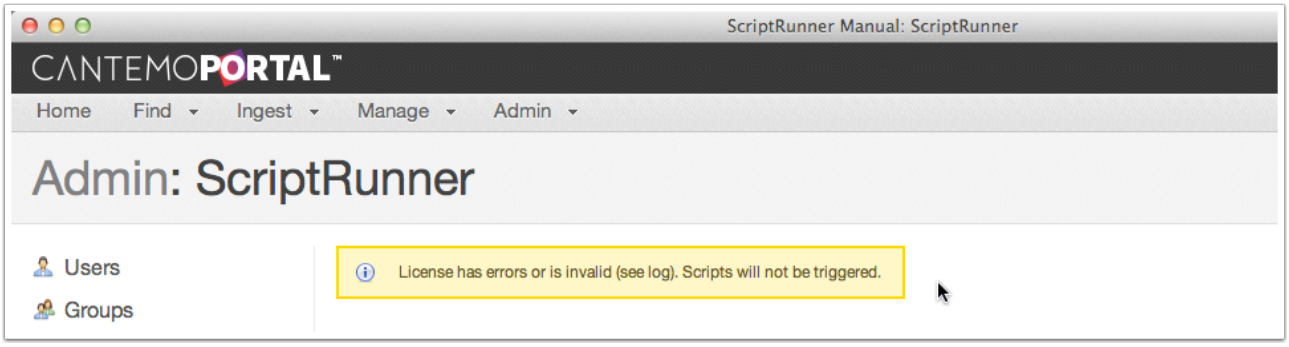
You can install ScriptRunner and create automations without a license key, but ScriptRunner requires a current license for triggers to function.

1. Open a web browser and login in to Portal with an administrator account.

2. Click on the Admin menu and select ScriptRunner from the list of available options.



The ScriptRunner administration page loads in the browser. Unlicensed versions display an error alert below the heading.



3. Copy the license key file, ScriptRunner.key, to /etc/cantemo/portal on your Portal server.

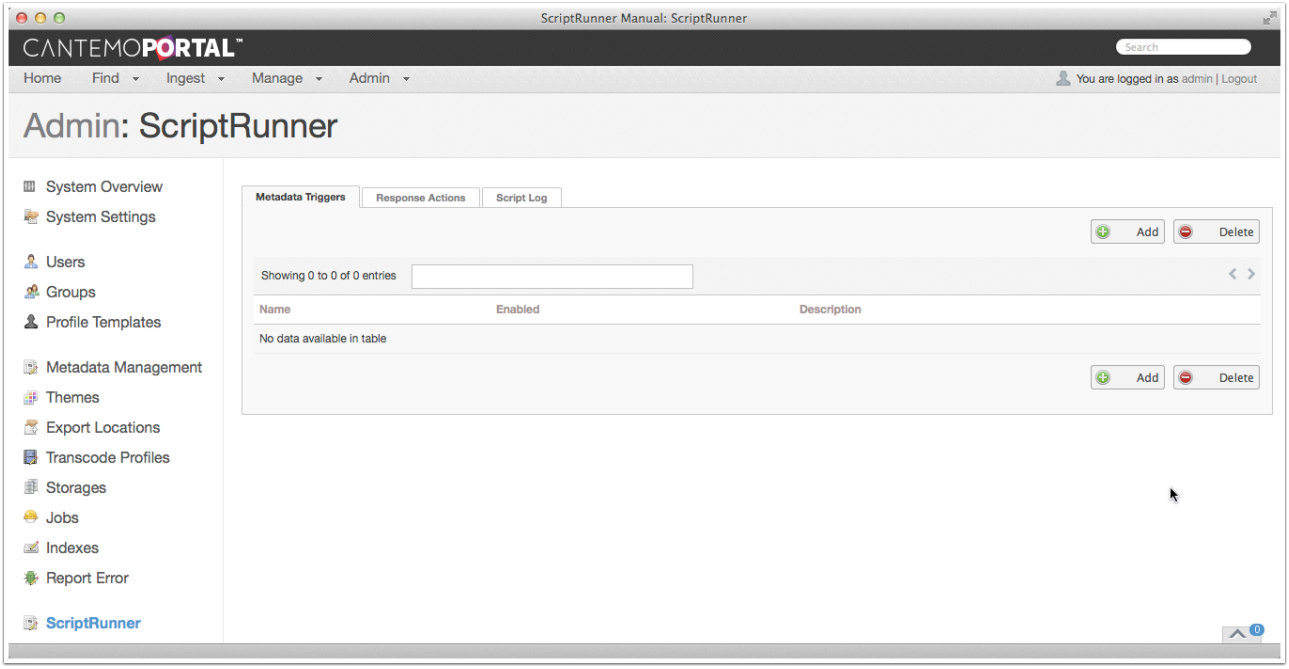
```
$ sudo cp ScriptRunner.key /etc/cantemo/portal/
```

Note: the following command will restart services including Portal.

4. Enter the command:

```
$ sudo supervisorctl restart portal
```

5. Refresh the ScriptRunner Admin page in the web browser.



With the license key correctly installed the warning is no longer be visible.

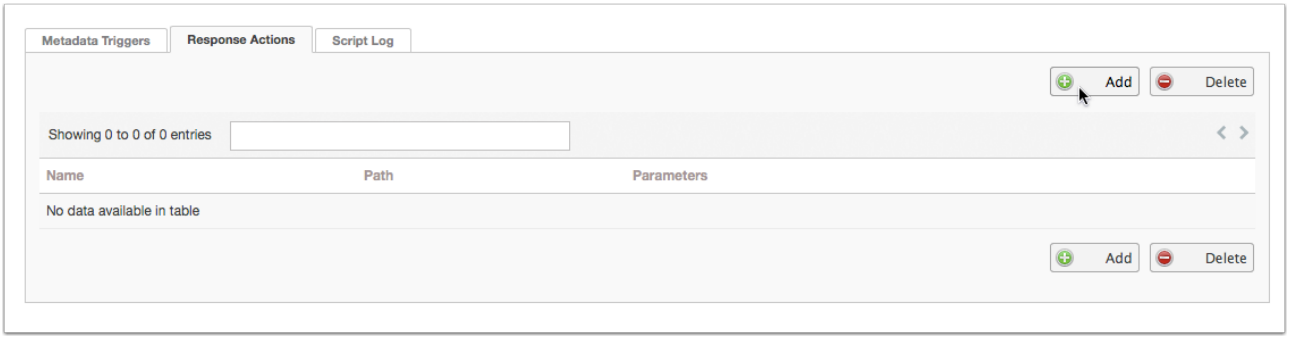
3. Response Actions

The process of creating a ScriptRunner automation begins with defining a response action. Each response consists of a name, a path to the external executable file and the parameters you would like to pass to the script.

3.1. Creating Response Actions

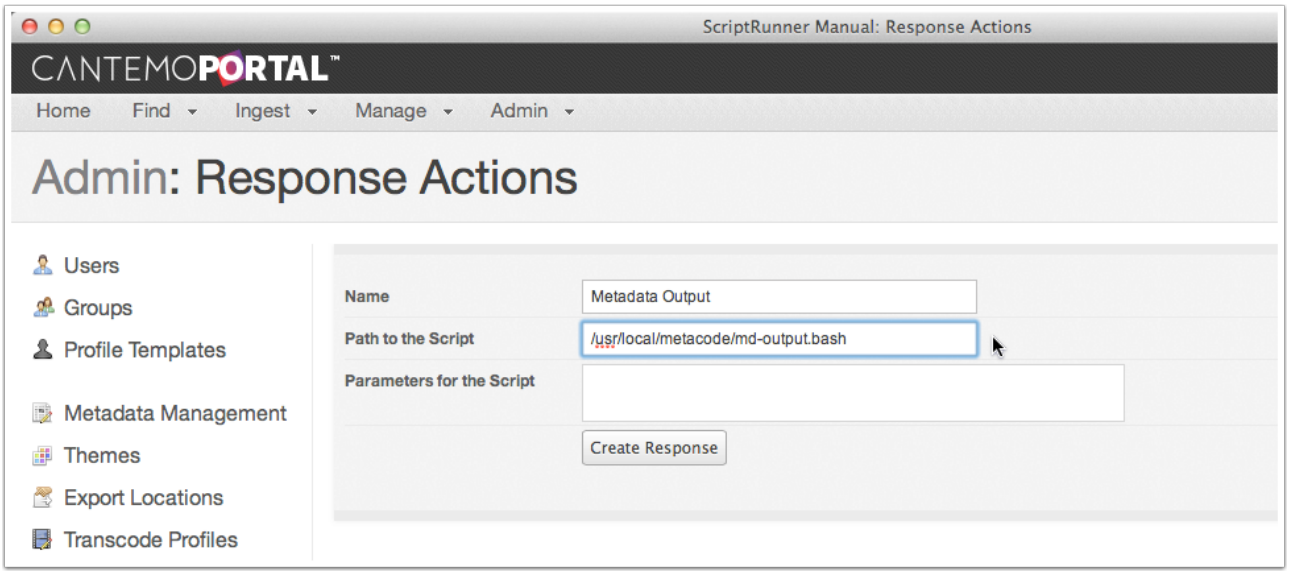
1. Select the Response Actions tab in the ScriptRunner Administration page.

2. Click one of the Add buttons to create a new response.



3. Enter a unique and meaningful name for the new Response.

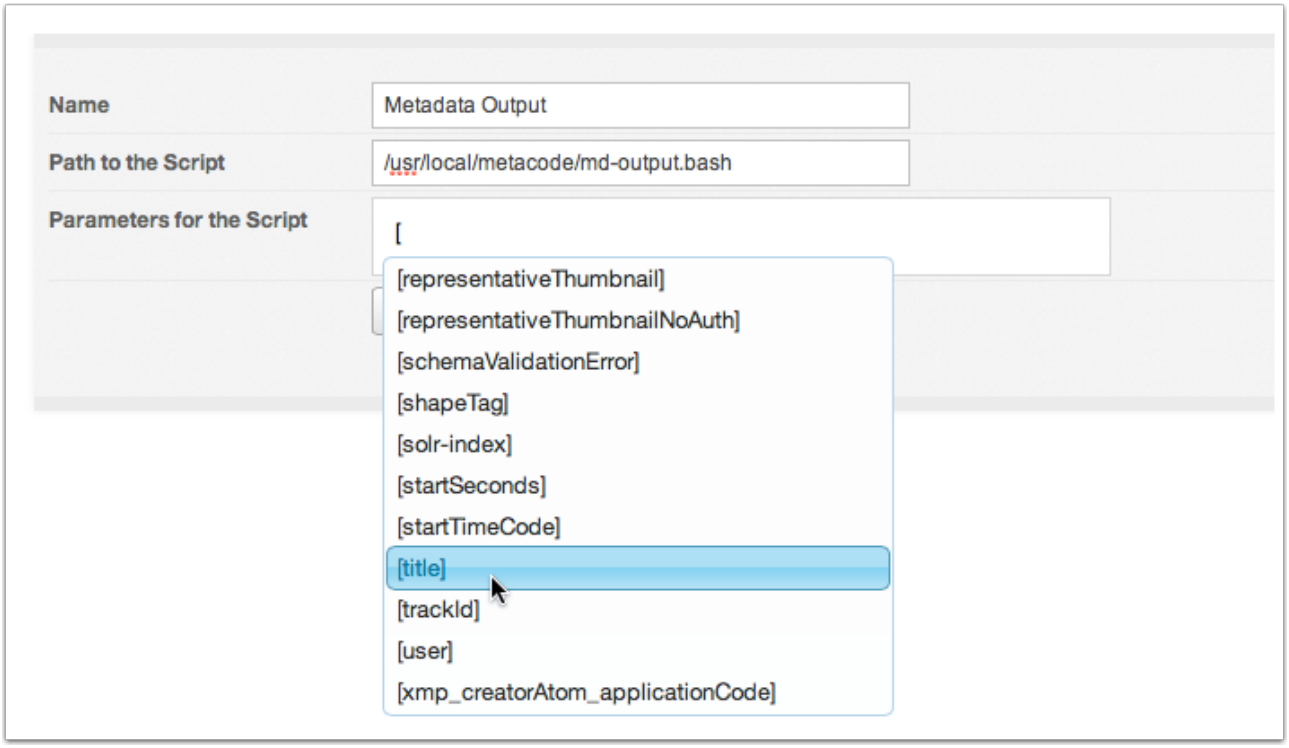
4. Add the absolute path to the external executable file.



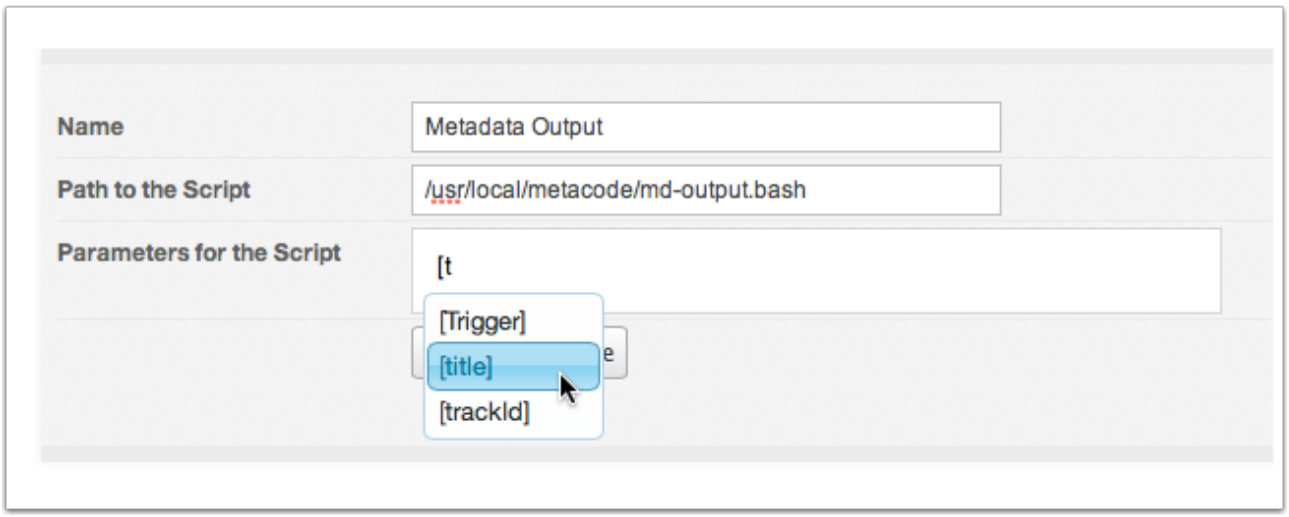
Note: It's important that www-data user has execute permission on the script file and write permissions to any external files or databases that you might modify with your script.

5. Type an open square bracket, “[” in the Parameters field to reveal a list of all of the available metadata fields in available to Portal.

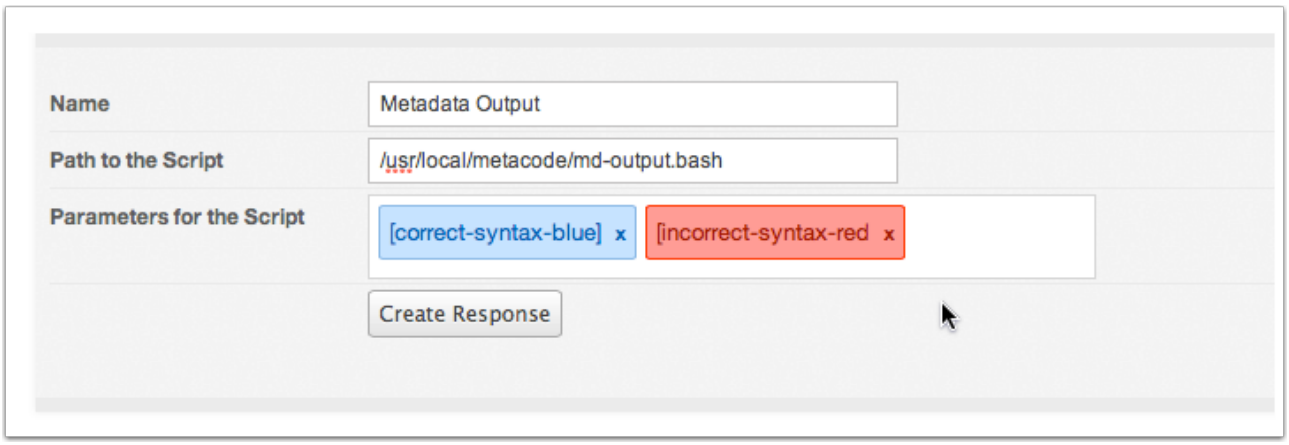
6. Scroll through the list to select the required field.



Alternatively, if you’re dealing with a lot of fields it may be easier to start typing the name of the required field. As you do the list will dynamically filter to display only results that match.



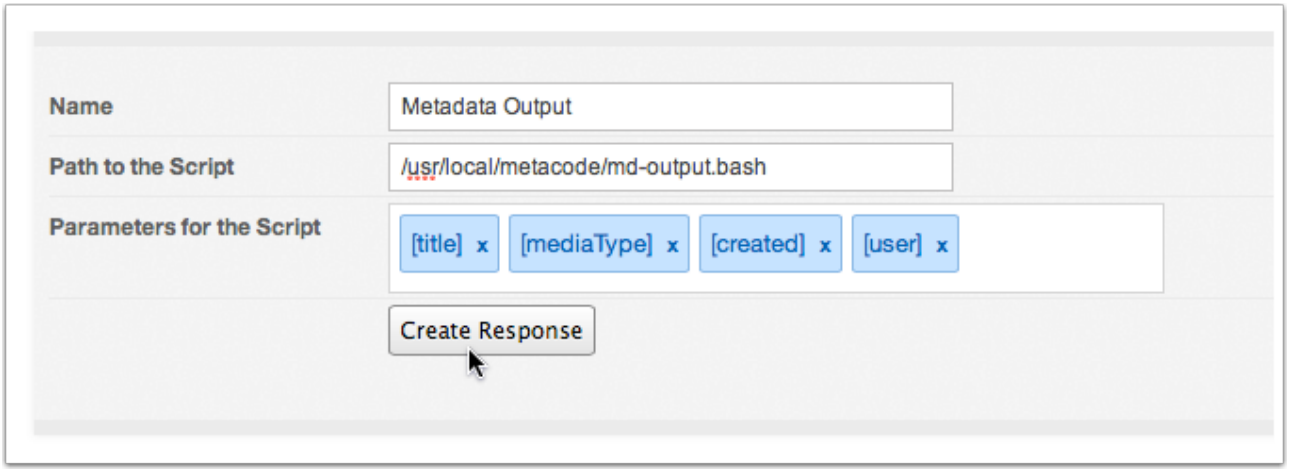
Note: ScriptRunner automatically completes the field name with the correct syntax. If a parameter is entered with incorrect syntax it appears with a red label. If you do make a mistake click on the Delete Parameter button (x) to remove the problem parameter.



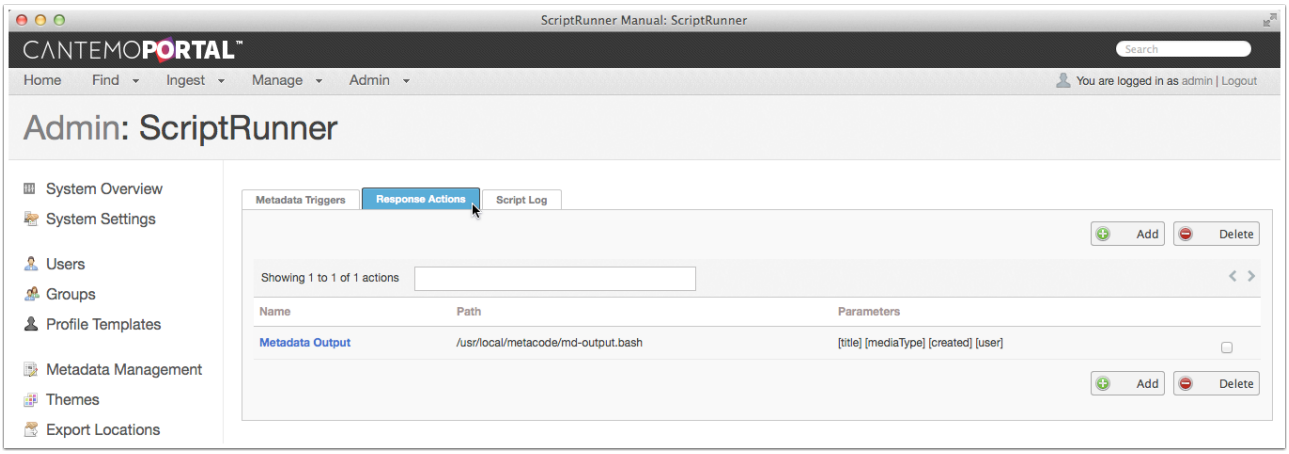
7. Repeat steps 5 and 6 to supply as many additional fields as required.

Note: You need to make sure that the parameters are entered in the order you require them as they cannot be rearranged. To modify the order you will need to delete the original parameters and create new entries.

8. Click the Create Response button to save your new response.



9. Select the Response Actions tab to confirm the new response has been added to the list of available Response Actions.

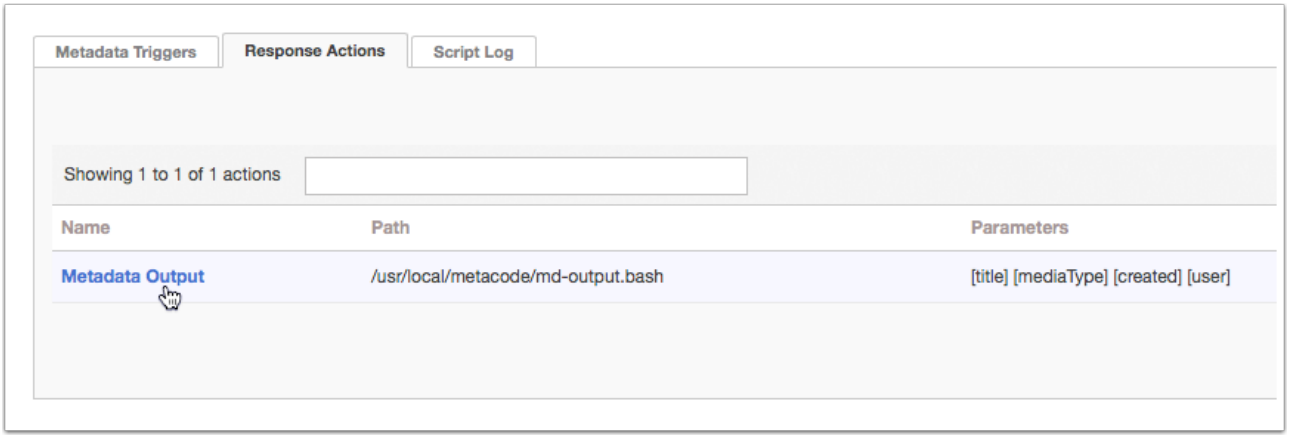


3.2. Editing Response Actions

If you need modify a Response Action you can edit the configuration settings at any time.

