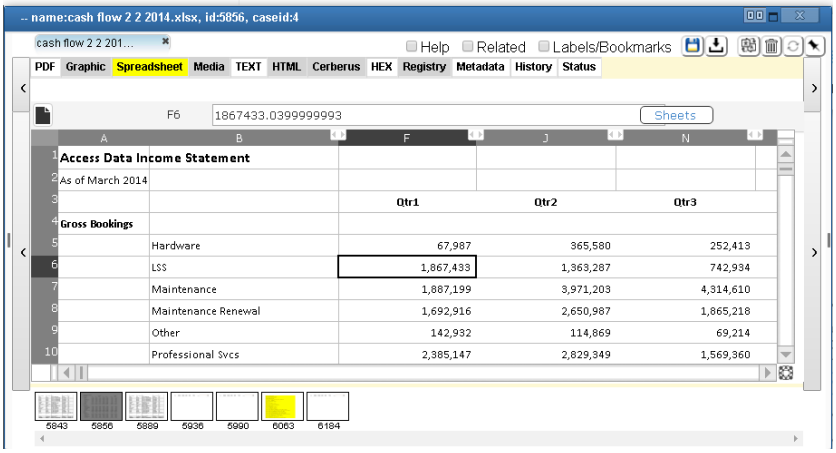
Viewing Documents In Quin-C:

The topic of viewing documents in Quin-C is the new bane of my existence. But not because Quin-C does it poorly, actually the problem is more because Quin-C does it so well. How can that be a problem you ask? Well the problem is people quickly get lulled into forgetting that there are fundamental limitations to viewing documents in a web browser. Then when they run into these limitations, it seems almost impossible to explain the various intricacies of web viewing and trade-offs available to users. It is exactly because of these intricacies that I am writing this blog. It is my hope that this can serve as a reference to Quin-C administrators so that they can tune the system to produce the best possible viewing experience for themselves and their users. So without further ado let me take you through both topic of document viewing in the web as well as how Quin-C's viewer works.

Lets start by discussing the generic topic of document viewing in the web. As should be obvious to anyone and everyone, an application that runs over the web is fundamentally different then one that runs on a desktop. A web application runs within the world of the browser and as such it is far more limited in what it can do than an application that runs on the desktop. Desktop application are essentially unconstrained and can do basically whatever they want. When viewing document desktop applications, such as FTK, take full advantage of the power available to them to parse, render and manipulate those documents so they appear exactly as they would have in the native applications. In many cases it is in fact the native application behind the scenes that is actually doing the rendering, which of course makes the rendering experience basically perfect. The other advantage that a desktop application has is that it is local to the data being viewed. There is no time lost to data transmission. In short a desktop application faces little to no obstacles when viewing data. If there is any challenge at all it is dealing with the one off odd file types, but these issues are on the margin and generally don't effect normal usage.

