**Retrieving Images with Splunk using custom command**

Every once in a while my customers ask for a functionality that is not natively supported by Splunk. Out of the box Splunk is a very capable platform, however, there are certain things it’s not designed for. But that never stops a Splunker from finding a solution. The use-case I am about to discuss in this blog is an example of that. The customer owns large chains of pharmacies across the country. The bulk of stores transactions end up in a Hadoop Data Lake. The customer want to use Hunk/Splunk to visualize and analyze the massive amount of information collected, which is something Hunk can do easily. The challenge came about when I was asked if Splunk could show RX TIFF images (doctor’s hand written prescription) along side with patient’s records. I was presented with the following criteria:

-Retrieve patient’s records from Hadoop and marry them to RX image residing on an imaging server(s).

-The image server(s) is running apache tomcat.

-RX Images are stored in TIFF format (but can be in different format)

-Must be able to handle billion of images and accommodate error conditions (ex: files not found, failed communication, …etc.)

The Hadoop part was easy with Hunk. After all that’s what we do best! But dealing with the image piece required more work. Initially I thought, I can use the workflow built-in Splunk to retrieve remote URLs <http://docs.splunk.com/Documentation/Splunk/6.3.1511/Knowledge/CreateworkflowactionsinSplunkWeb>

But I soon discovered that I have four problems to solve:

1. Images are stored in TIFF format and most browsers don’t know handle it (with the exception of Safari).
2. We need to process (download) large amount of image files from multiple images servers. So performance and error handling is critical.
3. Retrieved images need to be annotated for added information
4. Retrieved images may need to be resized before displaying (if larger than certain size).

To address those challenges I turned to the power of custom search commands. Splunk Enterprise lets you implement custom search command for extending SPL (Search Processing Language). I wrote a search command (*getimage.py*) that will satisfy all of the above requirements. To demonstrate the usage of this custom command I also created a little app the can be found here <github link>.

**How does it work?**

The script will accept two arguments <fieldname> <url>. The first argument is the field name that contains the image name in raw data. The second argument is the remote imaging server URL (without the destination file name). We use the image file name to retrieve the file from the remote imaging server(s)

Example:

source="patients.records" | *getimage* image\_file http://10.211.55.3/icons

| table Patient\_Name, Prescription, image\_file, new\_image, wget\_result, link, cached\_image

The script uses the infamous *wget* command <https://www.gnu.org/software/wget/> to download images.

Once retrieved; we use “*convert*” command from a well-known image manipulation package called ImageMagick <http://www.imagemagick.org/script/binary-releases.php>

The “*convert*” command (not be confused with Splunk own *convert* command) is used to transform images format from TIFF to jpg (or png) then add any required annotation or resizing. Additionally, the script utilizes a caching mechanism to minimize the impact on the network. So if an image is retrieved repeatedly (within a pre-configured time value); it will be fetched from the cache directory (residing on Splunk Search Head). The *getimage.py* script will append several fields to the search results. Most of them are used for troubleshooting. Depending on how the dashboard xml is written you may want to use the “*link*” field or “*new\_image*” field.

Here is a list of fields injected into the search output:

[rc\_wget]: The output of wget command

[rc\_convert]: The output of ImageMagick’s *convert* command

[new\_image]: The converted image name

[image\_size]: Size of newly converted image (shows up when cached only)

[file\_loc]: \*Experimental\* location new image URI (using file:///)