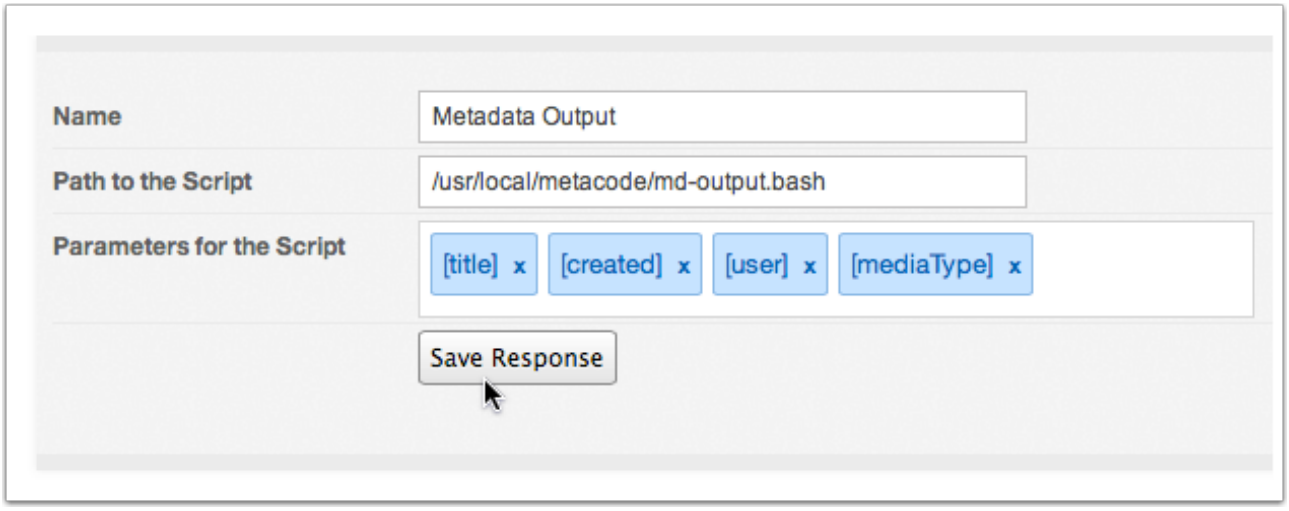
1. Select the Response Actions tab in the ScriptRunner Administration page.

2. Click on the name of the Response Action you wish to edit.



3. Edit the settings as required.

4. Click the Save Response button when you have finished to commit any changes.



3.3. Deleting Response Actions

The process of removing responses has been designed to protect you from creating orphan triggers.

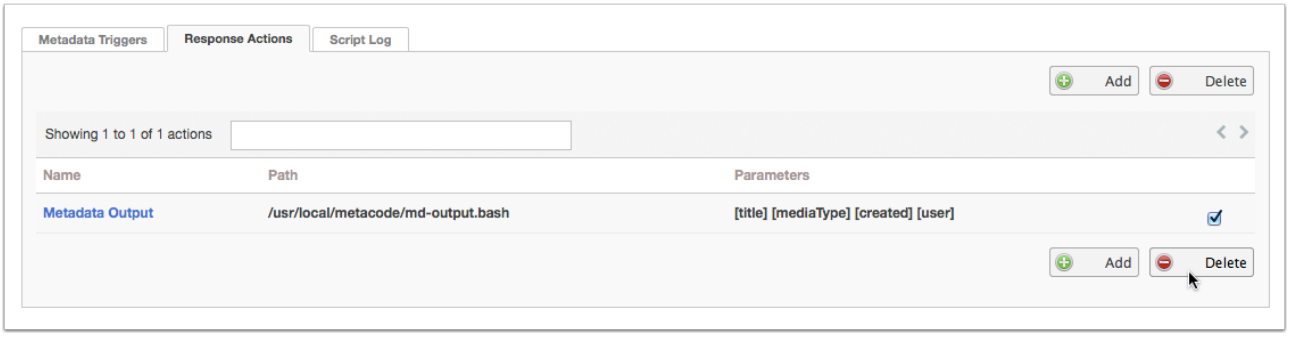
3.3.1. Deleting Independent Responses

1. Select the Response Actions tab in the ScriptRunner Administration page.

2. Click on the selection checkbox of the Response Action you'd like to eliminate.

Note: Selecting multiple checkboxes will enable you to delete several responses at once.

3. Click one of the Delete buttons.



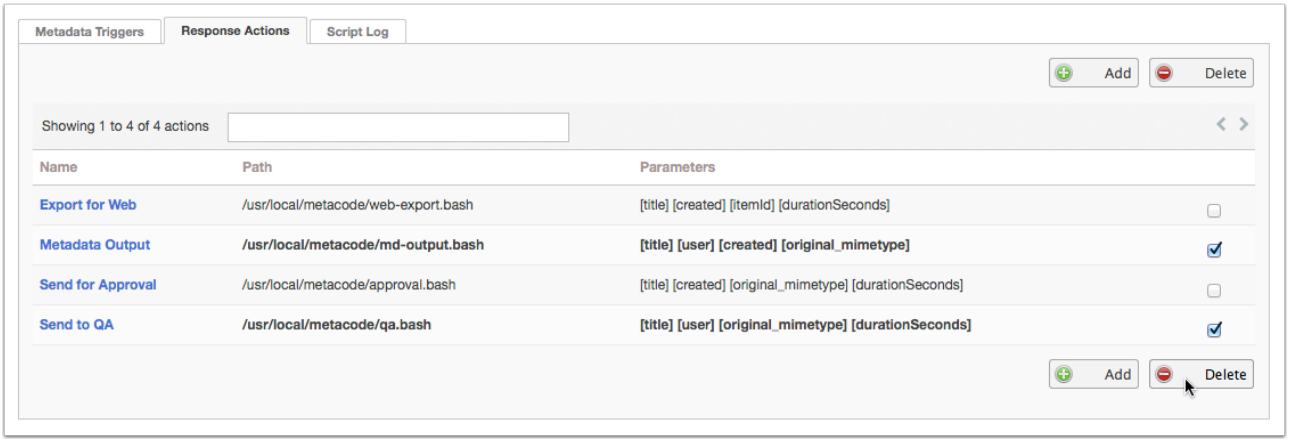
4. Click the OK button in the alert box to confirm that you understand the delete action is permanent.

3.3.2. Deleting Dependent Responses

1. Select the Response Actions tab in the ScriptRunner Administration page.

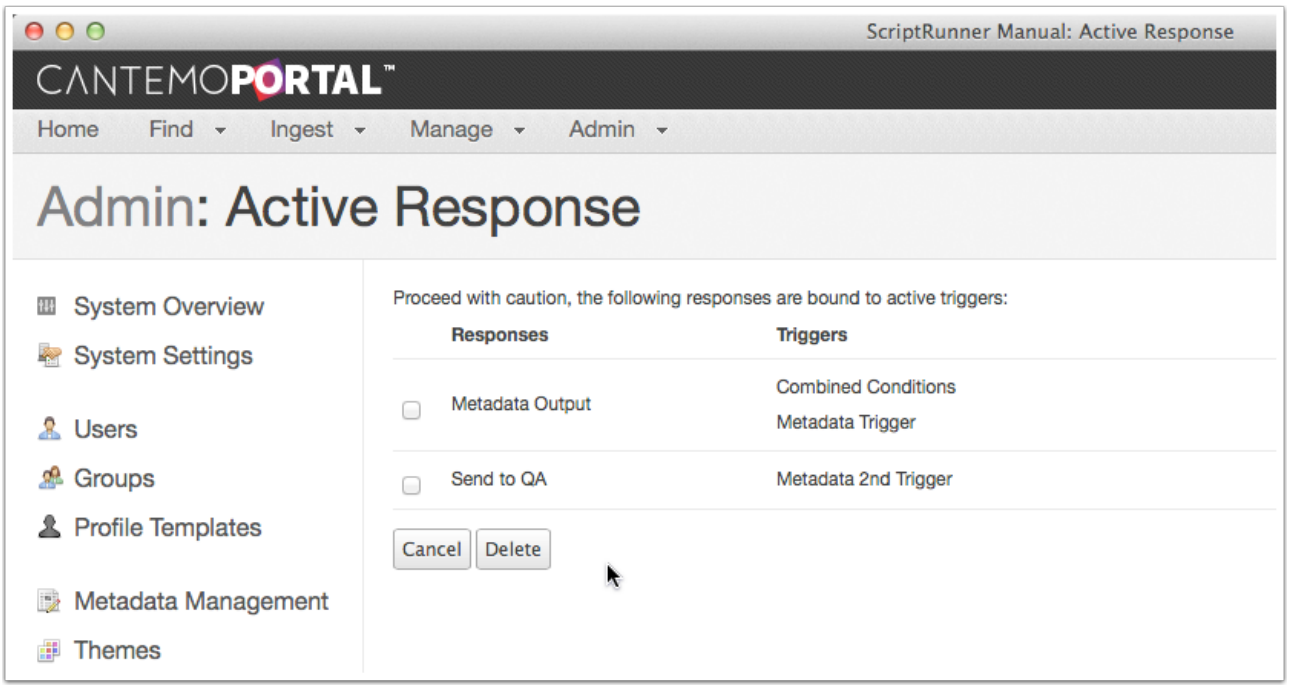
2. Click on the selection checkbox of the Response Action, or actions, you'd like to eliminate.

3. Click the Delete button.



4. Click the OK button in the alert box to confirm that you understand the deletion is permanent.

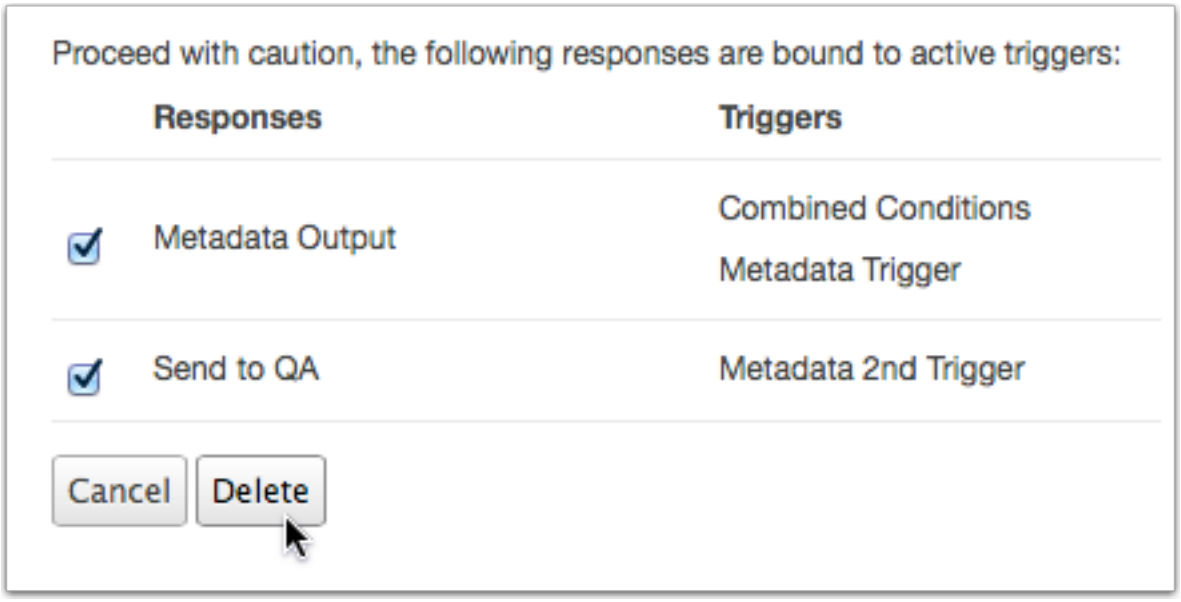
   When the response is connected to an active trigger, Portal loads the Active Response page, which identifies any triggers will be affected by the change and creates the opportunity for you to cancel the operation.



To delete active responses:

5. Select the checkbox of the response, or responses, you want to remove.

6. Click the Delete button.



7. Click the OK button in the alert dialogue to confirm that you understand the deletion is permanent.

8. Select the Response Actions tab in the ScriptRunner Administration page to confirm that the responses have been removed.

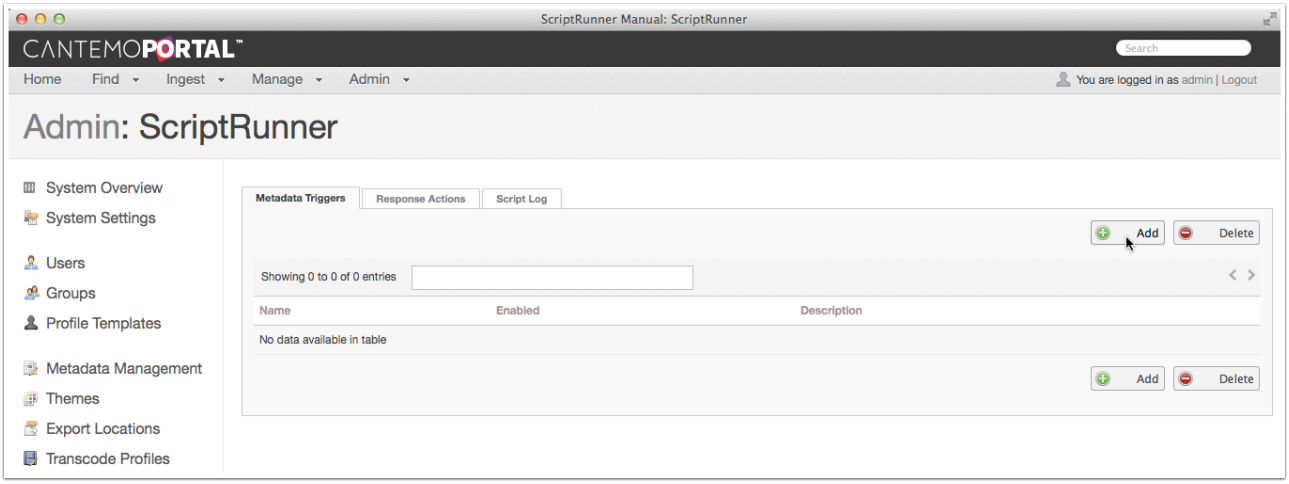
4. Metadata Triggers

Each response action can be invoked by a specific event, or aligned set of conditions. ScriptRunner currently relies on changes to metadata to trigger response actions. With careful planning and clever execution you can introduce some intricately defined automations to your Portal installation.

4.1. Adding a Metadata Trigger

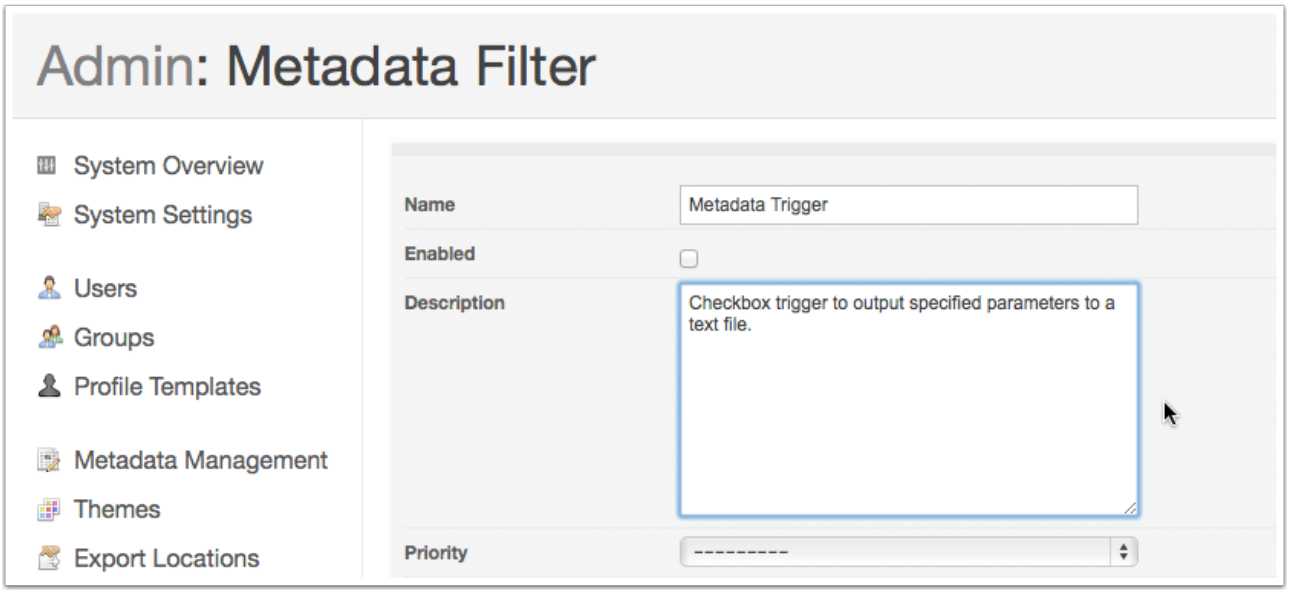
1. Select the Metadata Trigger tab in the ScriptRunner Administration page.

2. Click one of the Add buttons to create a new trigger.

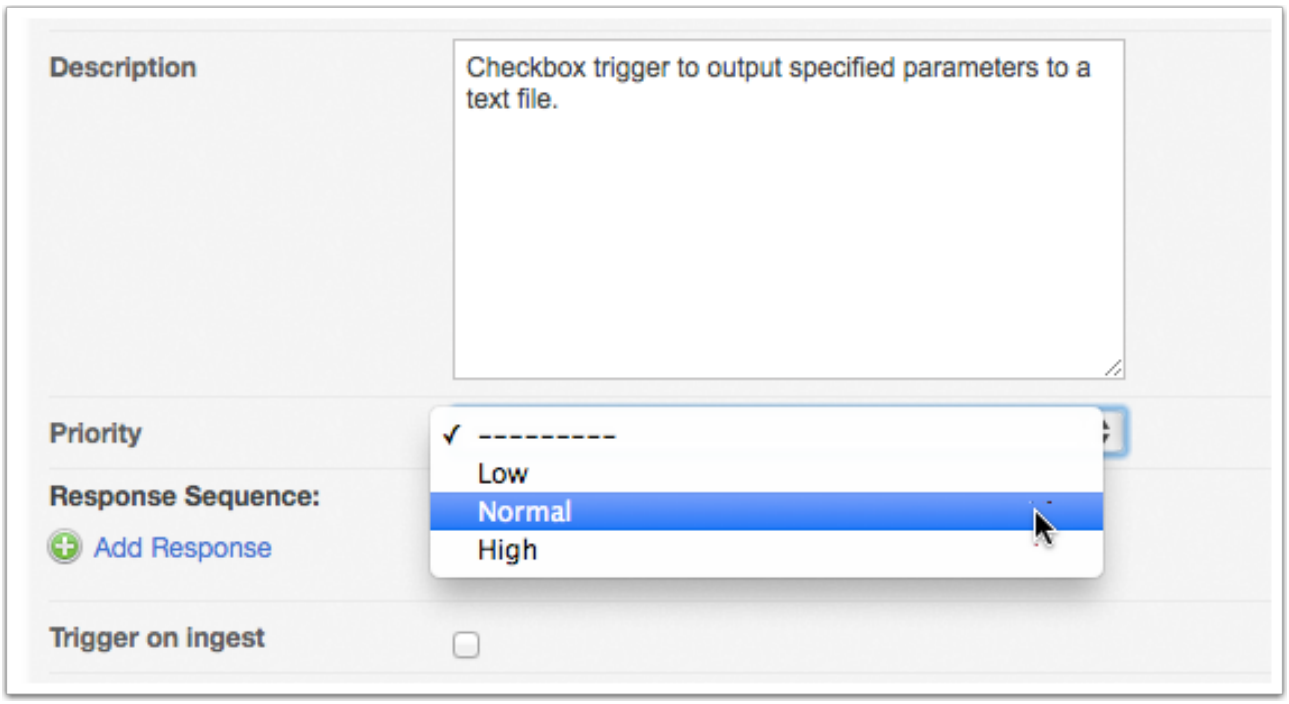


3. Enter a unique name in the Name field.

4. Add a meaningful description that outlines the function of the automation.

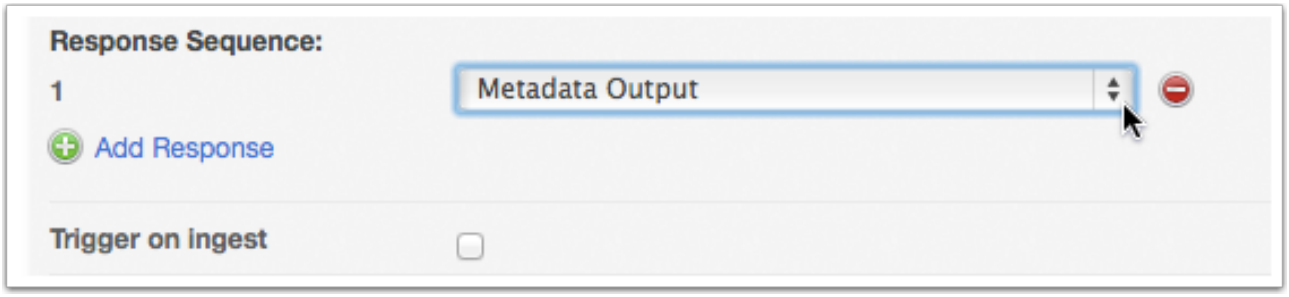


5. Select a Priority status to determine the precedence attributed to the automation.



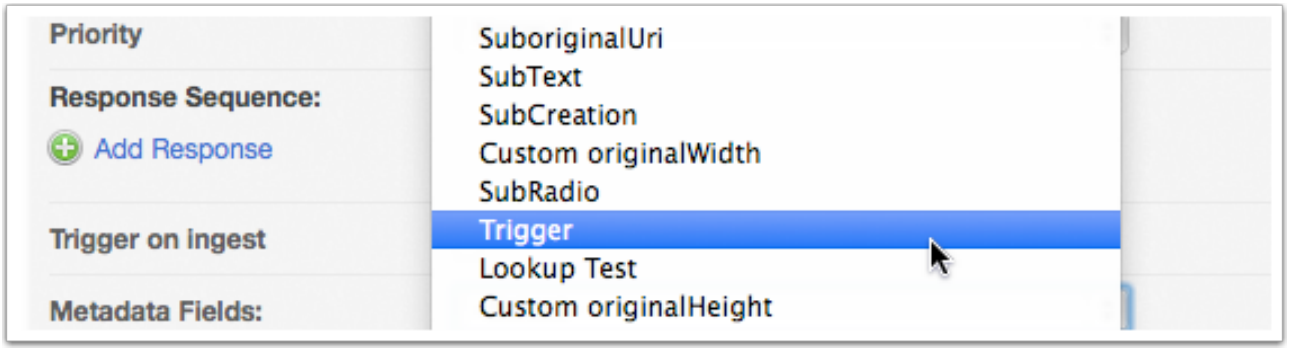
Note: The Priority you set here is limited to ScriptRunner automations. It has no bearing on Portal Job priorities.