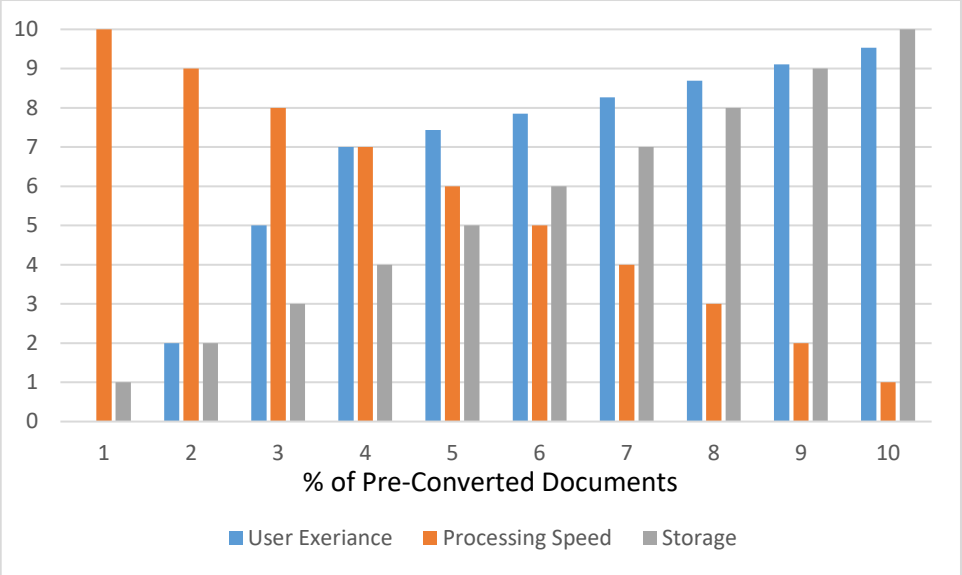
In the web nothing is that simple at all. The browser restricts what the application can do, and as a result of increasing security demands, they are getting stricter all the time. 3 years ago you could use flash viewers to display most document types but new browsers no longer support flash so now the options are far fewer. Basically modern browsers will allow you to do only two things: display the types they natively support (common graphic formats, some video options, text, pdf, and html) or write your own display technology in javascript. Since neither of these are great options Quin-C uses both. When the data is in a format the browser understands we let the browser render it. This is why the Quin-C viewer has so many different tabs dedicated to different types ike video, images, html, and text. When the browser doesn't support the underlying datatype, but it is easily converted into at format the browser does understand, then Quin-C converts the document and then sends it to the browser. Most of the time the converted format is pdf, since it is so widely accepted. However, pdf doesn't work for everything. Videos, archive files, audio, and other files don't convert to pdf well. For these datatypes Quin-C converts the data into the most logical format for the browser. For example videos and audios get converted to mp4 and mp3 respectively, while archive files are converted to navigatable html. However there are some file types that can't be rendered by the browser and that also can't be converted. Excel and database are examples of these. For these files Quin-C uses javascript base solutions that attempt to approximate the source application. This is why when using the spreadsheet tab or the database tab in Quin-C the user can actually interact with the documents and view the excel formulas or run sql queries. These are not capabilities that are normally possible within a browser but given the importance of these datatypes we felt it was essential to enable their viewing in Quin-C.

Spreadsheet Viewing In Quin-C



With these capabilities Quin-C can view a large variety of datatypes but not all viewing methods are the same and it is worth understanding the pros and cons of each. The best method is using the browsers native viewing capabilities. This method really has no costs. Documents are rendered quickly and well. The next best method is to write custom viewers for each file type. This is like what we have done for spreadsheets and databases. This method produces a great user experience but it has a very high development cost and frankly is impossible to do for a large number of documents. That leaves us with the final approach which is to convert documents into web view able formats. This method is the workhorse of most web solutions. It can be used for a huge number of file types and requires relatively little development effort. There are costs though and those are processing time and storage. Every document that is converted needs to be held both in its native form as well as its converted form so you are effectively doubling the storage requirements for each document. In addition, it of course takes time an cpu cycles to convert a document, which can meaningfully increase the overall processing time. These issues naturally leads people to want to reduce the amount of documents they convert. The problem with that is then you much convert documents on the fly when the user wants to view them. Depending on the document size and type conversion can range in time from one second to much longer. This can often create a negative user experience.

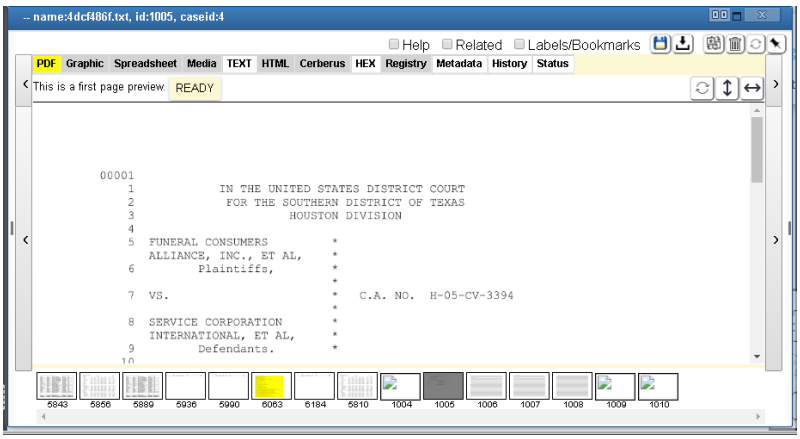
Taken all together these various factors create the basic relationships shown in the chart below. As the percentage of documents pre-converted increases the processing speed decreases in a linear fashion, the storage requirements increase in a linear fashion but the user experience increases in an asymptotic fashion. Assuming conversion of documents is done intelligently, with the documents that are likely evidentiary being converted first, then users generally see the impact very quickly. It is the goal of Quin-C to deliver admins the tool to hit that sweet spot where the users are happy but the percentage of documents converted is as low as possible.