[link]: URL of the new converted image on the Search Head

[cached\_imaged]: Indicate if image served from local cache or not

[wget\_result]: A cleaned up wget output

[app\_shortcut\_url]: location of all cached images for xml use

**How to test it?**

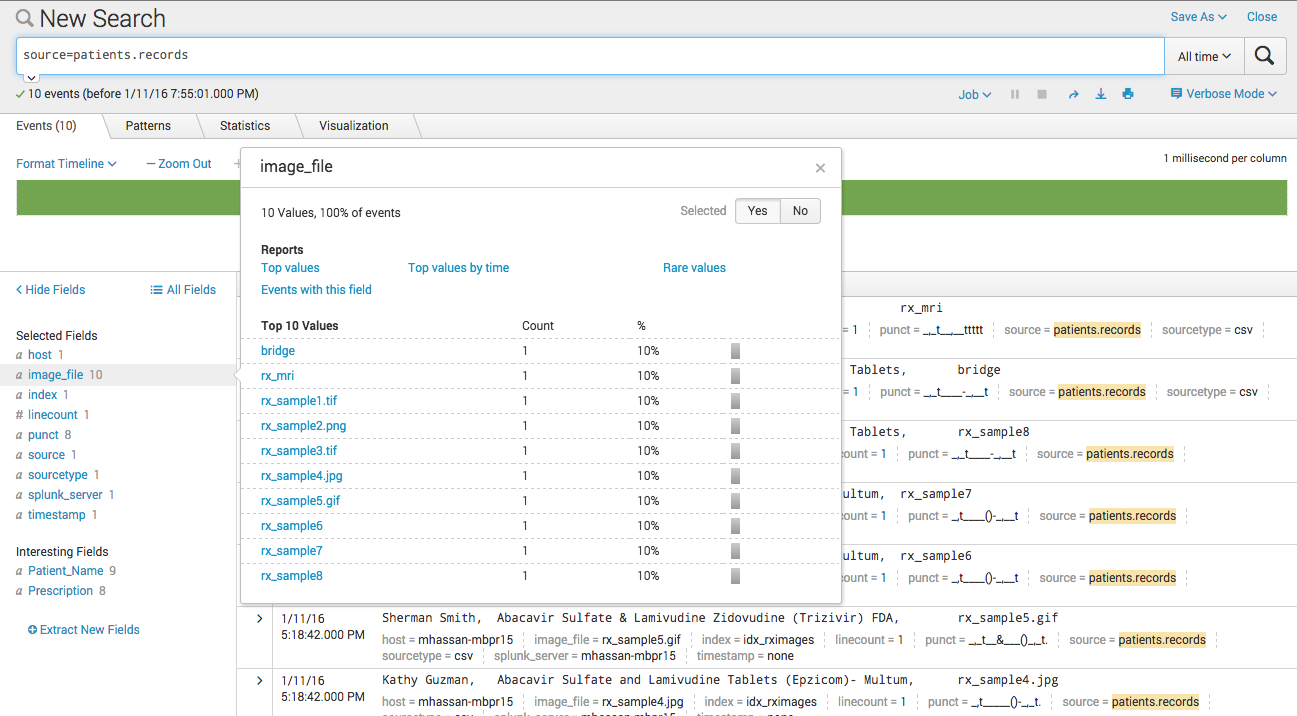
1. Setup a VM running apache server (my test VM is 10.211.55.3)
2. Install ImageMagick on your Search Head (interested in *convert* command only)
3. Install wget on your Search Head
4. Copy sample images (found in the apps sample\_images directory) to /var/www/icons/ directory on the apache server (default apache configurations for icons)
5. Manually import patients.records (under sample\_logs) into Splunk. Make sure you use CSV source type to get quick field extraction; otherwise you will have to manually extract the fields.
6. Verify the log import using Splunk UI. Most important field us is “image\_file”
7. Verify you are able to retrieve images manually using wget. Run this on the Search Head

wget --timeout=2 --tries=1 --no-use-server-timestamps http://10.211.55.3/icons/rx\_sample1.tif

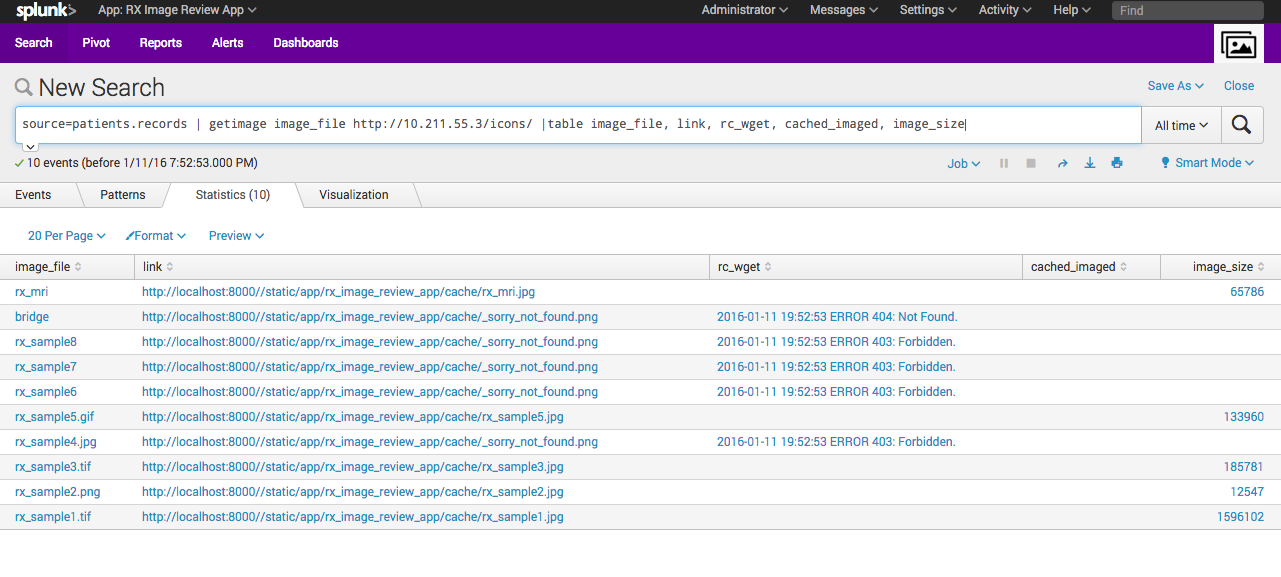
1. Verify you are able to use convert command (part of ImageMagick). Run this on the Search Head /opt/ImageMagic/bin/convert rx\_sample1.tif rx\_sample1.jpg
2. Modify *getimage.conf* to match your environment
3. The shipped app **rx\_image\_review\_app** should have the required files to make the script work (command.conf, authorize.conf)
4. To test connectivity kill httpd on the image server then try refresh your dashboard
5. To test 404 file not found error; remove one of the sample files from the image server
6. To test 403 (forbidden) error, mostly mean permission issues; change perms 600 on a sample file on the image server then connect again
7. To test images conversion, just convert an images to any of 100’s type the ImageMagick support on the image server; then refresh your dashboard
8. A log file to track activity is /opt/splunk/var/log/splunk/getimage.log