6. Click Add Response and choose a response action from the menu.



Note: A Metadata Trigger must have at least one response, but can trigger any number of responses. The first menu is labelled 1 and represents the first response (see 4.3. Triggering Multiple Response Actions).

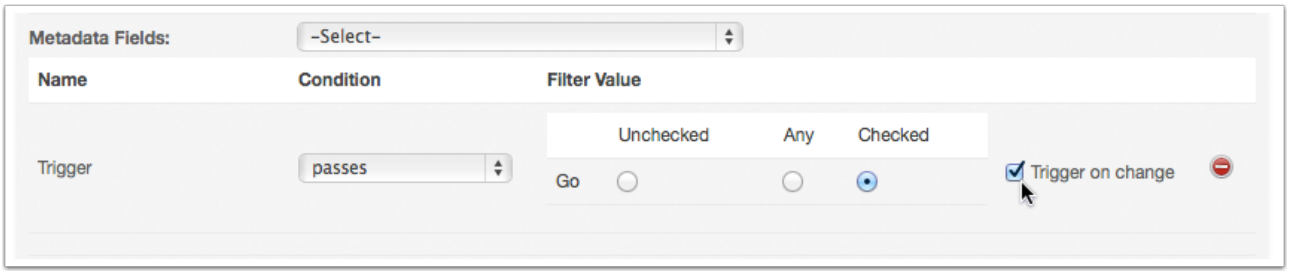
7. Select the relevant Metadata Field from the drop-down menu.



Note: If you've just created a new metadata field and it's not visible in the drop-down, you should restart memcached (see 7.4.3. Metadata Not Visible).

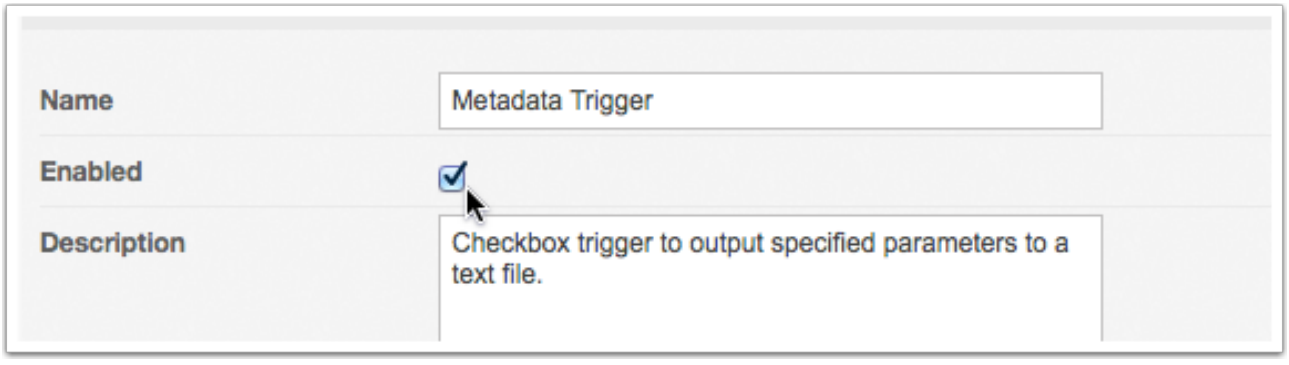
8. Configure the appropriate conditions (see 4.6. Conditions and Filter Values for further details).

9. Select the Trigger on change checkbox for at least one of the conditions (see 4.7. Pivotal Conditions).

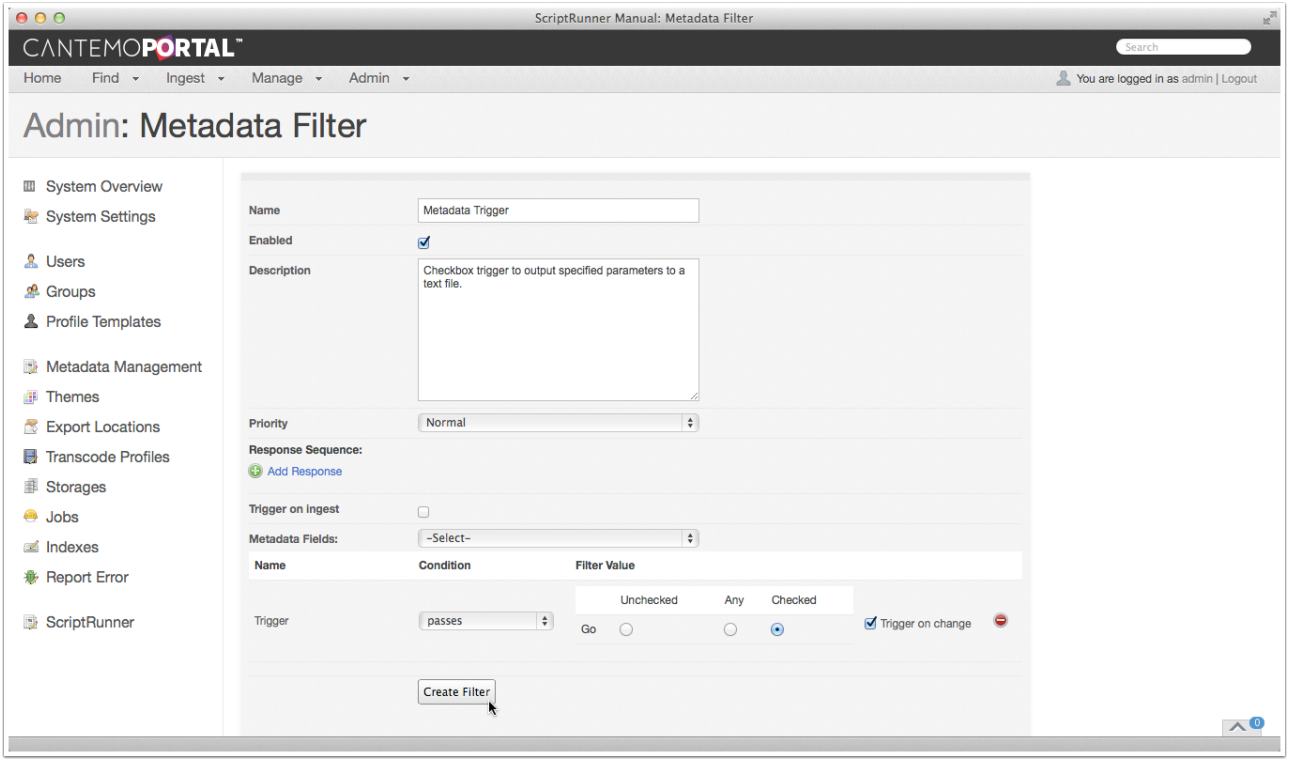


Note: You can select the Metadata Field drop-down again to add more conditions.

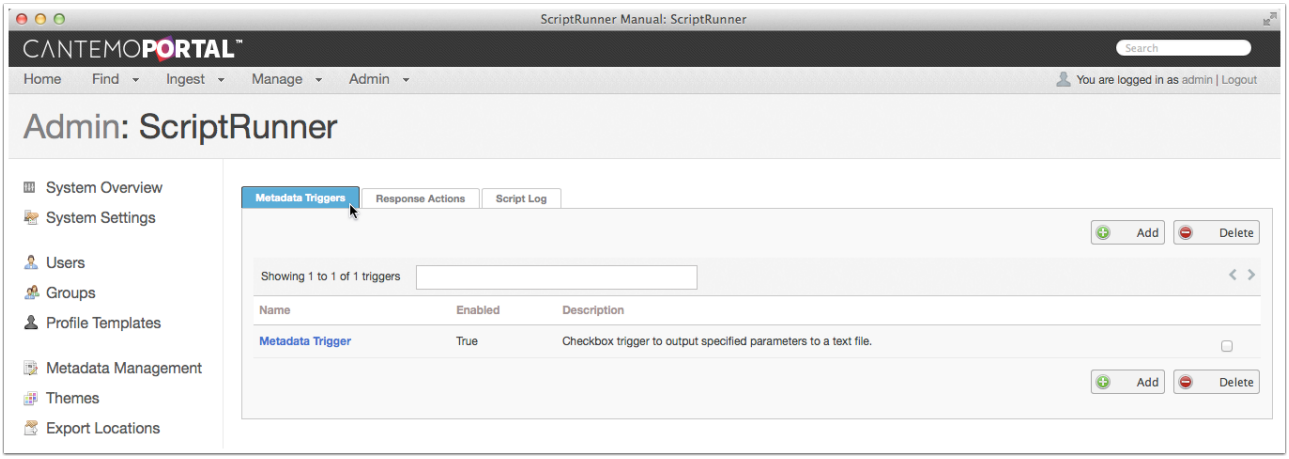
10. Double check your work to ensure the Metadata Filter has been defined correctly and click the Enabled checkbox to ensure the new trigger will be activated.



11. Click the Create Filter button to create the new trigger.



The new filter is listed under the Metadata Triggers tab on the ScriptRunner Administration page.

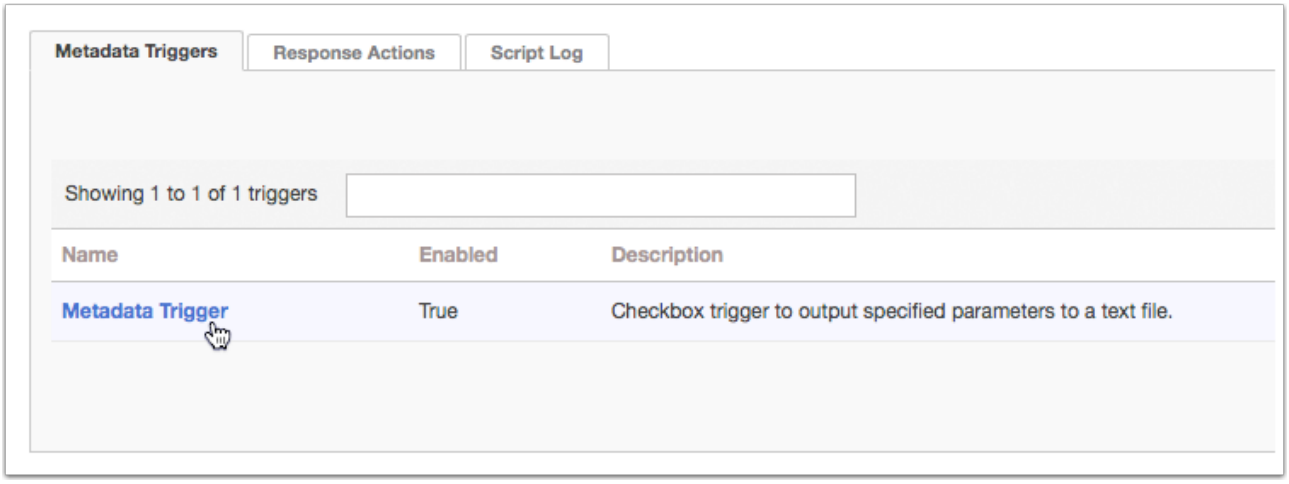


4.2. Editing Metadata Triggers

As with Response Actions, you can modify Metadata Trigger configurations at will.

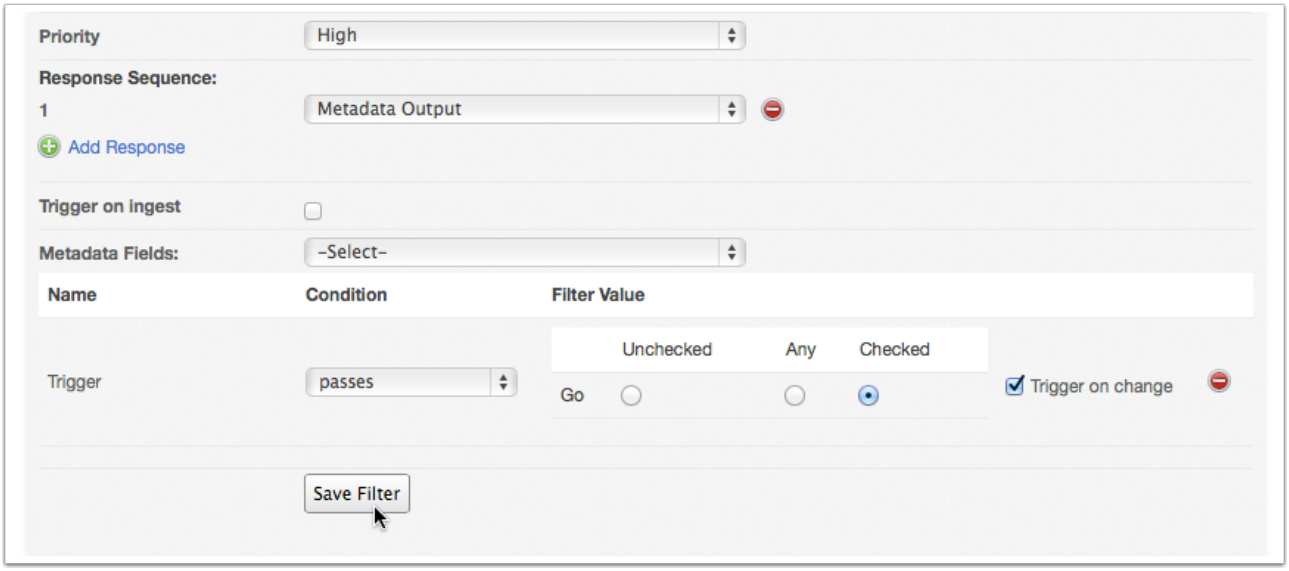
1. Select the Metadata Trigger tab in the ScriptRunner Administration page.

2. Click on the name of the Metadata Trigger you wish to edit.



3. Edit the settings as required.

4. Click the Save Filter button when you have finished modifying the filter to apply your changes.



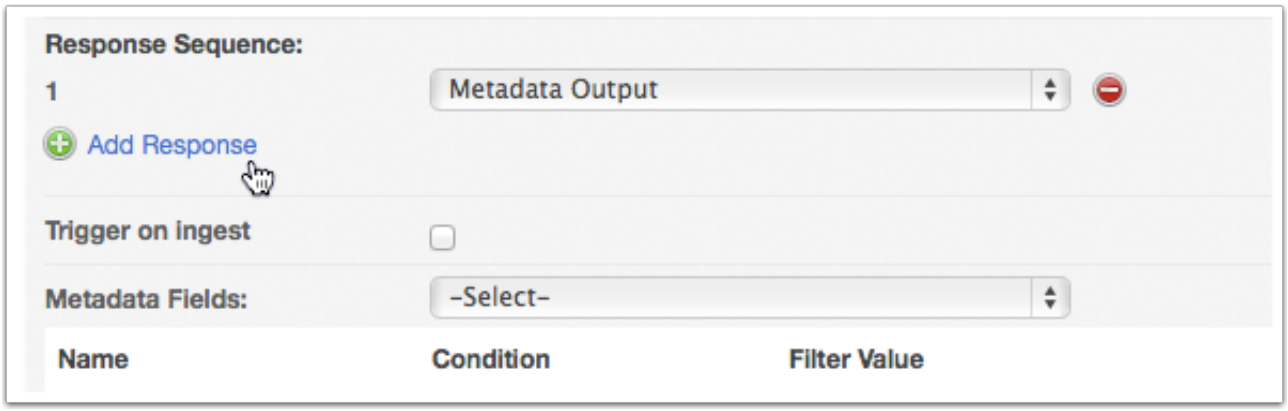
4.3. Triggering Multiple Response Actions

A single metadata trigger can be used to fire multiple responses and create more involved automation sequences.

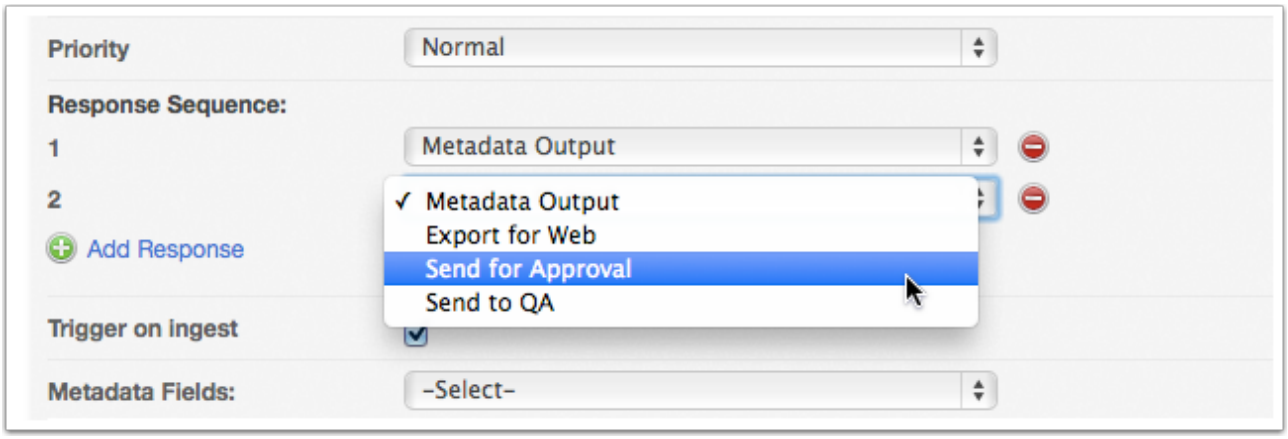
1. Select the Metadata Trigger tab in the ScriptRunner Administration page.

2. Click on the name of the Metadata Trigger you wish to edit.

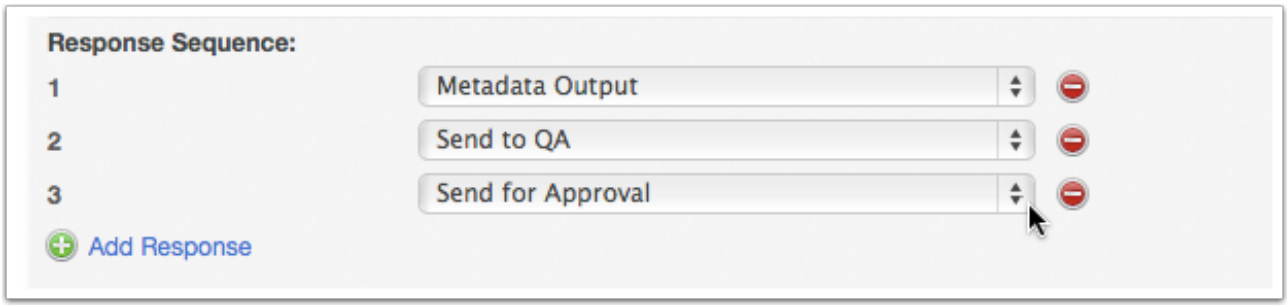
3. Click the Add Response button on the Metadata Filter page to create a secondary response.



4. Select the response you would like to add to the Response Sequence.



5. Repeat steps 1-4 to extend your automation.



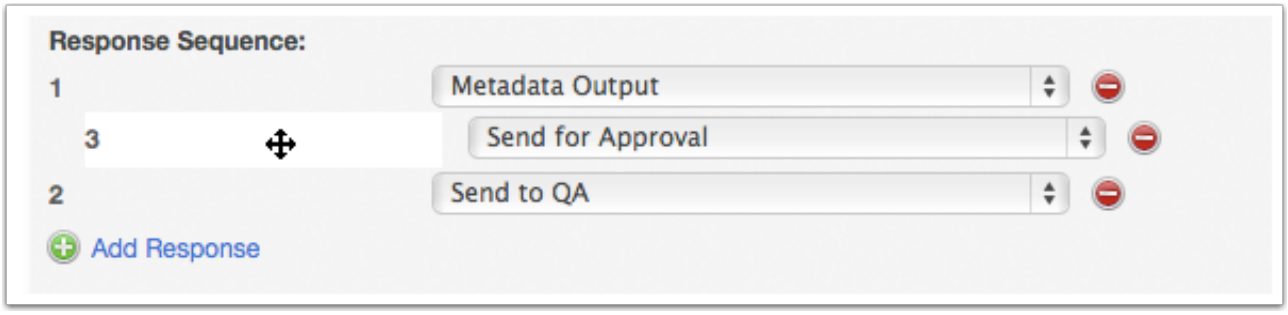
Note: Each response in the sequence is dependent on the last. If an individual script fails to fire, or exits with a code other than 0, then the chain is broken and the rest of scripts are not fired (see 5.1. Browsing Logs for additional details).

6. Click the Save Filter button to save your changes.

4.4. Changing Response Order

As you test and refine your ScriptRunner automations you may decide to switch the order of the Response Sequence. This operation really couldn't be any easier.