To achieve that outcome Quin-C offers the following tools:

- The first of these tools is the first page viewer. Something like 90%(I made this number up but it definitely a big number in reality) of the documents that a user reads deal are only one page in size. The number is even higher if you start to consider the practical content of the documents. Quin-C understand this and offers users the ability to generate an image of just the first page of a document. This is powerful because these images take up minimal storage and can be generate very quickly. In addition because the files are small there is not issue with transit speed. Users get the feeling of instantaneous viewing but the system doesn't suffer from the storage or processing times associated with converting 100% of the documents.

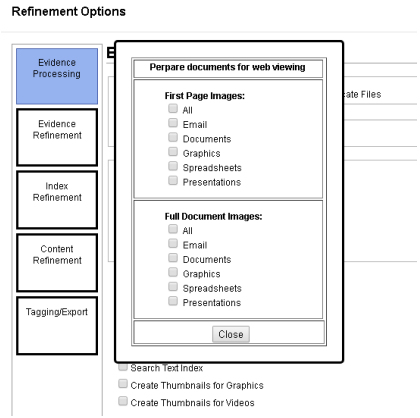
Viewing a First Page Image in Quin-C Gives a Positive Users Experience



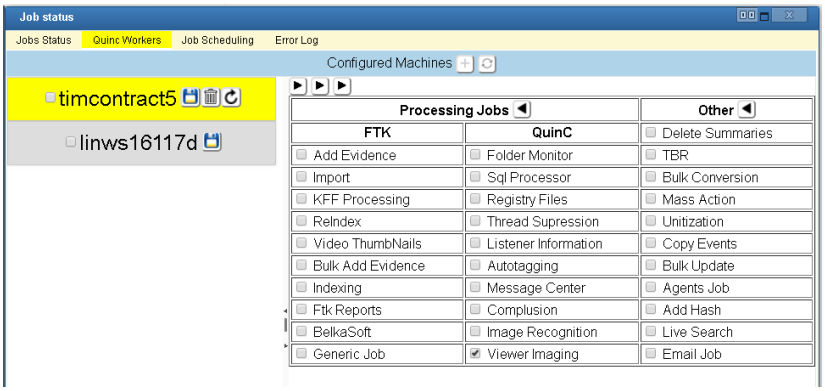
- The second tool is the option to pre-convert documents that users are more likely to view. Quin-C makes this delineation by document type. For example, it might make sense to convert word/office documents ahead of time during evidence ingestion. It is true that this will slow processing speeds marginally and increase storage requirements but by limiting the process to common user created documents the impact should be minimal but the improvement in the reviewers experiance should be significant. Pre-conversion can also be used for the generation

of the first page images mentioned in the bullet point above. It often makes sense to case a wide net with the pre-processing of first page images but be much more selective regarding the full document conversion.

Quin-C Processing Options



- Finally, Quin-C offers users the ability to manage the processing associated with document conversion. Which, machines are used for conversion, and how many resources those machines have allows the Quin-C Admin to fundamentally adjust the time and impact of document conversion. As the screenshot shows the admin can assign a worker to the dedicated task of converting documents:



While all of these measures taken together can produce a very powerful viewing experience, they still fall short of a desktop viewer. This is exactly why we have one last feature in development with a target release of September. Specifically we will be introducing an integrated desktop viewer into Quin-C. The viewer will allow you to download a desktop viewer, for windows OSes, that will seamlessly integrate with Quin-C and offer yet another way to view documents. This will not only dramatically expand the number of file types available for viewing but it will also eliminate the need for document conversion. With that final addition, Quin-C will have what we believe will be an untouchable document viewer that will make Quin-C the most superior web enabled document review platform.