**Screenshots:**

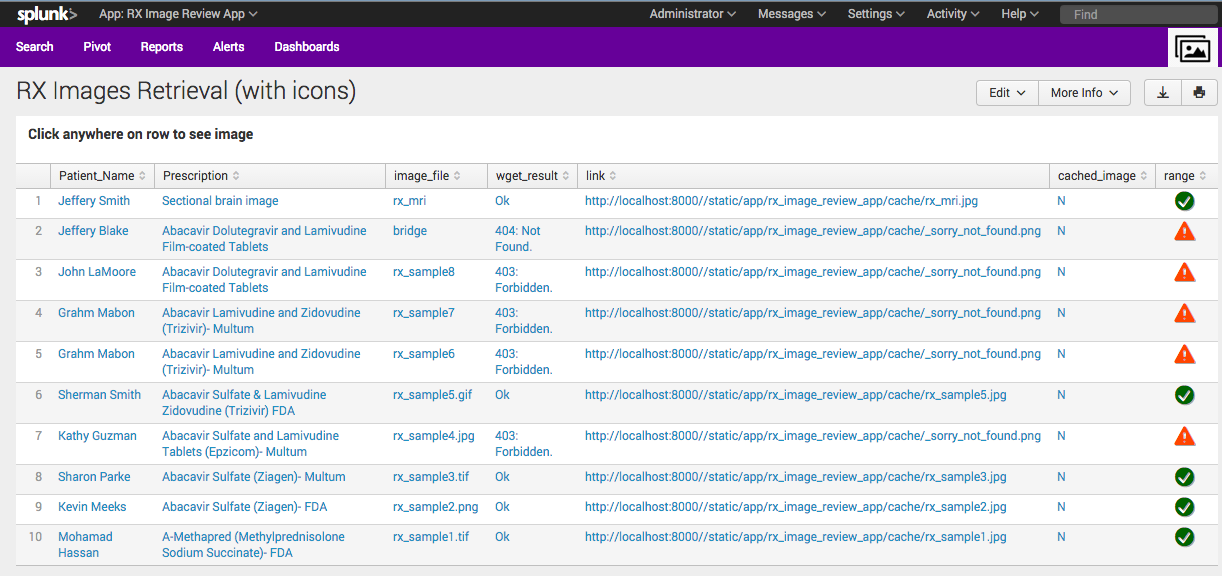
Here is the output from the sample input file (patients.records):



Here is an example of how the output will look like using *getimage* command:



Using the dashboard (RX Images Retrieval (with icons)



As you can see the script added additional field called “*link*” which is a link to the location of the image on the search head

**Script capabilities:**

1. Can handle images without file name extension.
2. Can convert over 100 major image file formats (ImageMagick).
3. Built-in caching (can be turned off).
4. Cached files do not linger around forever. There is a cleaning mechanism.
5. Dynamically handles many caching conditions scenarios
6. Multiple configurable parameters (set in a config file) for agile deployment.
7. Can handle network connection failures, 404 File Not found and 403 Forbidden errors.
8. Images are annotated before displaying (Image name, cached condition, and/or error conditions).
9. The script produces multiple fields that can be used for troubleshooting.

**What else can you do with this script?**

I wrote this custom command and created a showcase app in order to solve a specific problem for my customer. I am sure there are a lot more use-cases around images with Splunk. So feel free to borrow whatever you need to solve your problem. I attempted to document as much as possible of the code with the intention that someone is going to read, dissect and/or reverse engineer it. Here are few improvements you want to add:

* Remove dependency on shell commands (wget, convert) and use equivalent python modules (this may require additional packages imported into Splunk’s shipped python)
* Add JavaScript/xml to show images inline without having to open new browser
* Add more annotation to the images to communicate precise message to users
* In heavy duty retrieval scenarios, you can download images in bulk to speed up response time

**The App (rx\_image\_review\_app):**

A simple app was created to demonstrate how *getimage.py* could be used. The app is shipped with sample data and sample images files. I borrowed some xml and java script code (Table Icon Set-rangemap) from *Splunk 6.x Dashboard Examples app* <https://splunkbase.splunk.com/app/1603/>

I used the icons to give a quick visual feedback to the status of communication with the imaging server.

Here are the relevant directories starting from the main app directory **rx\_image\_review\_app**

appserver/static/cache all retrieved images are deposited here.

bin location of python script and configuration files

default authorize.conf and command.conf must have for script to work. Please note web.conf is for testing (turn of splunkweb caching)

/default/data/ui/view all dashboards xml documents

local props.conf and inputs.conf in case you need to use them

sample\_log sample input data patients.records

sample\_images sample images (need to copy to your imaging server)

Enjoy!