1. Click to select the response you'd like to move and drag it to a new position in the sequence order.



2. Click the Save Filter button to preserve your changes.

4.5. Deleting MetaData Triggers

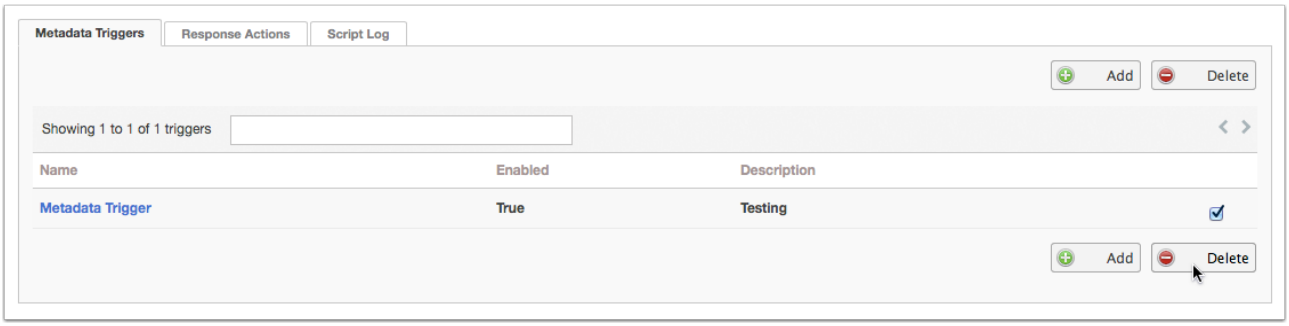
To delete a trigger:

1. Select the Trigger tab in the ScriptRunner Administration page.

2. Choose the selection checkbox of the trigger you'd like to eliminate.

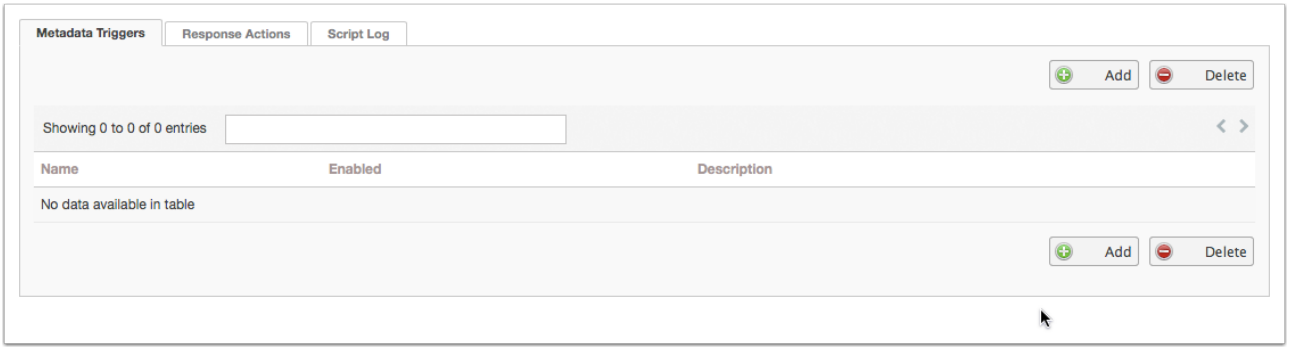
Note: Selecting multiple checkboxes will enable you to delete several triggers at once.

3. Click one of the Delete buttons.



4. Click the OK button in the alert box to confirm that you understand deletion is permanent.

When the page is refreshed the trigger will have been removed from the list.

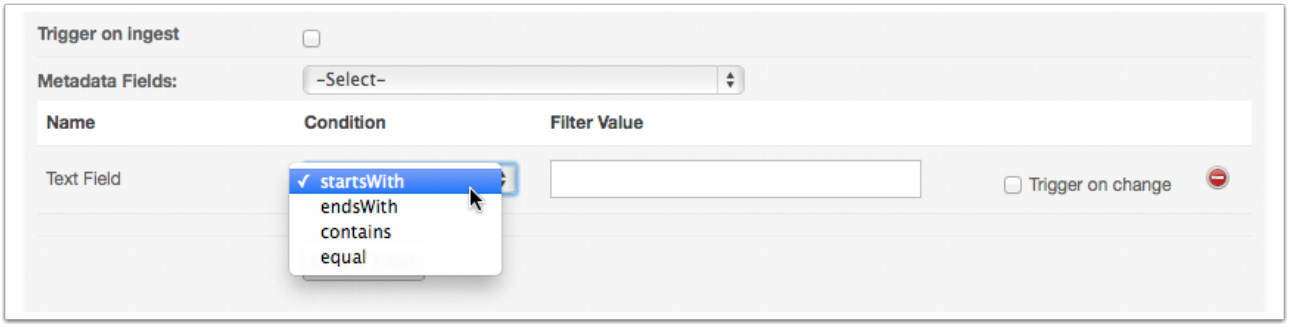


4.6. Conditions and Filter Values

You can test the status of different types metadata field with ScriptRunner. As you might expect, different kinds of field enable you to test against different sorts of conditions.

4.6.1. Text and Textarea Fields

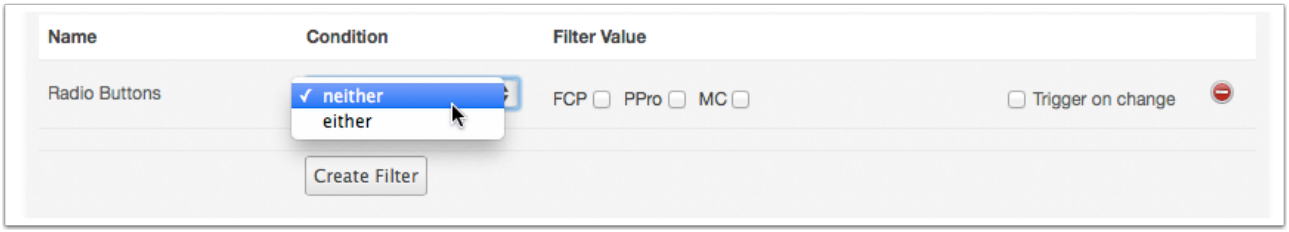
Condition	Status
startsWith	The test string must appear at the head of any text in the field.
endsWith	The test string must appear at the end of any text in the field.
contains	The test string must appear somewhere within field value.
equal	The text in the field must equal the test string exactly.



Note: ScriptRunner test strings are case sensitive and spaces register as characters.

4.6.2. Multi-Choice, Radio, Lookup and Tags Fields

Condition	Status
neither	If any of the conditions or options are not set or deselected by the user the script will fire.
either	If any of the selected conditions or options are set or selected by the user the script will fire.



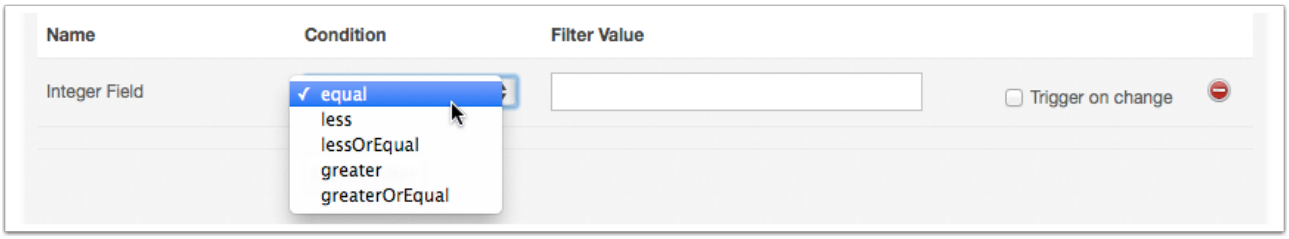
Note: Be aware that choosing a null value in a multi-choice or radio field could trigger a script that uses a neither condition.

You should take special care when using a Lookup field to trigger scripts. While a Lookup field may contain numerous values, Portal regards a single field as a cohesive entity. Any changes to that field will register as a change to ScriptRunner and potentially trigger a script.

New Tag field values cannot be created in ScriptRunner. For Tag fields to function correctly you must test against pre-existing values.

4.6.3. Integer and Float Fields

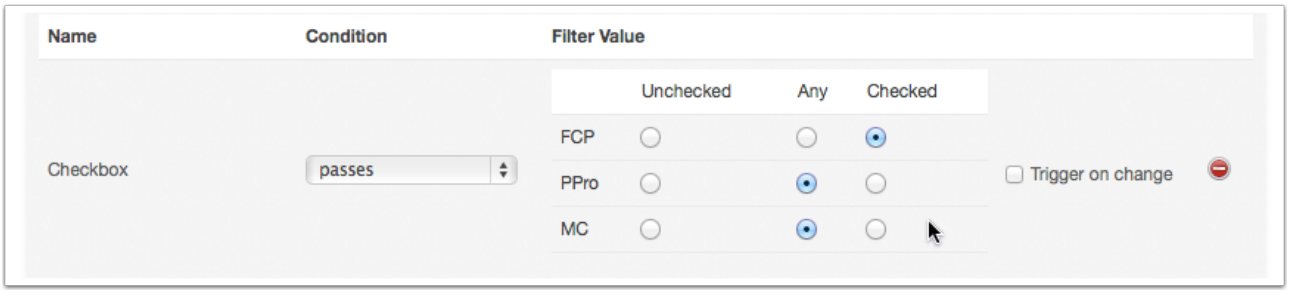
Condition	Status
equal	The test entry must be equal to the specified value.
less	The test entry must be less than the specified value.
lessOrEqual	The test entry must be less than or equal to the specified value.
greater	The test entry must be greater than the specified value.
greaterOrEqual	The test entry must be greater than or equal to the specified value.



Note: Be mindful when testing against integers in a float field as an equal match is exact — 1.0 does not equal 1.

4.6.4. Checkbox Fields

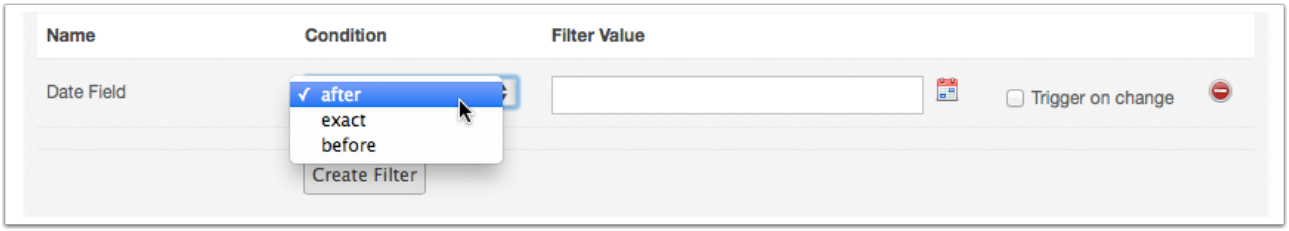
Condition	Status
passes	Checkboxes must be selected as defined for the trigger to fire.



A single checkbox field is able to support an array of individual checkboxes. ScriptRunner is able to isolate changes to each checkbox to trigger individual scripts.

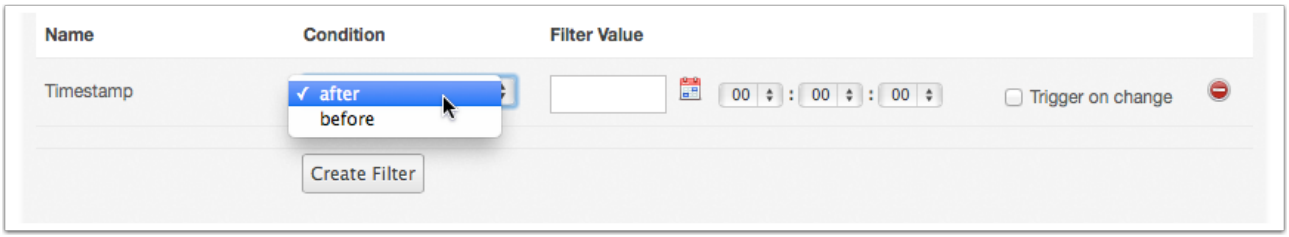
4.6.5. Date Fields

Condition	Status
after	The field must specify a date after the test condition.
exact	The field must specify the same date as the test condition.
before	The field must specify before after the test condition.



4.6.6. Timestamp Fields

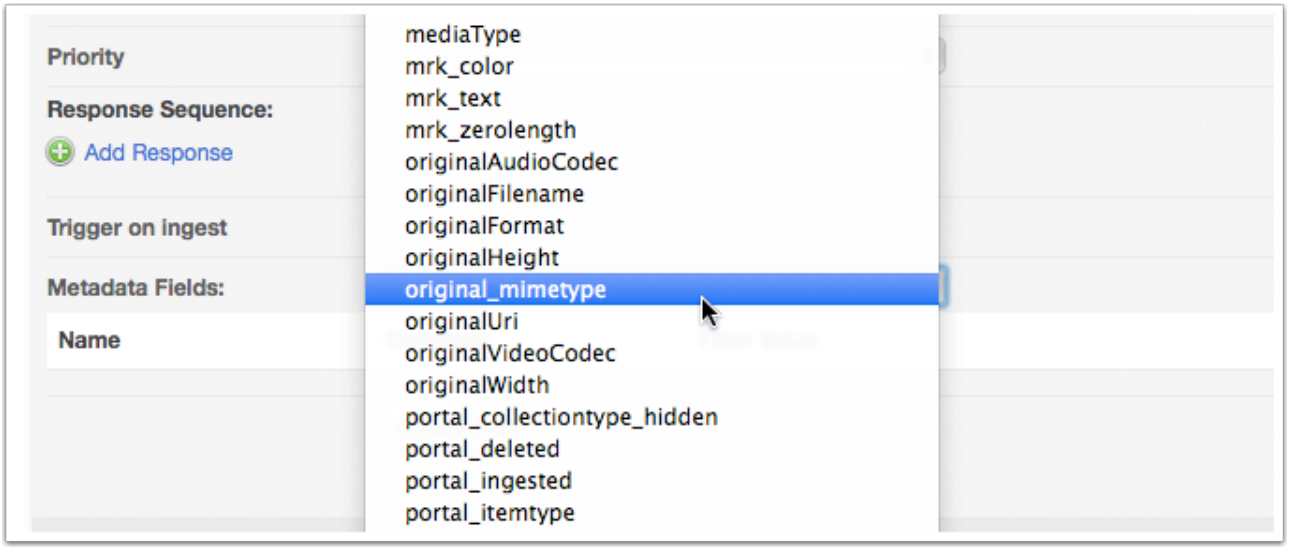
Condition	Status
after	The field must specify a time greater than 59 seconds after the test condition to fire (see 9.2.3. Testing Against Timestamp Values).
before	The field must specify a time before the test condition to fire.



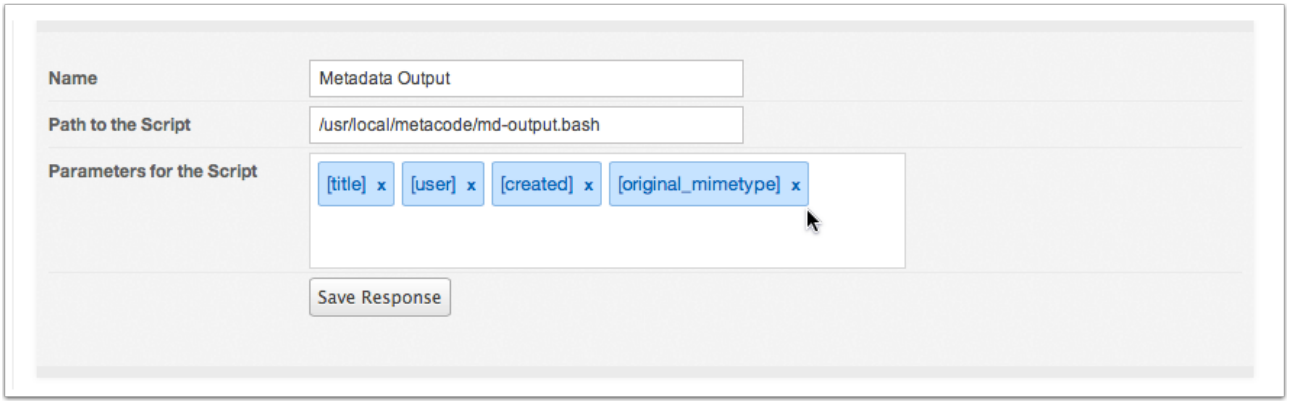
4.6.7. System Fields

There are essentially two distinct contexts where you encounter system fields in Portal. ScriptRunner has a different relationship with each and you should be mindful about how you can work with them as you plan your automations.

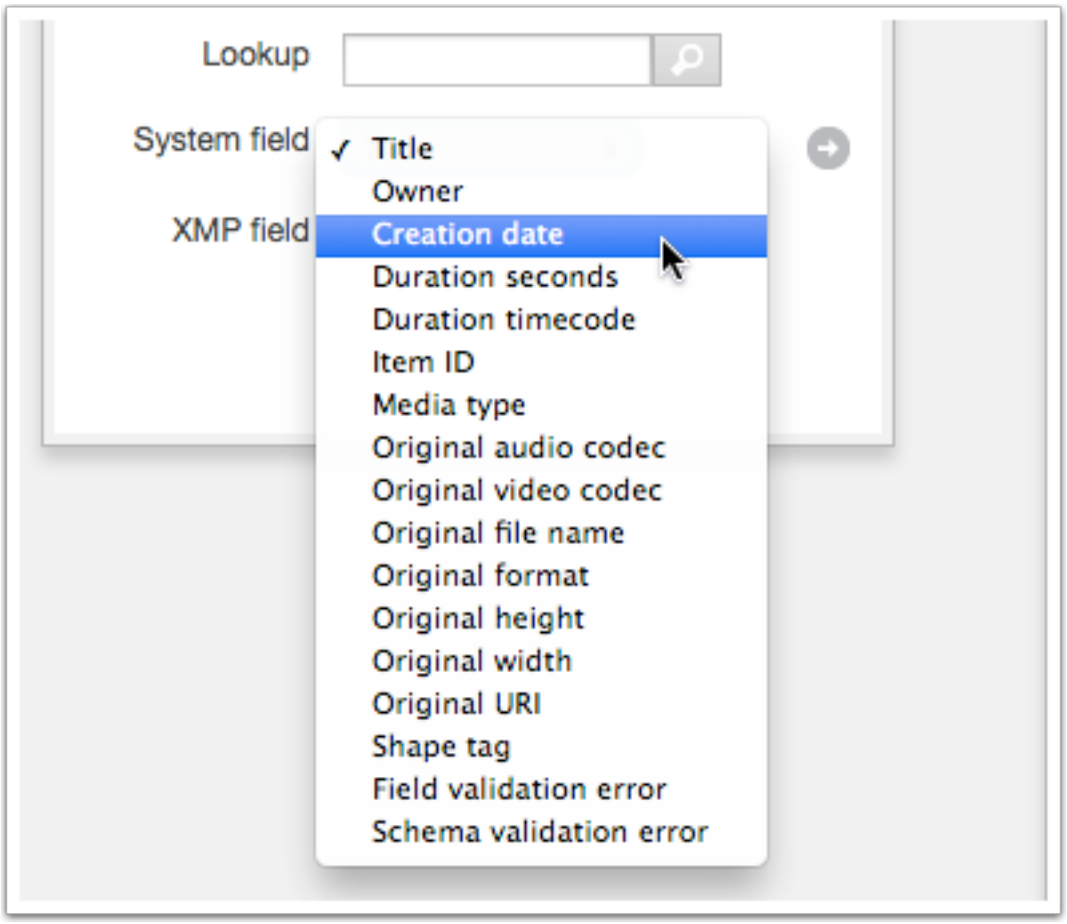
The first context is *Intrinsic system fields*, which are gathered by the Vidispine transcoder when files are initially ingested into Portal. All of these fields are listed under their default name in Metadata Field drop down on the Metadata Filter page. Examples include the *created*, *originalFilename*, *original_mimetype* and *title* fields.



Intrinsic system-level fields are identifiable by the way the name is formatted. The first character is always lowercase and they never include spaces. Often any additional words in the name will be capitalized, such as: *durationSeconds* and *originalFilename*; or words might be separated by an underscore, for example: *original_mimetype* or *portal_ingested*.
You can pass Intrinsic system-level fields to scripts as parameters or check against their values when creating Metadata FIlters.



The second context is Custom system-level fields, such as: Title, Owner, Creation date and item ID. A Portal administrator can add these fields to Metadata Groups and has the ability to assign custom names to better suit each individual context.



When viewed on the item page Custom system-level fields are always read-only.

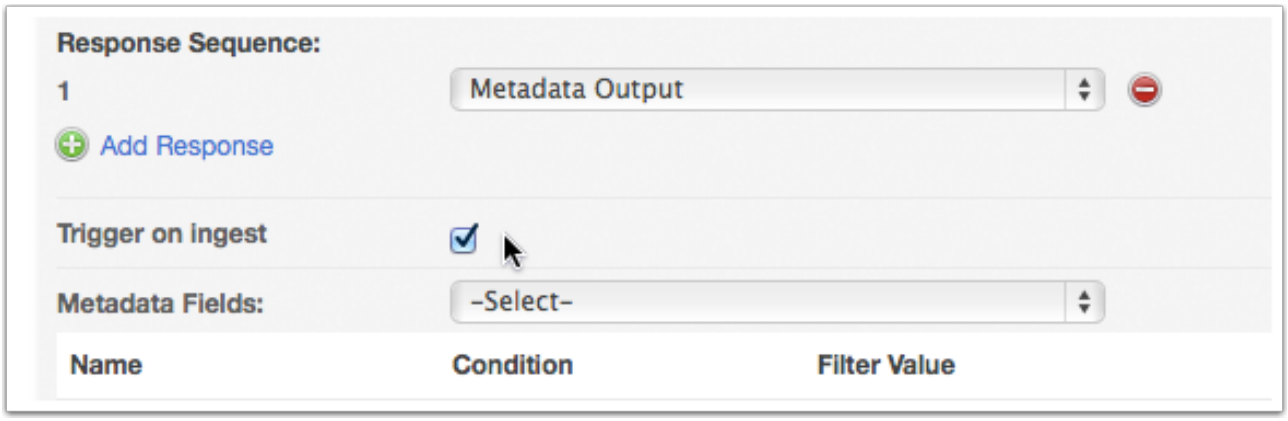
Custom Title	IMG_2734
Custom User	admin
Custom Creation Date	June 18, 2013, 5:35 a.m.
Custom durationSeconds	11.145578
Custom durationTimeCode	11145578@1000000
Custom itemId	VX-63
Custom mediaType	video
Custom originalAudioCodec	aac
Customoriginal VideoCodec	h264
Custom originalFilename	IMG_2734.MOV
Custom originalFormat	mov,mp4,m4a,3gp,3g2,mj2
Custom originalHeight	1080
Custom originalWidth	1920

Custom system-level fields are not supported as conditions to test against in Metadata Filters or as Response parameters to pass to your scripts. Use Intrinsic system-level fields for both tasks instead.

4.7. Pivotal Conditions

Every Metadata Filter should contain one “pivotal condition” to initiate the automation. The Trigger on change checkbox is used to set the condition that fulfils this role. ScriptRunner will fire scripts when metadata is updated to match the specific pivotal condition or conditions. Changes to fields are made by users manually saving updates to metadata fields.

ScriptRunner 1.52+ introduces a new Trigger on Ingest checkbox. This functions as a special pivotal condition that should always be used to create automations that need to be triggered by the ingest of new items into Portal. When the Trigger on Ingest checkbox is selected you do not need to select a Metadata field or use the Trigger on change checkbox.

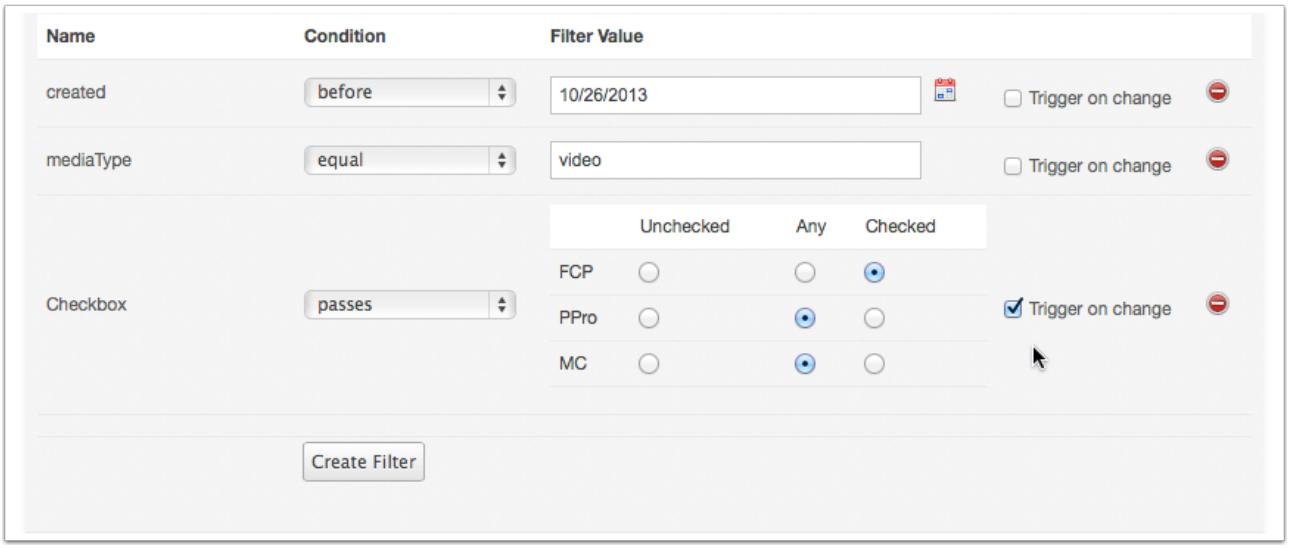


Note: It is important to understand that for Portal “ingest” does not only refer to initial item creation, but can also represent other operations, such as the restore of the original shape or the creation of additional shapes.

4.8. Combining Metadata Conditions

ScriptRunner supports the use of triggers with multiple conditions. This means you can test against several different criteria before a script will fire.

To add additional conditions select a new field from the Metadata Fields drop down menu and configure the individual conditions.



Typically you're likely to only want one condition to act as trigger, the pivotal condition, but you can assign multiple triggers and the script will only fire when all of the designated fields are changed together.

Multiple conditions can also be combined with the Trigger on Ingest checkbox. For example, you could check for different media types and trigger different responses for video, audio or files as they are added to Portal.

4.9. Metadata Subgroups

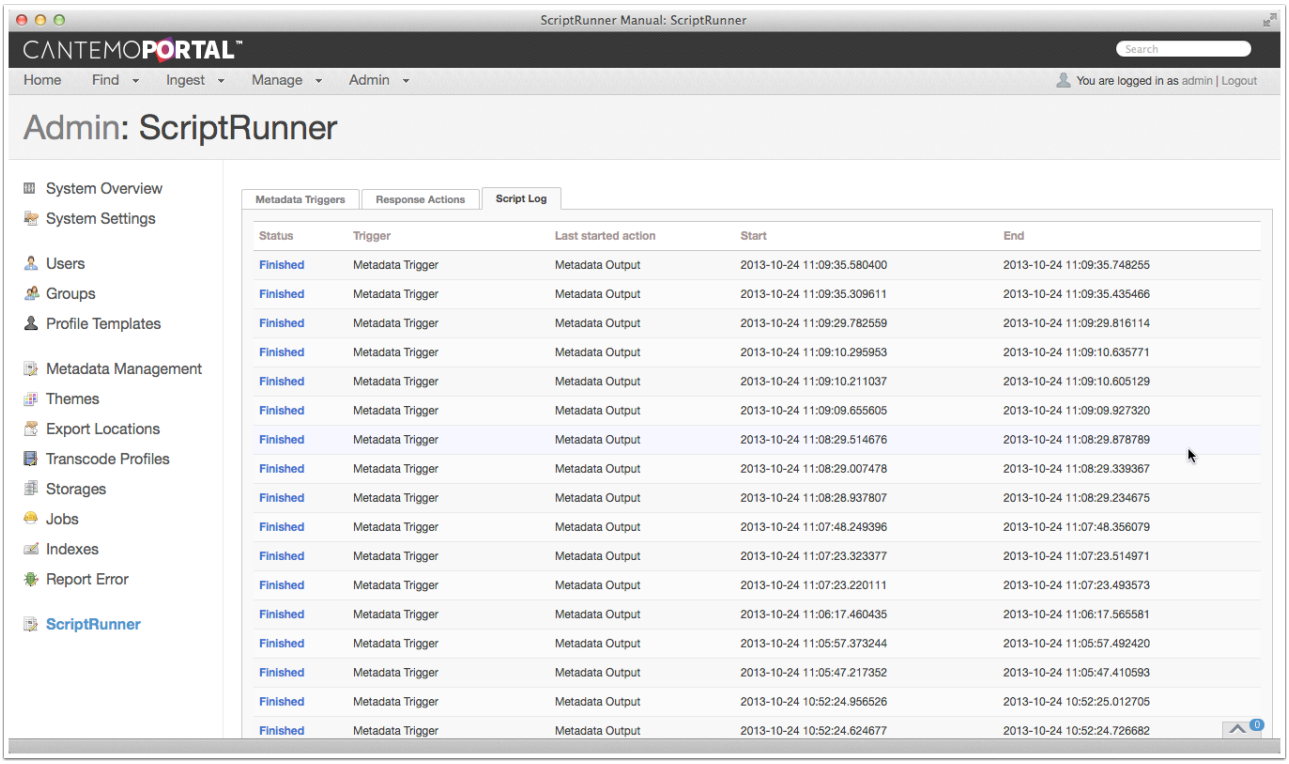
Fields in metadata subgroups can be used as triggers. It is important to note that if you use a field with the same name in multiple subgroups it will trigger scripts using the same conditional values wherever its location.

5. Reviewing the Script Log

ScriptRunner includes a dedicated log to track the status of automations. It can be used to monitor which automations have been fired and help identify issues you might encounter.

5.1. Browsing Logs

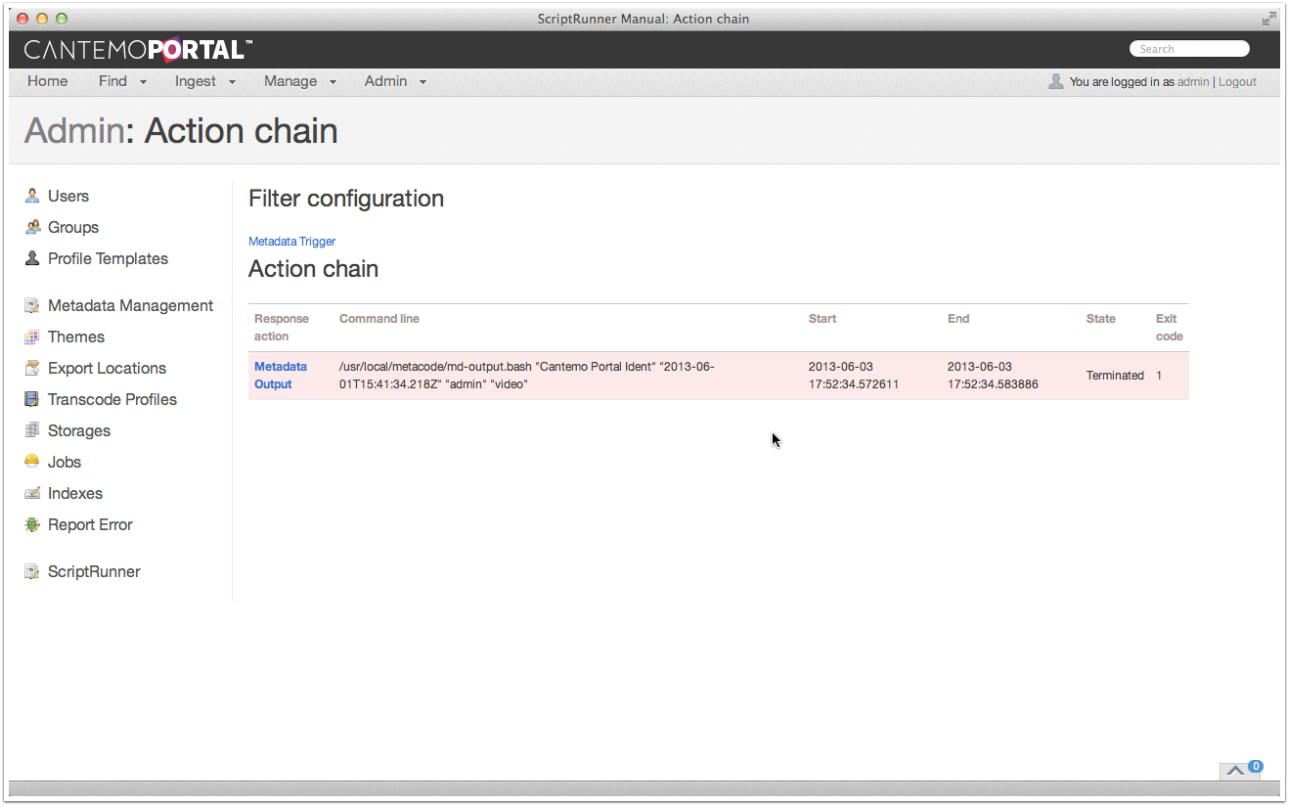
1. Select the Script Log tab in the ScriptRunner Administration page.



The log displays information about the status of each automation, the trigger and response used, as well as timestamps for both initiation and termination of the process.

A Finished status indicates that an automation completed successfully. A Failed status and pink highlight indicate that there was a problem in the execution of the automation.

2. Click on the Status of an automation to access details about its execution.

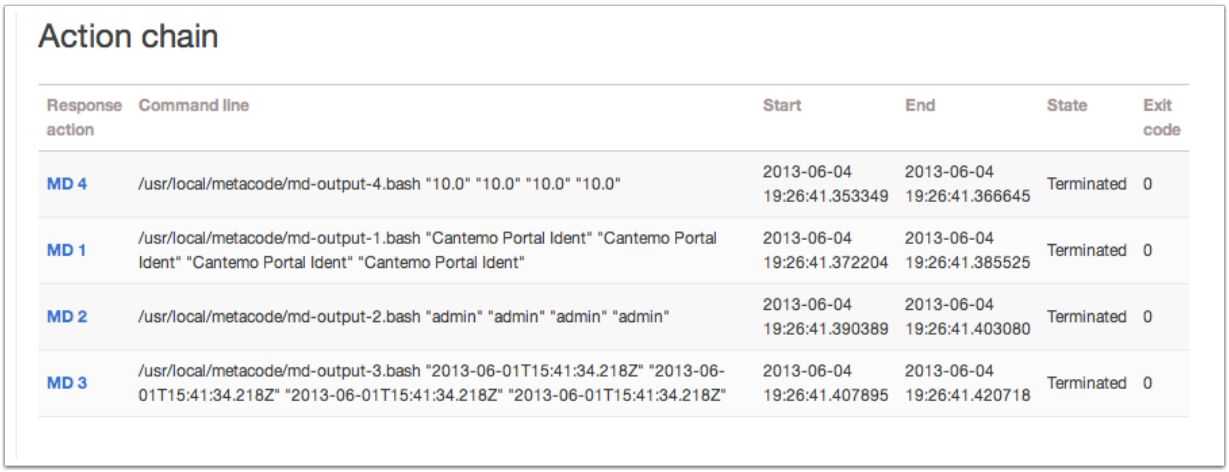


The Action Chain page features a more detailed breakdown of which Metadata Filter was used as trigger and the response action, or actions, that were fired.

The Command line column shows the path to external script and details of the parameters.

An Exit Code of 0 means the automation completed successfully, while an Exit Code of 1 registers a failure.

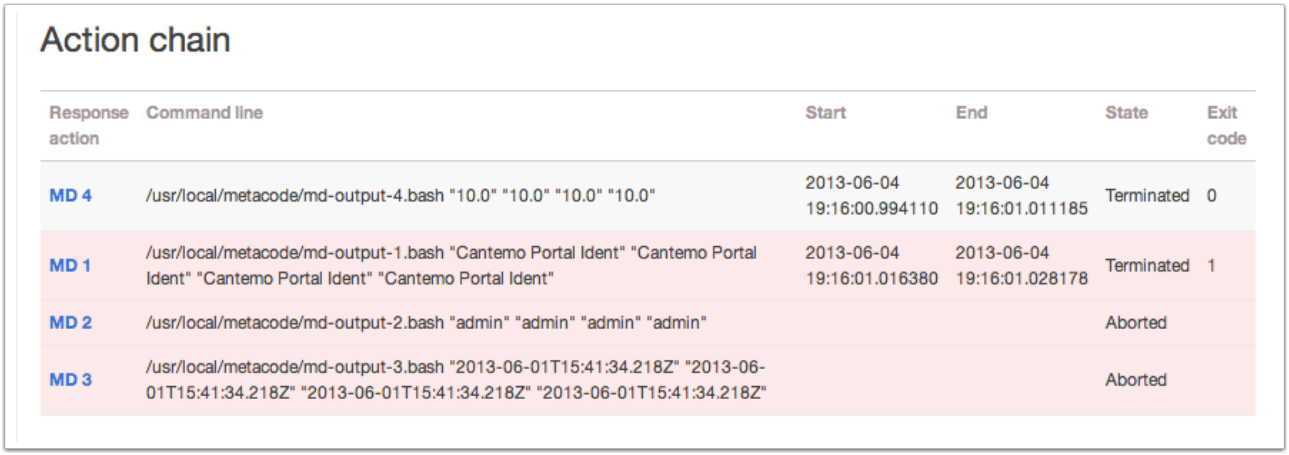
If a trigger is used to fire a sequences of responses you can use the Action Chain to see which responses were fired and the order in which they were executed.



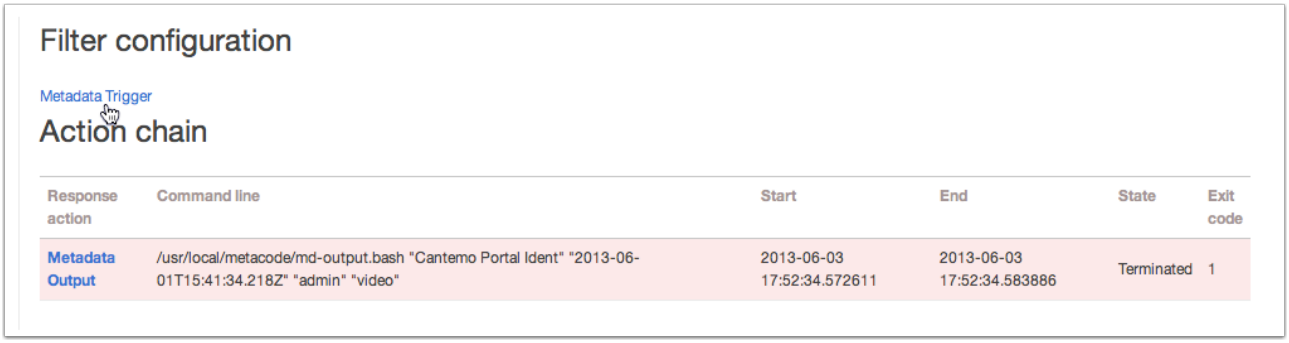
Action chain

Response action	Command line	Start	End	State	Exit code
MD 4	/usr/local/metacode/md-output-4.bash "10.0" "10.0" "10.0" "10.0"	2013-06-04 19:26:41.353349	2013-06-04 19:26:41.366645	Terminated	0
MD 1	/usr/local/metacode/md-output-1.bash "Cantemo Portal Ident" "Cantemo Portal Ident" "Cantemo Portal Ident" "Cantemo Portal Ident"	2013-06-04 19:26:41.372204	2013-06-04 19:26:41.385525	Terminated	0
MD 2	/usr/local/metacode/md-output-2.bash "admin" "admin" "admin" "admin"	2013-06-04 19:26:41.390389	2013-06-04 19:26:41.403080	Terminated	0
MD 3	/usr/local/metacode/md-output-3.bash "2013-06-01T15:41:34.218Z" "2013-06-01T15:41:34.218Z" "2013-06-01T15:41:34.218Z" "2013-06-01T15:41:34.218Z"	2013-06-04 19:26:41.407895	2013-06-04 19:26:41.420718	Terminated	0

Note: Each response in the sequence will only fire if the previous script completed successfully. If an individual script fails, or exits with a code other than 0, the remaining scripts in the sequence will not be triggered.



3. Click the name of the Metadata Filter to navigate to the relevant settings page.



Tip: It can be useful to open the link in a new browser window or tab to avoid having to click back through to the Action Chain page.

4. Click the name of the Response Action to view its configuration settings.

5.2. Clearing the Log

It can be useful to periodically clear the ScriptRunner log. The ScriptRunner-1.6.zip includes a script to perform this task.

1. Navigate to the utilities folder inside the ScriptRunner-1.6 directory.

Note: The following step will restart services on your system.

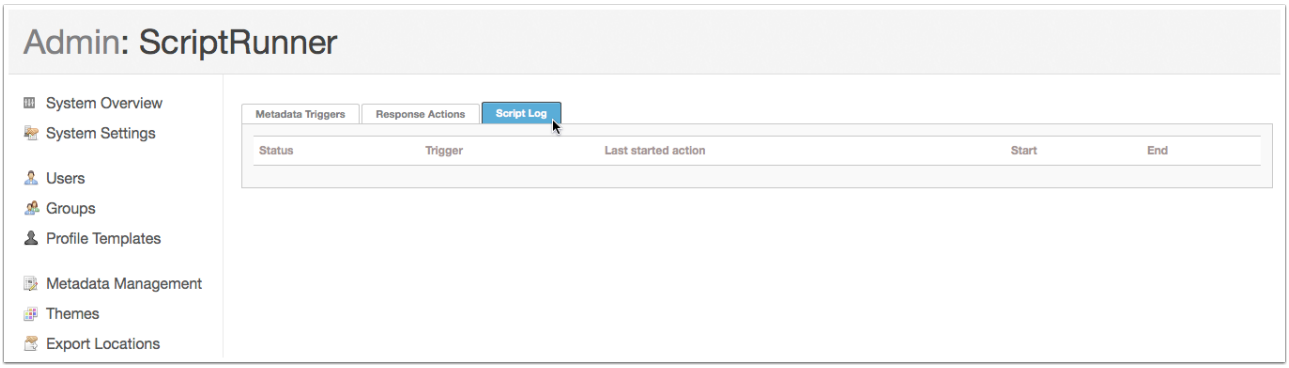
2. Enter the command:

```
$ sudo ./clear_sr_log.bash
```

The output should indicate the successful restart of both Portal and memcached:

```
Python 2.6.6 (r266:84292, Feb 22 2013, 00:00:18)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-3)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> >>> >>> portal: stopped
portal: started
Stopping memcached:                                        [  OK  ]
Starting memcached:                                        [  OK  ]
```

3. Refresh the ScriptRunner Admin page in your browser and click on Script Log tab to see confirm that list has been cleared.



6. Testing Automations

The ScriptRunner-1.6.zip package includes a very simple script file that you can use to confirm that the app has been installed correctly on your Portal server and automations function as expected.

6.1. Copy md-output.bash

The included file is a simple script that outputs 4 metadata parameters from Portal to a log file. It is very easy to setup and configure as a test for your ScriptRunner installation.

1. Navigate to the /usr/local directory.

2. Create a new folder metacode:

```
$ sudo mkdir metacode
```

3. Navigate to the sample_scripts folder in the ScriptRunner-1.6 directory.

4. Copy md-output.bash to the new metacode folder:

```
$ sudo cp md-output.bash /user/local/metacode/
```

5. Check the file has suitable permissions — ScriptRunner executes all external scripts as user, www-data. Ownership and permissions should be set appropriately:

```
$ sudo chown www-data md-output.bash
$ sudo chgrp www-data md-output.bash
$ sudo chmod 744 md-output.bash
```

6. Run a manual test of the script:

```
$ sudo ./md-output.bash 1 2 3 4
```

   (Where 1, 2, 3, 4 are sample, space-delimited parameters.)

   The script creates a log file, md-output.log.

7. Change the ownership and permissions of the md-output.log file to match md-output.bash:

```
$ sudo chown www-data md-output.log
$ sudo chgrp www-data md-output.log
$ sudo chmod 744 md-output.log
```

8. Enter a command to view the last entries in the log:

```
$ tail -f md-output.log
```

If the script executed successfully you should see this output with timestamps:

```
Variable One is: 1
Variable Two is: 2
Variable Three is: 3
Variable Four is: 4

-----
```

Leave the terminal window open to see subsequent updates as you test ScriptRunner.

Tip: Press Ctl-C to terminate the tail command.

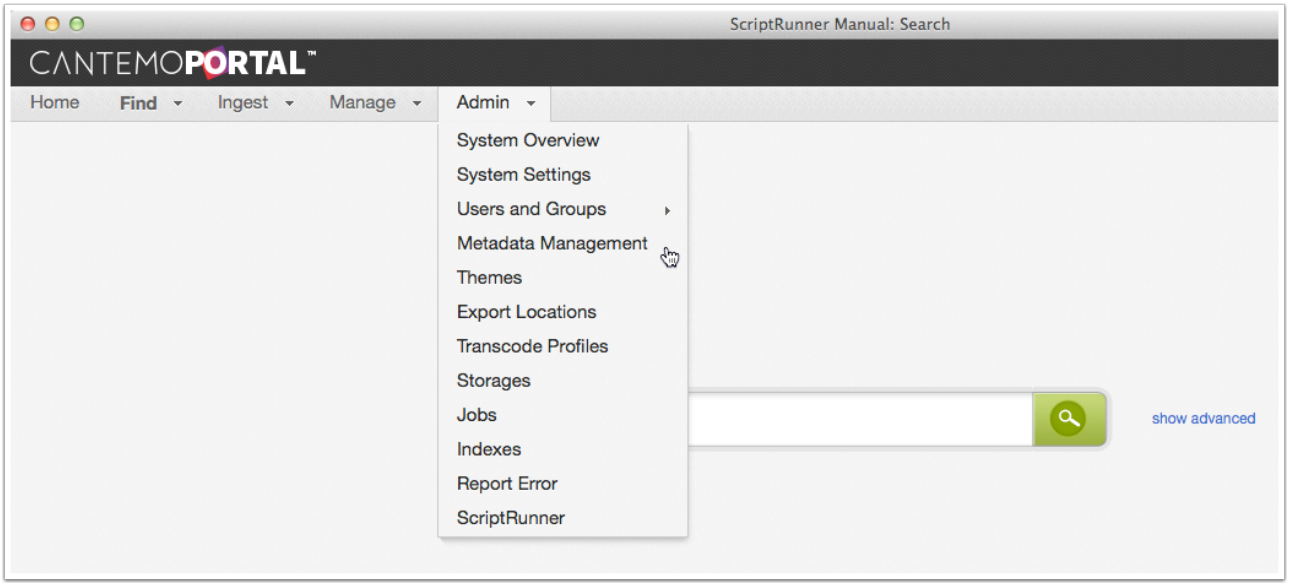
6.2. Create a Metadata Checkbox

The current version of ScriptRunner relies on metadata triggers. The process of creating metadata fields in Portal is relatively straightforward, but you should consult the Metadata Editor section of the Portal Administrator's Documentation for more details.

In this example you'll create a simple checkbox.

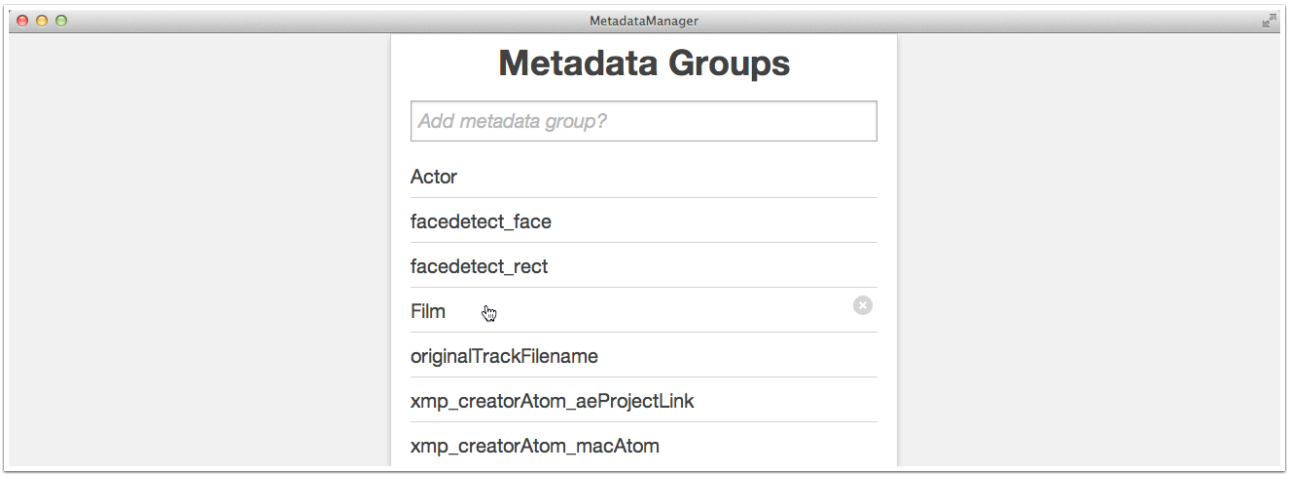
1. Open a web browser and login to Portal as an administrator user.

2. Choose Admin > Metadata Management to open the Metadata Editor window.

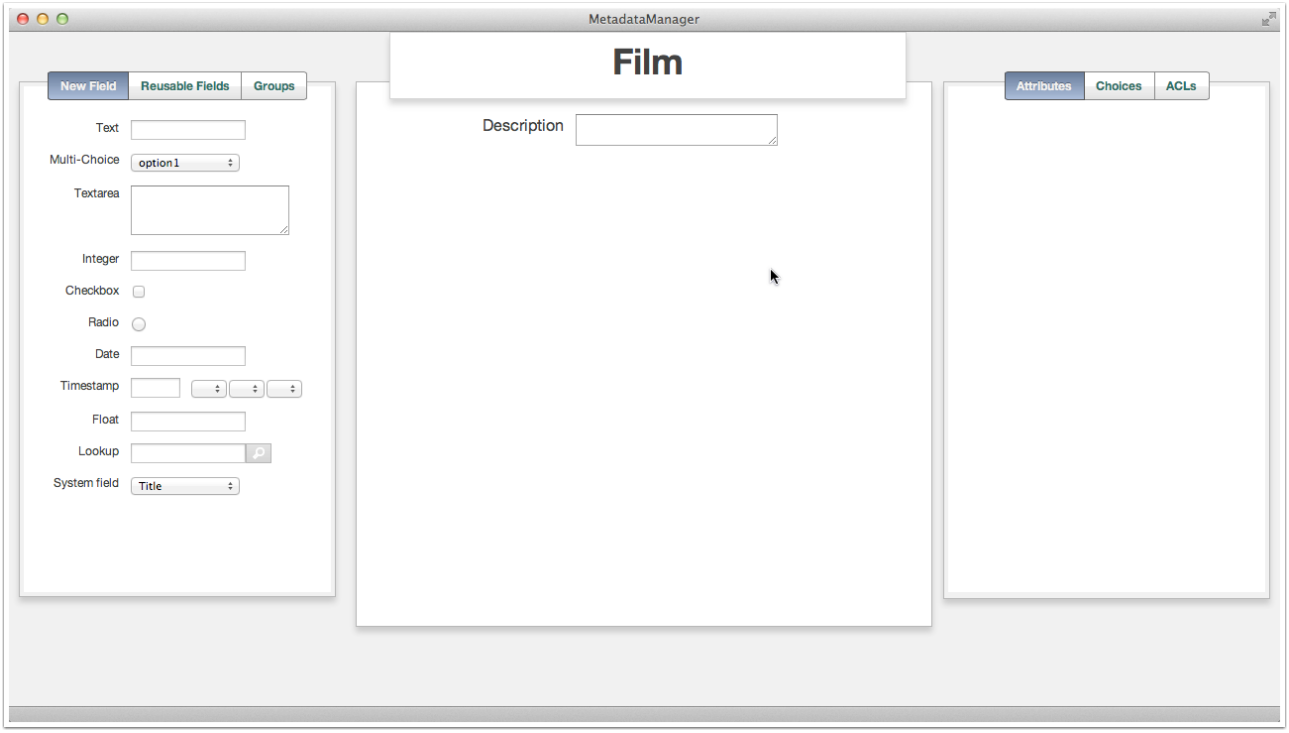


3. Choose Film from the list on the Metadata Groups page.

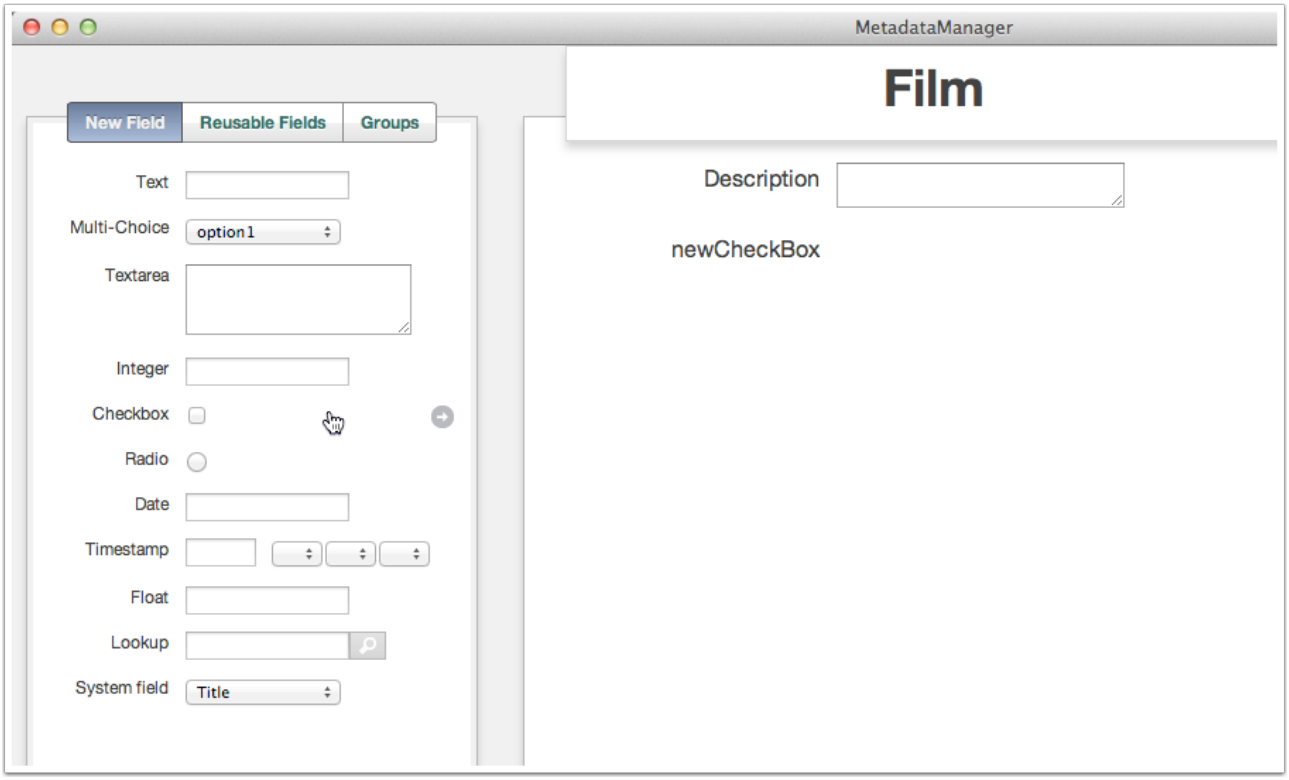
Note: Film is the default metadata group in new Portal installations and is the group used in this example workflow.



The Metadata Group Editor has three panels. You use the Metadata Fields Chooser on the left to select fields from the range of available items. Any element you select in the Chooser is automatically added to the Metadata Group Preview in the middle. Once you've added a field, select it in the Preview panel to edit the associated attributes on the right.

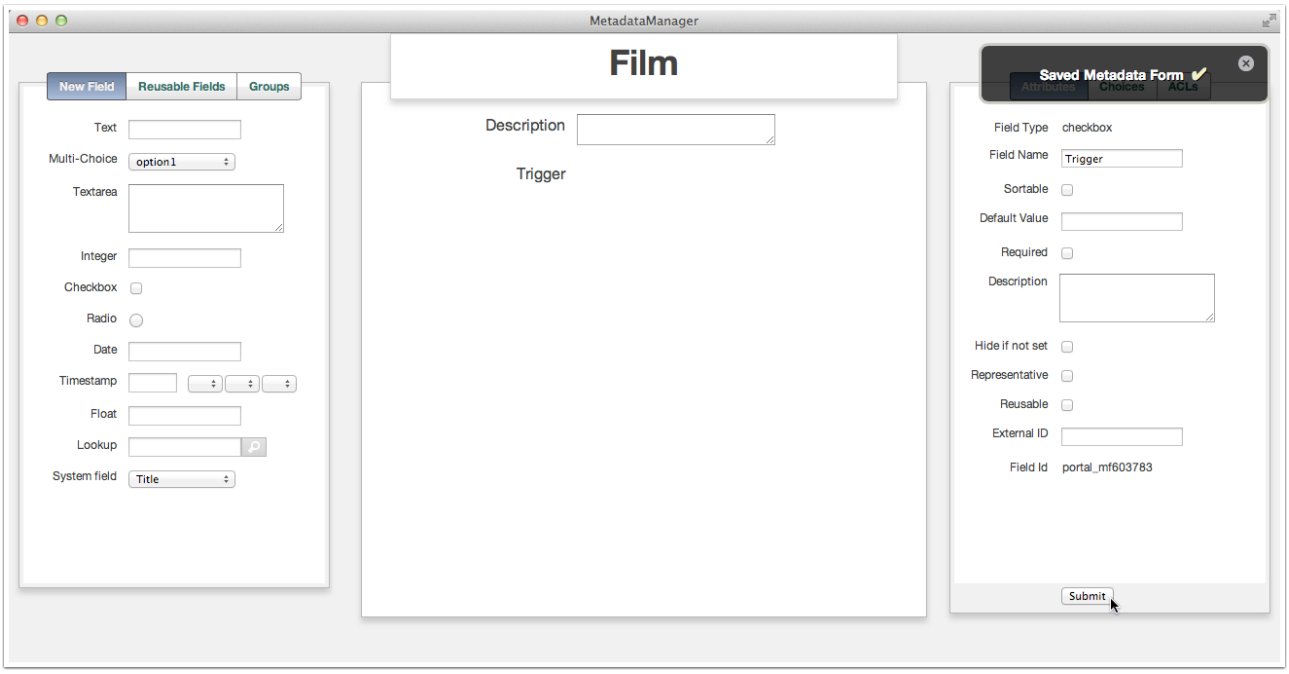


4. Click the Checkbox option in the Fields Chooser to add a newCheckBox item.



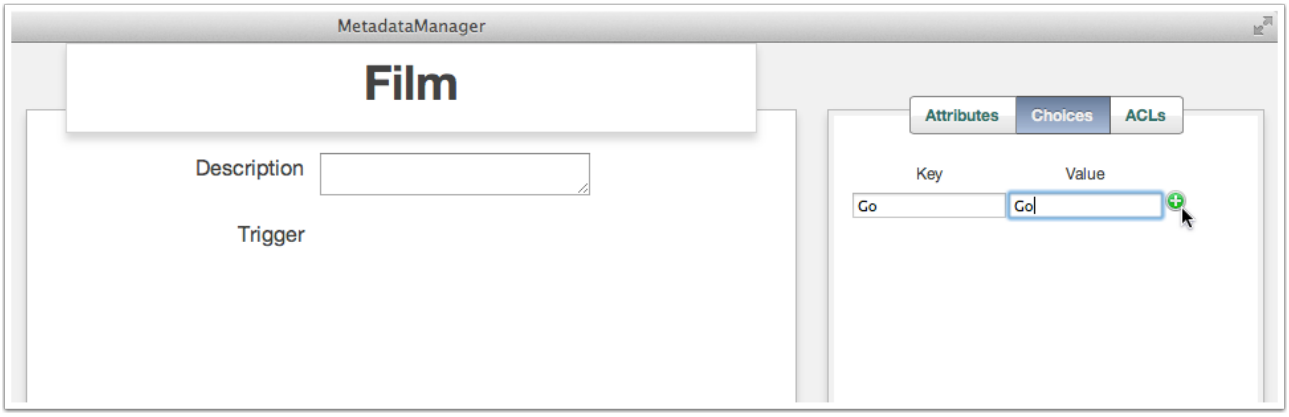
5. Select newCheckBox and enter an appropriate name: Trigger.

6. Click the Submit button to save your changes.

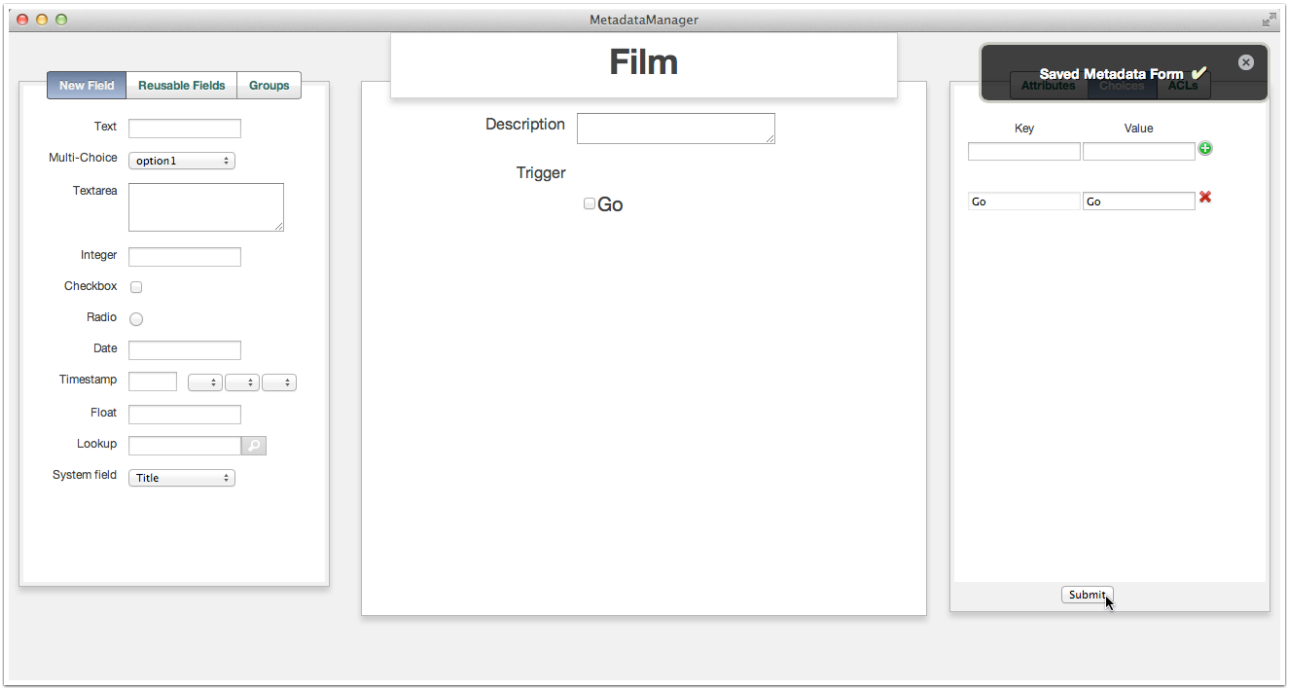


7. Select the Choices tab to enter a unique value and a key: Go.

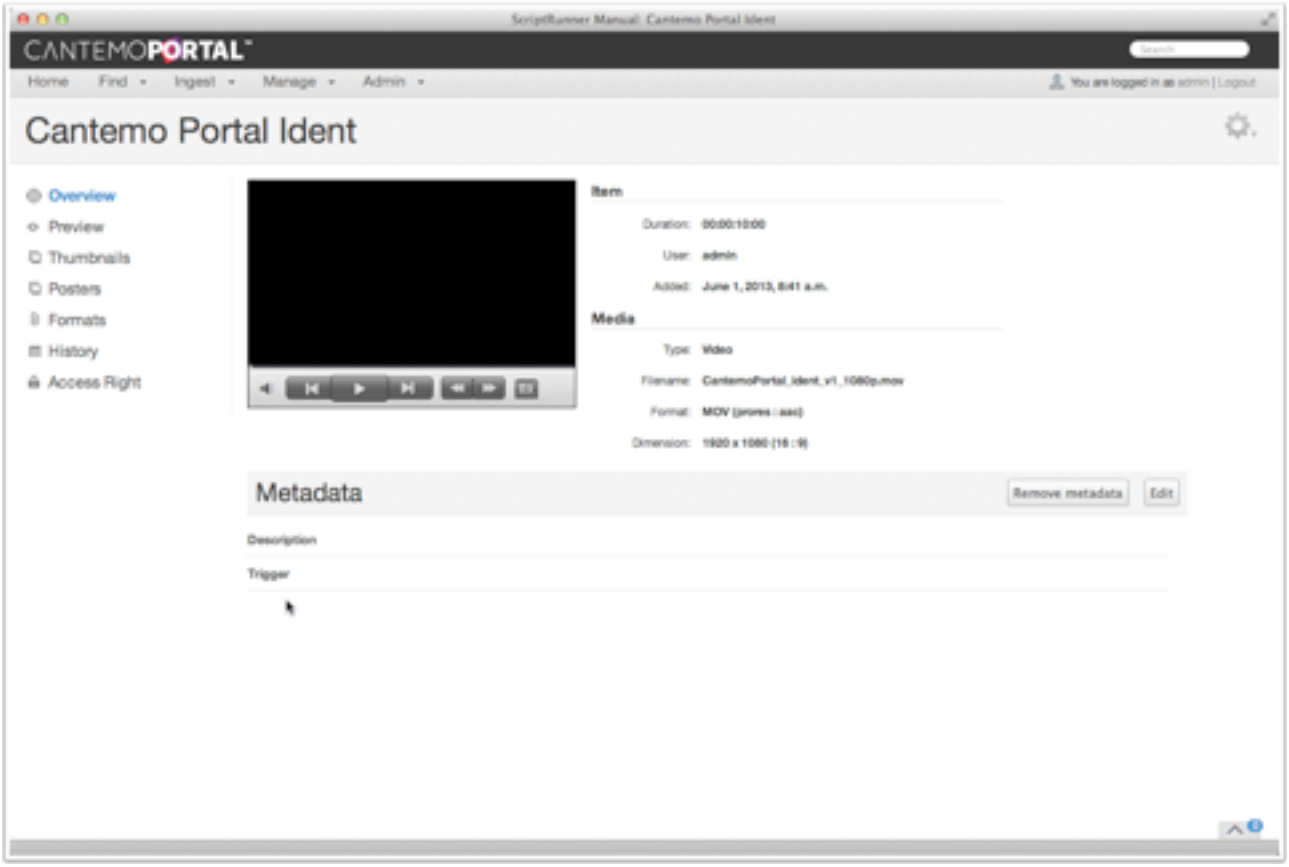
8. Click the plus button (+) to add the new option.



9. Click the Submit button to create the new option.

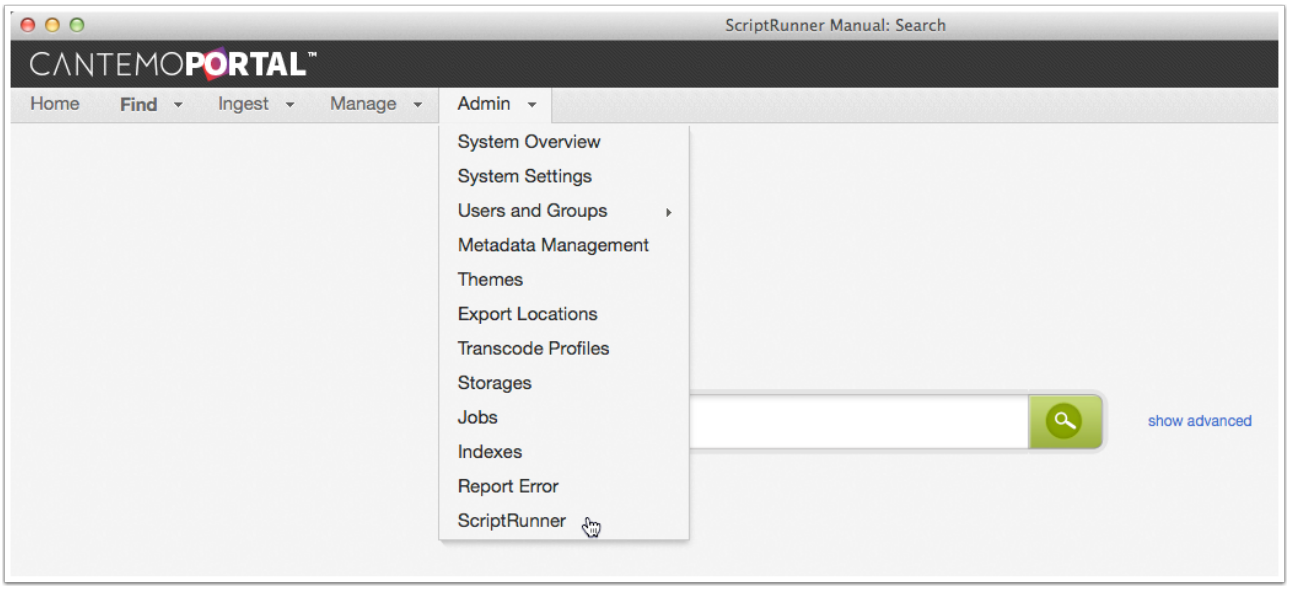


10. Open an individual item page to view your new metadata field in context.



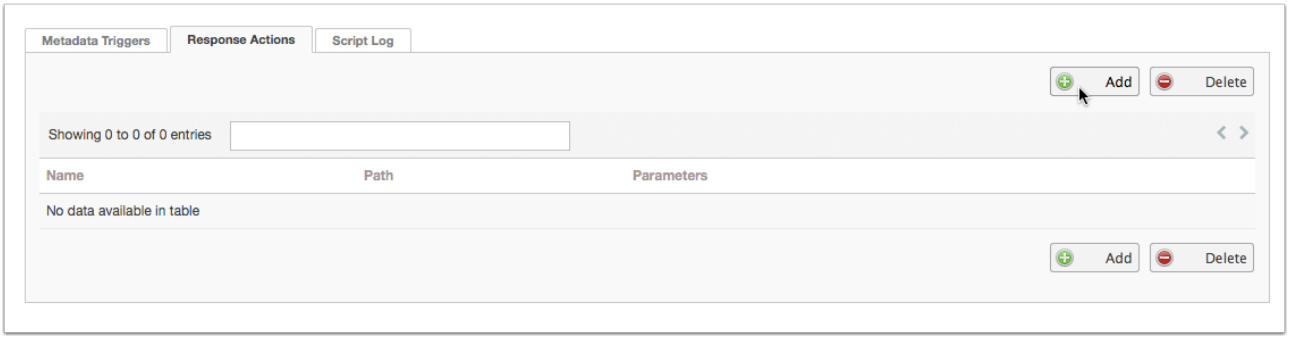
6.3. Create the Metadata Output Response

1. Open the ScriptRunner Admin page.



2. Select the Response Actions tab.

3. Click the Add button to create a new response.

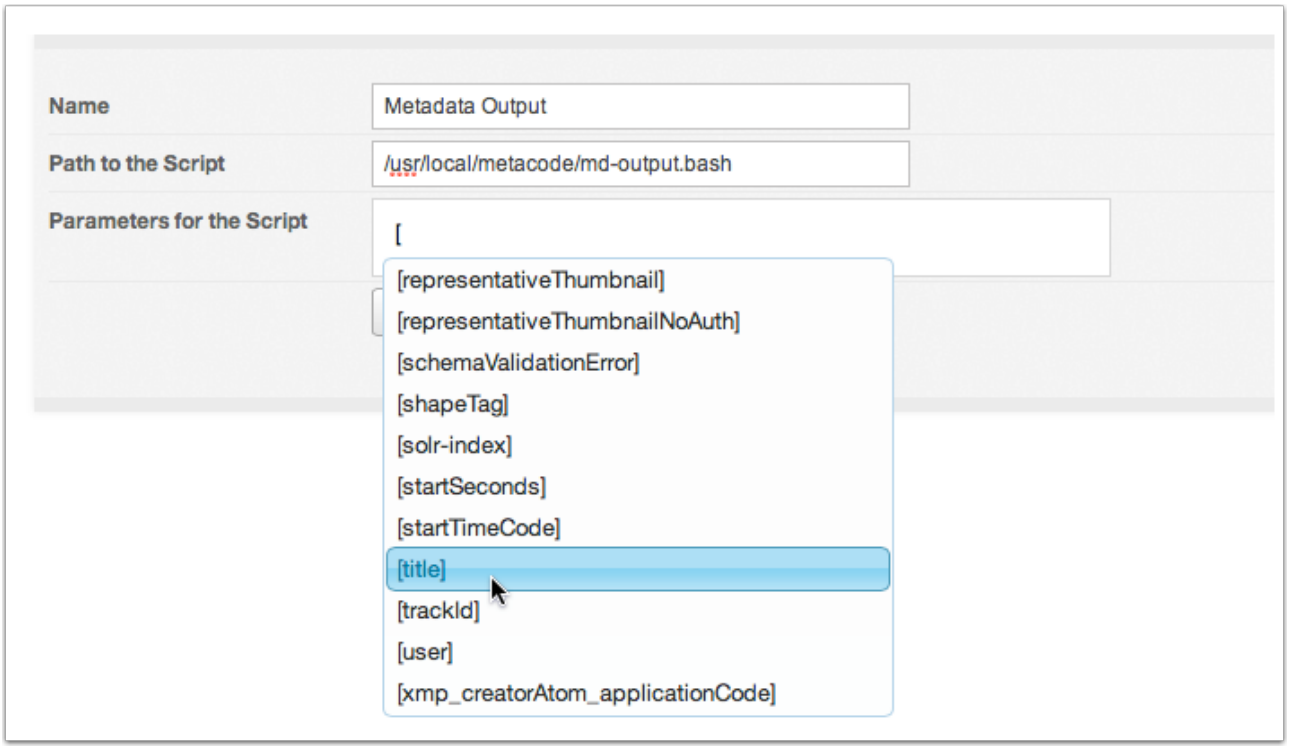


4. Enter the name: Metadata Output.

5. Enter the absolute path to the script: /usr/local/metacode/md-output.bash.

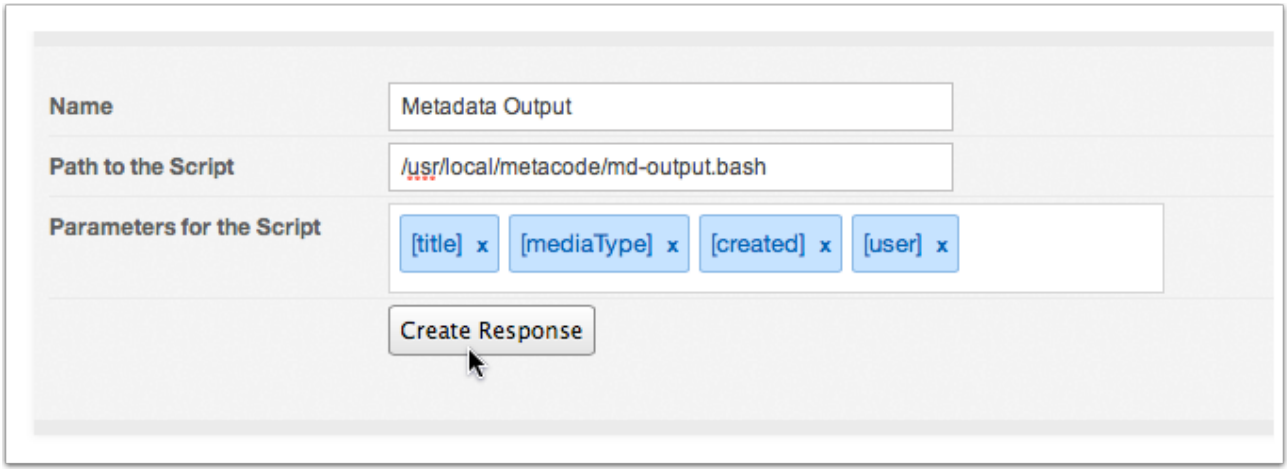
6. Type an open square bracket "[" into the Parameters for the Script field.

7. Choose the first parameter you'd like to send to the script from the list.

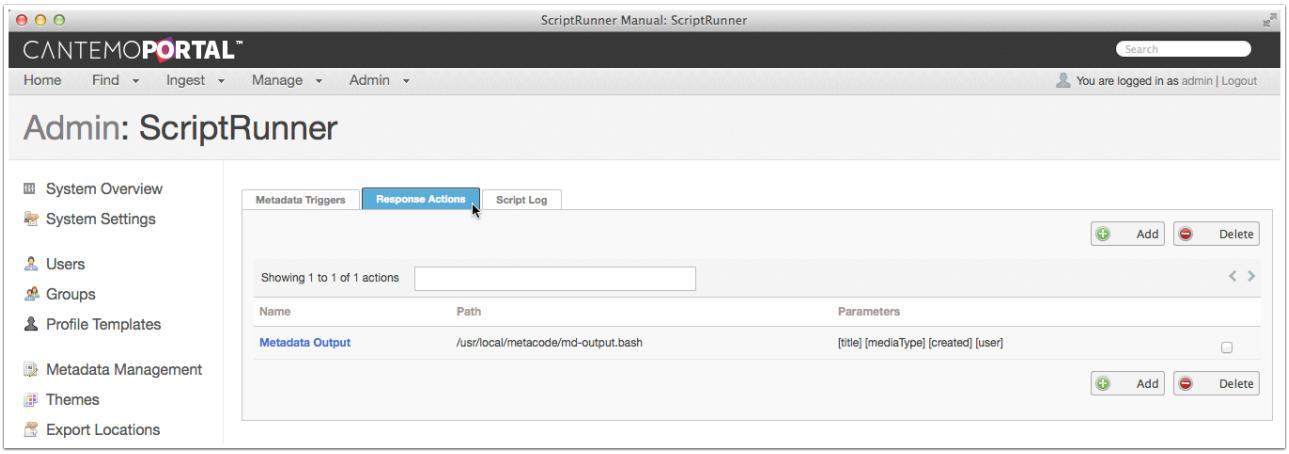


8. Repeat steps 6-7 to create 3 more parameters for the script to output.

9. Click the Create Response button to save your work.



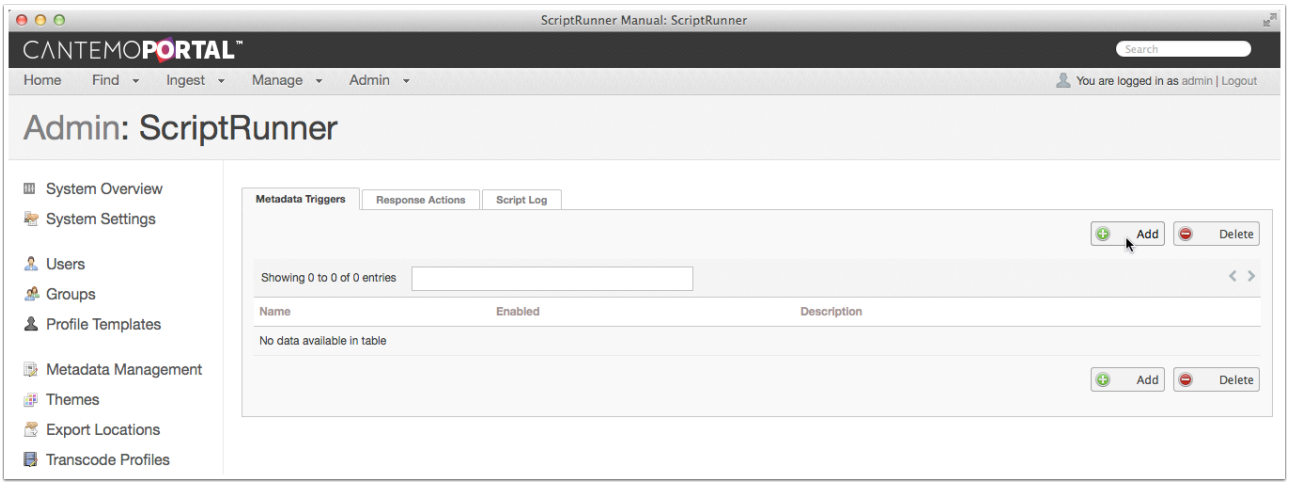
10. Click the Response Actions tab to confirm your new response has been created successfully.



6.4. Create the Metadata Trigger

1. Click the Metadata Trigger tab in the ScriptRunner Admin page.

2. Click the Add button to create a new trigger.



3. Enter the name: Metadata Trigger.

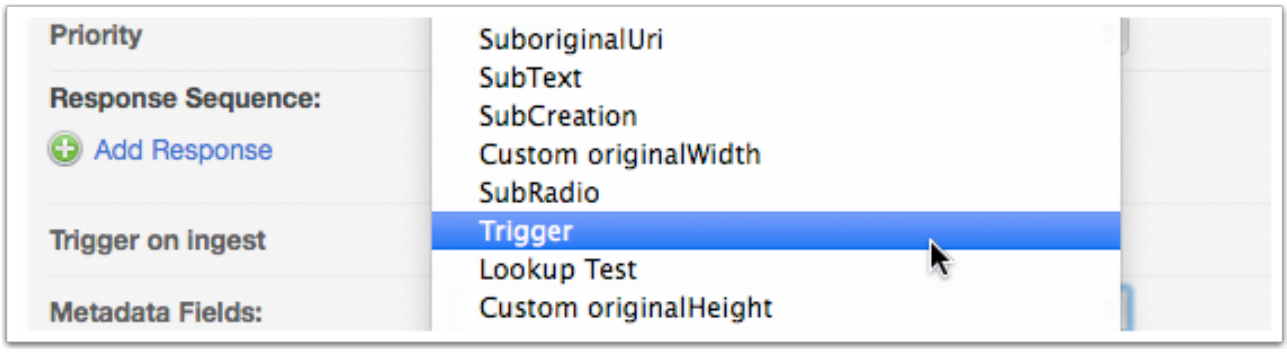
4. Click to select the Enabled checkbox.

5. Enter a simple a straightforward description: Checkbox trigger to output specified parameters to a log file.

6. Set the Priority to Normal.

7. Click Add Response and choose Metadata Output from the drop-down menu.

8. Choose the new Trigger metadata field from the Metadata Fields menu.

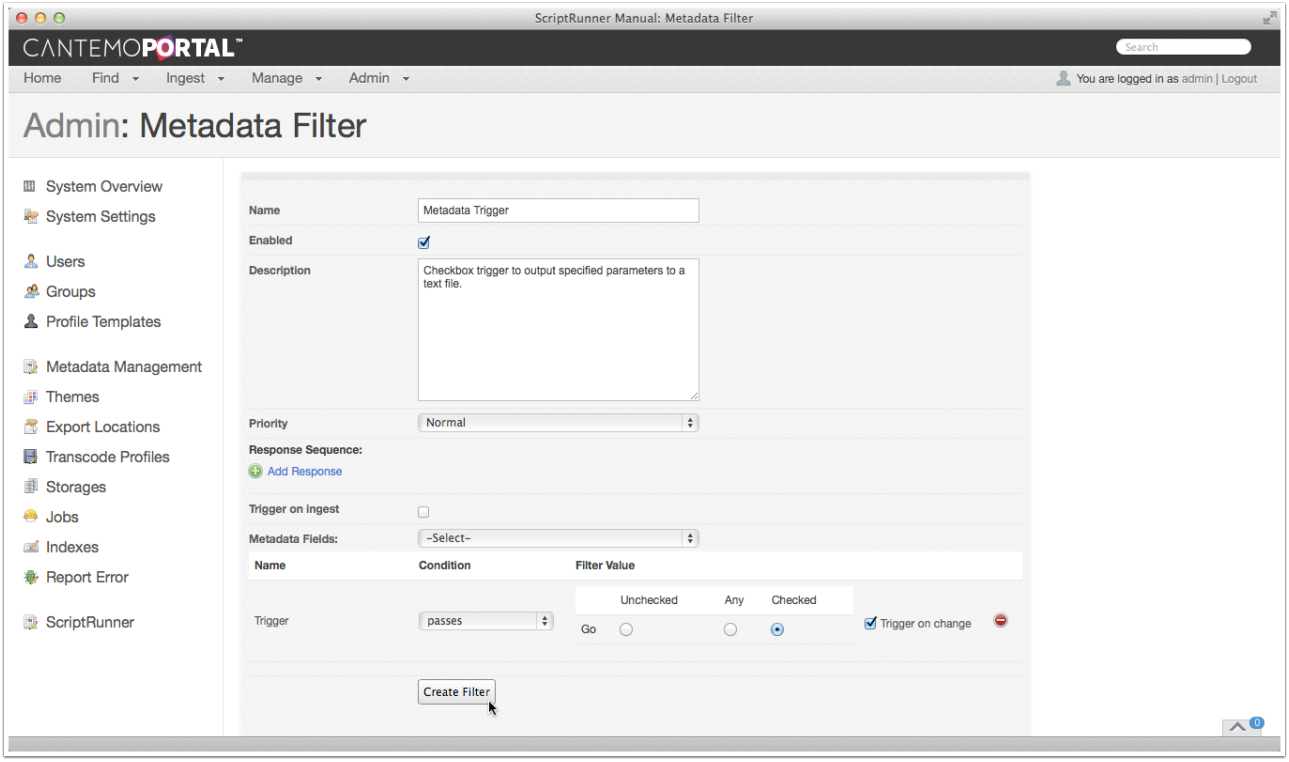


Note: If the Trigger field is not visible you may need to restart memcached (see 7.4.3. Metadata Not Visible).

9. Ensure that the Condition is set to passes and the Go option is set to Checked.

10. Click to select the Trigger on change checkbox.

11. Click the Create Filter button to save your changes.

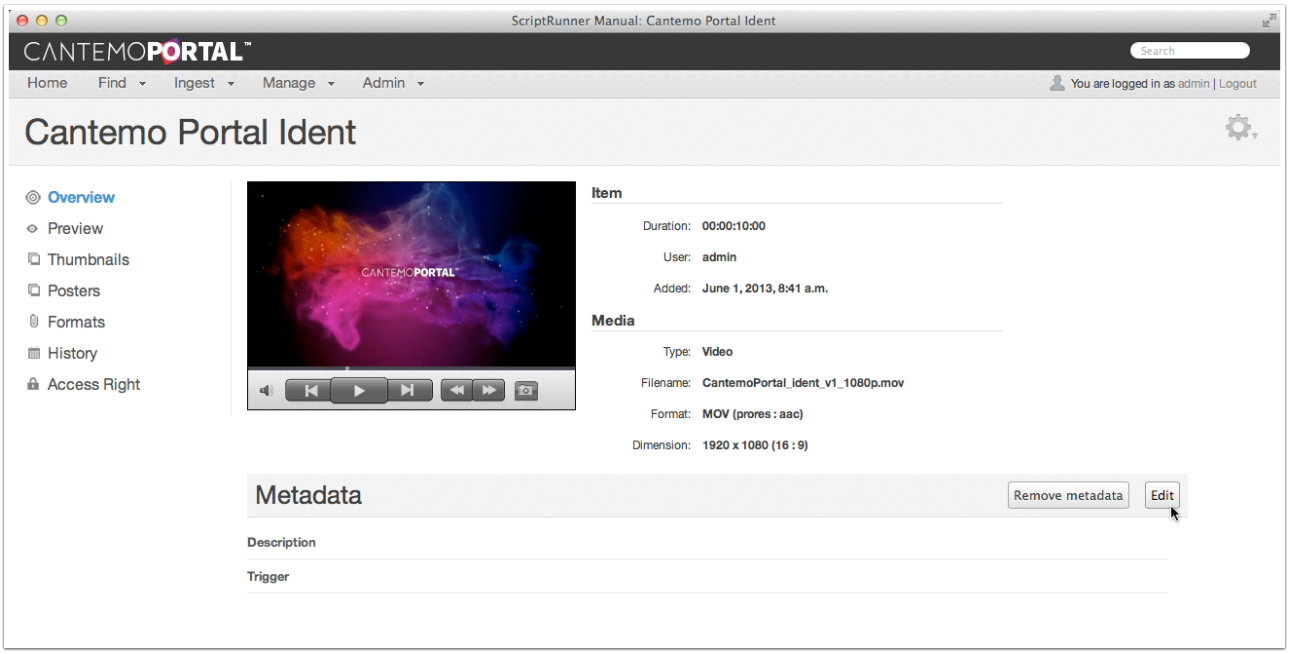


12. Confirm that your new filter appears in the list of Metadata Triggers.

6.5. Testing the Automation

1. Open an individual item page.

2. Click the Edit button to make the metadata fields modifiable.



3. Select the Go checkbox.

4. Click the Submit button to apply your change and trigger the response.

5. Check the terminal to see if the log file has been updated:

```
$ sudo tail -f md-output.log
```

If the automation has been successful you'll see new values. Check the Script Log in the web browser for more details.2

7. Troubleshooting

Should you encounter unexpected behavior there are a few procedures and steps you can take to get ScriptRunner back on track.

7.1. Installation

ScriptRunner installation has been designed to be a straightforward and painless process. However, if you do experience issues the following areas are important to check.

7.1.1. Version Number

It is quite likely that ScriptRunner will receive maintenance updates to ensure compatibility with the latest Portal release. It is important that you check compatibility before you update your system and confirm the release is supported.

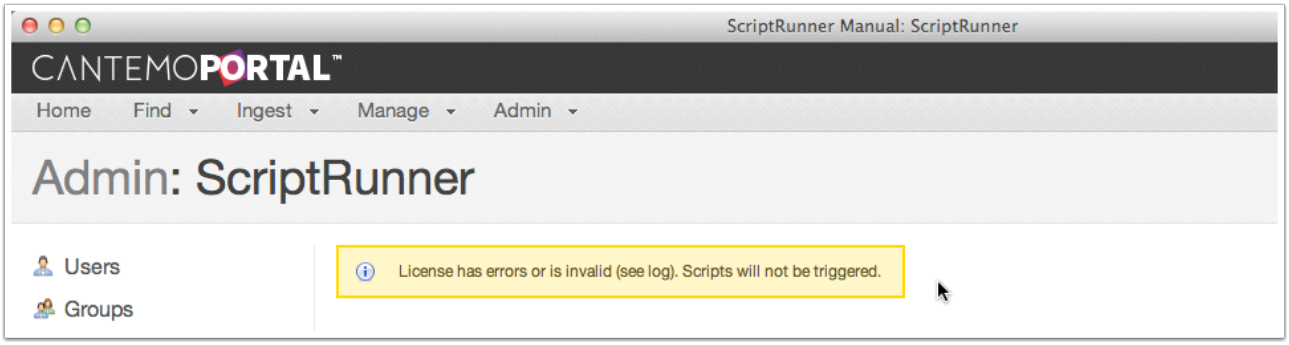
You can check the version number of your ScriptRunner installation under Registered Apps on the Portal Admin page or the command line.

The command is different depending on the OS of your Portal system. On Red Hat/CentOS the command is:

```
$ yum list | grep mmctscripts
```

7.1.2. License Key Error

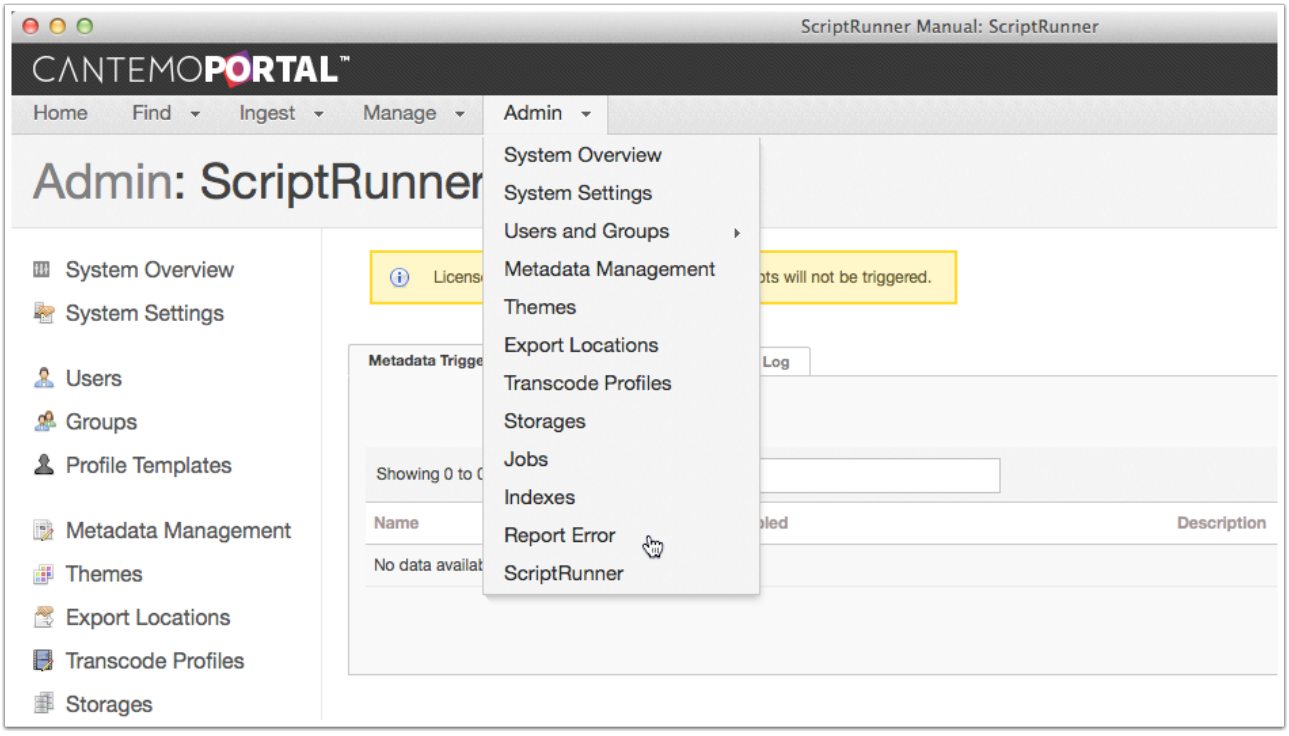
If you do not have a current license key for ScriptRunner, Portal will display an error message on the ScriptRunner Admin page.



The easiest way to review the Portal log file is to generate an Error Report.

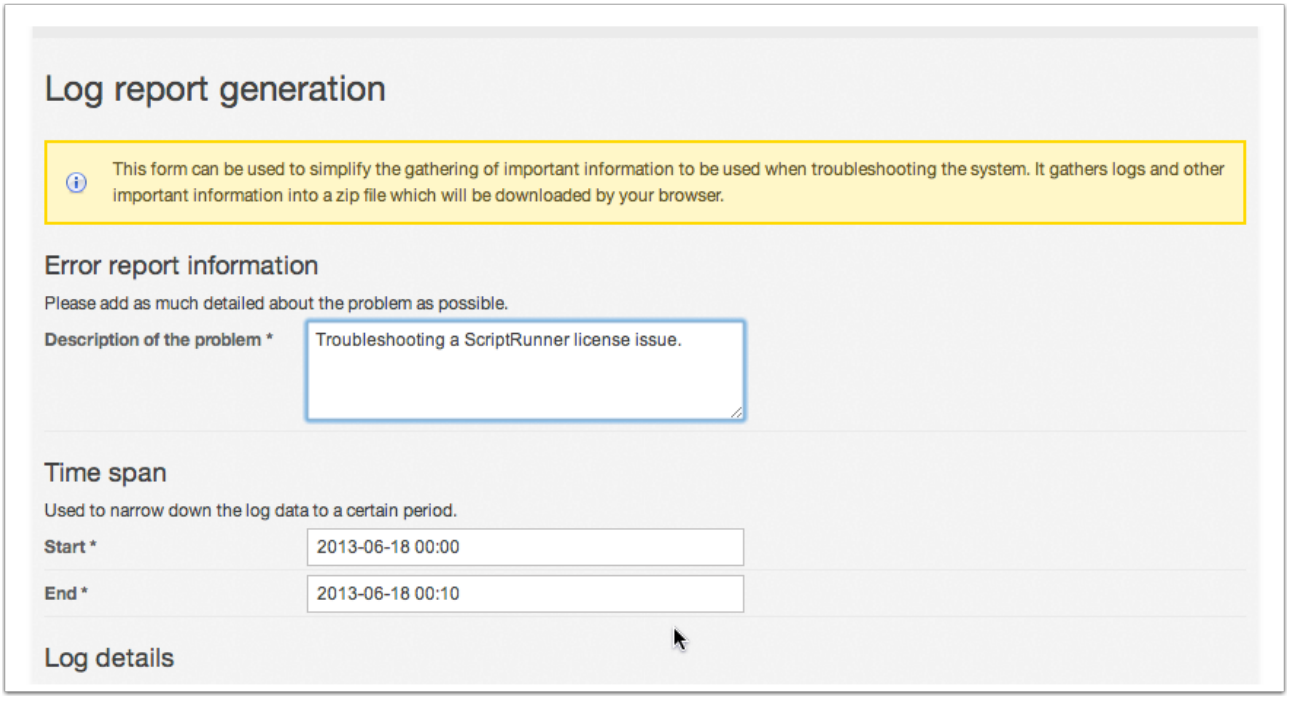
1. Open a web browser and log into Portal as an Administrator user.

2. Click on the Admin link and select the Report Error option.



3. Enter a meaningful description of the error.

4. Set a useful time span — too short a period may not reveal the error and too long can become unwieldily.



5. Click the Submit button to generate the report. The report downloads as a .zip file. Depending which browser and OS you're using you may be prompted to choose a save location.

6. Navigate to the error report folder (it is timestamped to help you locate the correct information) and locate the portal.log file.

7. Open the portal.log file.

You are likely to see several errors, which will help you identify the problem:

```
portal.plugins.mmctscripts.plugin - ERROR - Failed to load plugin license:
License is invalid for sitename yoursitename.

portal.plugins.mmctscripts.views.notification - INFO - Not handling
metadata modification since license is invalid/expired

APIUtility - ERROR - The server could not fulfill the request.
APIUtility - ERROR - Error code: 402
APIUtility - ERROR - Message: Payment Required: No payment -- see charging
schemes
APIUtility - ERROR - <?xml version="1.0" encoding="UTF-8" standalone="yes"?
><ExceptionDocument xmlns="http://xml.vidispine.com/schema/
vidispine"><licenseFault><message>Your license did not validate. Please
check /API/version for more information</message></licenseFault></
ExceptionDocument>
```

The license key file is configured for individual Portal installations and cannot be reused. Please contact MetaCode should you require a new license key.

You should ensure that the license key has been installed correctly. The designated location for the file is: /etc/cantemo/portal. You need to restart Portal whenever you install a ScriptRunner license key using the command:

```
$ sudo supervisorctl restart portal
```

7.1.3. Restarting Services

Different stages in the ScriptRunner install process require various services to be restarted. Should one service not have restarted correctly, you may experience erratic behavior. The easiest remedy to ensure everything has been restarted is to reboot your server. You can also use the following commands to restart Portal and memcached.

```
$ sudo supervisorctl restart portal
$ sudo /etc/init.d/memcached restart
```

If you have other services that utilize supervisorctl you can just restart portal with the command:

```
$ sudo supervisorctl restart portal
```

(See the Portal Administrator's Documentation for more information.)

7.2. Log

If everything is installed correctly and your scripts are not functioning as expected you should use the Script Log to identify issues.

7.2.1. Failing Scripts

The Script Log will reveal whether an automation has failed (see 5. Script Log for more details).

7.2.2. Incomplete Actions

The Script Log includes a detailed Action Chain page for each automation. Here you can see which scripts were utilized and the parameters that were passed from Portal. If the path to the script is correct and the parameters listed as expected it's possible that the issue resides in the external script. You should test that independently to verify all is correct (see 7.3.2. External Test).

7.3. Response Actions

There are several reasons a response may not fire as expected.

7.3.1. Correct Path

Check the path you entered for typos, improper structure, or an incorrect filename.

7.3.2. External Test

If ScriptRunner fails to execute a script correctly, it is always worth checking to make sure the script functions as expected when you trigger it manually from the command line. You'll soon be able to tell where the problem resides.

7.3.3. Permissions

ScriptRunner requires the external script have the correct permissions. The www-data user must be able to execute the file and permissions need be set appropriately.

7.3.4. Correctly Formed Parameters

You can introduce issues if the parameters you enter in your Response are not correctly formed. ScriptRunner currently flags parameters that are formed with an incorrect syntax, but it's unable to detect typos in the name or whether a parameter does not exist in Portal. For this reason you should use the lookup function to reduce the risk of error (see 3.1. Creating Response Actions for details).

7.4. Metadata Triggers

Should a response refuse to fire at all there are few options to double-check on the Metadata Filter page.

7.4.1. Enabled

Confirm that the trigger has been properly enabled (see 4.1. Adding a Metadata Trigger).

7.4.2. Response Assignment

Check to make sure the correct response has been set and that it appears in the correct position in the Response Sequence.

7.4.3. Metadata Not Visible

Newly created metadata fields may not appear in the Metadata Fields drop-down. You need restart memcached with the command:

```
$ sudo /etc/init.d/memcached restart
```

Once you've refreshed the page in your web browser any new fields should appear in the drop-down list.

7.4.4. Trigger Conditions

Should your trigger appear erratic you ought to confirm that the conditions have been set correctly and that domain.xml has been updated as required (see 7.1.3. Correct Transaction Support).

If your trigger is configured correctly and the conditions are right for it to fire, restarting memcached will make usually resolve any inconsistencies:

```
$ sudo /etc/init.d/memcached restart
```

7.5. Broken Sequence

When working with a response sequence, the individual responses in the sequence are wholly dependent on the successful execution of the last. A response will only fire if the previous script completed successfully. If one script fails, or exits with a code other than 0, the remaining scripts in the sequence will not fire (see 5.1. Browsing Logs).

8. Uninstalling

Note: Uninstalling ScriptRunner requires Portal to restart.

8.1. Uninstall from Red Hat/CentOS

Enter the commands:

```
$ sudo rpm -e mmctscripts
$ sudo supervisorctl restart portal:
```

Note: The uninstall command does not remove the license key file or any of the Metadata Triggers and Response Actions from your system. You need to delete these manually. If you reinstall ScriptRunner without removing these files you will not need to update a valid license key or recreate your previous automations.

9. Appendix: RabbitMQ Acceleration

Anyone familiar with high availability installations of Portal will have encountered RabbitMQ message brokering software. Its uses extend beyond HA applications however and it can also be used to accelerate background processes in a single Portal node. To maximize performance, ScriptRunner 1.6 features support for RabbitMQ on Red Hat/CentOS installations.

Please note that with Portal 2.0+, the installation of RabbitMQ is not necessary since Portal 2.0+ installs RabbitMQ by default.

Note: The installation and configuration of RabbitMQ message brokering requires Portal be restarted.

9.1. Installing Erlang

RabbitMQ requires an Erlang language interpreter to run.

1. Download and install the Fedora yum repository:

```
rpm -Uvh http://download.fedoraproject.org/pub/epel/6/i386/epel-
release-6-8.noarch.rpm
```

2. Download and install erlang:

```
$ sudo yum install erlang
```

9.2. Installing RabbitMQ

Before you install RabbitMQ check the official web site to ensure you have the latest full release (http://www.rabbitmq.com/releases/rabbitmq-server).

1. Use rpm to install the appropriate version of RabbitMQ:

```
$ sudo rpm -Uvh http://www.rabbitmq.com/releases/rabbitmq-server/v3.x/
rabbitmq-server-3.x.noarch.rpm
```

2. Initialise RabbitMQ:

```
$ sudo chkconfig rabbitmq-server on
```

3. Startup RabbitMQ:

```
$ sudo service rabbitmq-server start
```

9.3. Setup RabbitMQ and Portal

By default Portal uses the main database for all messaging transactions. Once RabbitMQ is installed the Portal configuration needs to be adjusted to take advantage of the new brokering service.

9.3.1. Configure RabbitMQ

Before Portal can use RabbitMQ you must setup a new virtual host and user with the correct permissions to read, write and queue messages.

1. Create a Portal user:

```
$ sudo rabbitmqctl add_user portal secret
```

2. Add a virtual host:

```
$ sudo rabbitmqctl add_vhost portal
```

3. Define the appropriate user permissions:

```
$ sudo rabbitmqctl set_permissions -p portal portal ".*" ".*" ".*"
```

9.3.2. Configure Portal

For Portal to utilise RabbitMQ you must make a modification to the portal.conf file.

Note: It is best practice to create a backup of the portal.conf file before you make any changes.

1. Navigate to the portal.conf directory (/etc/cantemo/portal).

2. Open portal.conf in an editor.

3. Add a setting to to the [celery] section:

```
BROKER_URL = amqp://portal:secret@localhost:5672/portal
```

4. Save and close the document.

5. Restart Portal:

```
$ sudo supervisorctl restart portal:
```

6. To confirm the changes have been applied, check /var/log/cantemo/portal/portal.log for the line:

```
portal - celery.worker.consumer - INFO - consumer: Connected to amqp://
portal@127.0.0.1:5672/portal.
```

Note: In Portal 1.6.4 and earlier releases, it has been reported that the installation of RabbitMQ may cause celerybeat to fail on startup. This is easily solved by changing the ownership of the ***/opt/cantemo/portal/portal/*** *directory to www-data:www-data and changing the permissions to 755. This should be fixed in later releases of Portal.*

9.4. Uninstalling RabbitMQ

Should you wish to remove the RabbitMQ message brokering service and restore previous functionality to Portal the process is equally straightforward.

1. Stop Portal and celery services (celerybeat and celeryd):

```
$ sudo supervisorctl stop portal:
```

2. Uninstall RabbitMQ:

```
$ sudo rpm -e rabbitmq-server
```

3. Uninstall erlang:

```
$ sudo yum remove erlang
```

4. Delete the RabbitMQ database:

```
$ sudo rm -rf /var/lib/rabbitmq
```

5. Remove the additional setting from the [celery] section in portal.conf (/etc/cantemo/portal).

```
BROKER_URL = amqp://portal:secret@localhost:5672/portal
```

   When RabbitMQ has been removed the [celery] setting should be:

```
[celery]
CELERY_ALWAYS_EAGER = False
```

6. Save and close portal.conf.

7. Start Portal and celery services:

```
$ sudo supervisorctl start portal:
```

With RabbitMQ safely uninstalled Portal and ScriptRunner should operate as before.

10. Release Notes

10.1. New Features

10.1.1. Support for Portal 2.0+

SR 1.6: Javascript refinements and UI enhancements to allow ScriptRunner to run in Portal 2.0+.

10.1.2. Support for RabbitMQ

SR 1.52: Promotes the use of Celery and CeleryBeat for internal messaging and includes support for RabbitMQ message brokering software to further accelerate background processes.

10.1.3. Improved Timestamp Responses

SR 1.52: Checking against a Timestamp value with the *after* condition now supports values with a time difference of +1 second for the result to be true.

10.1.4. Support for Tag Fields

SR 1.52: Includes support for the new Tag field type.

10.2. Fixed Issues

10.2.1. Streamlined Backend Calls

SR 1.51.3: Optimized to take advantage of new features in Vidispine 4.x with simplified backend process structure.

10.2.2. Trigger on Ingest

SR 1.51: Introduces a new Trigger on Ingest checkbox for all automations that test for new items.

10.2.3. Uninstall Conflict

SR 1.51: Removes a potential conflict with the celery process created by uninstalling ScriptRunner.

10.2.4. Portal 1.6 Compatibility

SR 1.5: Added support for significant changes in the way Portal 1.6 updates metadata.

10.2.5. Subgroup Fields

SR 1.21: Fields in Metadata Subgroups can now be used as triggers.

10.2.6. Empty Parameter Values

SR 1.21: Automations no longer fail when empty parameters are sent to a script.

10.2.7. Checkbox Arrays

SR 1.21: Each checkbox in a checkbox array is able to function as an independent trigger.

10.3. Known Issues

10.3.1. Custom System Fields

Custom system fields are not currently supported as valid metadata triggers.

10.3.2. Empty Lookup Fields

If you remove all of the entries from a Lookup field and pass the empty field as a parameter to an external script the value may not be blank.

10.3.3. Original_mimetype Parameter

The value of the original_mimetype system field value can sometimes be blank when parsing as a parameter on automations triggered by the creation of a new item. The original_mimetype field must be the pivotal condition when used in a combined Metadata Filter (see 4.8. Combining Metadata Conditions).

10.3.4. Multiple Responses

Using unsupported system fields (see 4.6.7. System Fields) to trigger automations on ingest and batch updates to collections will trigger duplicate responses erroneously. The new Trigger on Ingest checkbox should be used as the pivotal condition for all automations that test for newly created items (see 4.7. Pivotal Conditions).

11. Support

For queries, additional support and feedback, including feature requests, please contact: support@mmct.com.

12. MetaCode, Inc.

MetaCode, Inc is the development arm of Meta Media Creative Technologies, established in 2012, to solve a need for specialist applications and workflow tools. With MetaCode, MMCT is able to provide new opportunities for clients and share solutions with key technology partners. ScriptRunner™ is the first of several applications on the company's roster.

13. About the Author

Jonathan Eric Tyrrell is a freelance workflow consultant, trainer and technical writer. He has clients across North America, Europe and the Middle East. He works with vendors, broadcasters and media producers of various shapes and sizes to help define, enhance and implement post-production workflows. He has worked as an author in the creation of Apple Authorized Training titles including Optimizing Your Final Cut Pro System and is currently an Apple Certified Trainer for Final Cut Pro X and an Adobe Certified Instructor for Premiere Pro. You can read more of his writing about NLEs and asset management workflow at postpost.tv.