
A Guide to Performance Analysis for ViewONE[®] and ViewONE Pro[®]

A DAEJA WHITE PAPER



Author: S. K Moss

Revision 1.0
28 June 2005.

© Copyright Daeja[®] Image Systems Ltd. All Rights Reserved.

Email: info@daeja.com

Web site: <http://www.daeja.com>

CONTENTS

Introduction	3
Understanding Performance	4
The Stages of Click-to-View	5
Start-up	5
Document Retrieval	10
Decompression and rendering	13
Summary	16
Avoiding & Diagnosing Problems During Click-to-View	17
Diagnosis Flow chart	17
QuickStart for ViewONE & ViewONE Pro	22
Deploying QuickStart	22
Using QuickStart	23
Switching Off QuickStart	23
Availability	24
How to	25
How to Install Java	25
How to Test Java itself	25
How to get Java console output	26
How to get the Java Engine version	27
How to take a screen shot	27
How to get the Windows Operating System (O/S) version	28
Dealing with class-not-found startup error	28
Switching on viewer network trace	29
Interpreting the viewer's network trace	29
Using separate thumbnail images	34
How to and why pre-install ViewONE-Pro	35
Using ViewONE Pro's text index feature	40
Using Javascript to remove start-up delays	40
Where to find viewer manuals and downloads	42
Summary of ViewONE Pro features that help with performance	42
Other areas to consider	43
Viewing on-line	43
Page separated documents	43
Separate thumbnails	43
Viewer controlled Pre-fetching	44
Compression technology and file format	44
Conclusion	45

INTRODUCTION

As with just about any piece of software, the time it takes to perform the job in hand is always an issue, even when it's super quick!

One might consider the issue of performance (in terms of speed of job completion) to be a fairly simple one, in principle at least. In practice however this is rarely the case because each user and each company often has unique requirements and constraints which in turn may be as a result of the specific application, company policies or attitudes.

The purpose of this document is therefore to try to provide a basis for analyzing, and in some general cases, identifying common causes of slowness rather than addressing the needs of a specific application. This document also tries to provide a background and understanding of some of the typical issues, the sorts of issues that ViewONE already addresses and some perhaps it may help address in the future.

For further information about ViewONE please consult the following documents:

- A Guide to Image-based Anti-Aliasing with ViewONE
- ViewONE Applet User Manual
- ViewONE Applet HTML and Installation Manual
- ViewONE Pro HTML and Installation manual
- ViewONE Applet JavaScript Manual
- ViewONE Applet Annotations User Manual
- ViewONE Applet Annotations Configuration Manual

or visit our web site at www.daeja.com

UNDERSTANDING PERFORMANCE

When analyzing performance issues for image viewing in particular there are a number of potential areas that can be looked at.

Essentially, as far as the end-users are concerned, performance will be the time taken following their selection of a document (probably by clicking on a 'view document' web link) to the time that document appears on the screen. This will include any necessary annotations/overlays and in the case of COLD document, all the necessary formatting including potentially an additional background template/form.

There are many steps from the 'first click' to the 'document view' ('click-to-view') and users simply do not (and should not) necessarily understand this. Consequently if the user feels that performance is inadequate, it is usually the viewer itself that is assumed to be the cause. In addition, Daeja's many years of experience have shown that it is often also assumed to be a viewer issue even by the 'integrators' or systems-engineers that are responsible for the back-end systems.

This experience has led Daeja to develop a logging system that permits analysis of each crucial stage. This logging system is uniquely embedded in the core product and automatically included in each installation. Ordinarily it is disabled since the actual process of logging can, in itself, affect performance, but it can be enabled easily by the user or systems engineers (we will run through this process later).

The entire click-to-view process can be summarized broadly by three main stages.

- Start-up
- Document retrieval
- Decompression and rendering

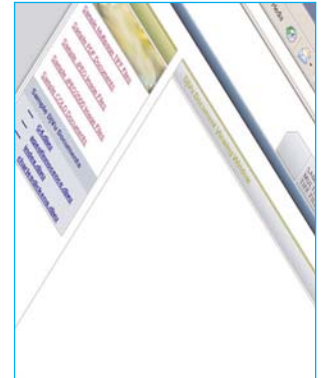
Each main stage has within it several steps and each can affect performance. To help understand performance issues within these stages the next section runs through each stage in more detail.



THE STAGES OF CLICK-TO-VIEW

Start-up

1. User logs on to ECM/Document repository system.
2. User enters search criteria to locate desired document(s), or selects a prepared set (batch) of document(s).
3. User chooses one of the documents then clicks on some sort of link to view the document.



- A new web page is generated that will ultimately contain the viewer. This stage could be rather complex (depending on the nature of the application).
- For example, many systems implement complex security that limits what each particular user, user group may or may not be able to do with a document. This may be as simple as preventing viewing or printing, to preventing (or allowing) viewing of specific pages, annotations and/or editing (or not) of a variety of those annotations.
- This stage may also involve construction of a list of URLs (one for each page of the document, one for annotations and potentially one for a COLD background template). Those URLs may be dependent again upon security impositions for that user, versioning or source of the document (ViewONE permits each page of a document to be sourced from any number of different servers).
- This stage may also trigger internal server processes that, perhaps, prefetch document pages or additional indexing information.
- The new web page may contain customized web graphics and document and user specific information

All the above processes would normally need to be completed prior to the viewer even being initiated as the processes are all engineered and produced by the ECM system itself.

THE STAGES OF CLICK-TO-VIEW (cont.)

4. If this is the first time the viewer has been used (for that users web session) then it may be necessary to start the 'Java Engine'. This is a piece of software that is responsible for 'running' Java applets (or any application developed using Java technology).

This stage once completed for the user's session, occurs once only for that 'start-up' speed of the Java engine can be dependent on a number of factors not least the available resources of the users machine (memory, CPU, disk space).

Daeja has observed that the Sun JRE (this is the usual Java engine used on new machine) is slower at this process than the Microsoft JVM (which until recently was shipped by Microsoft with all versions of its Internet Explorer browser). In some cases the Sun JRE may take as long as 5-8 seconds whereas the Microsoft JVM is normally sub-second.

It is important to note that once the engine has been started, this step is normally so quick so as to be unnoticeable.

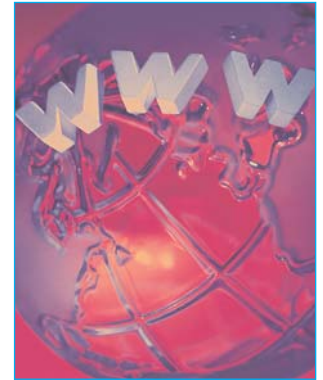
5. If this is the first time a specific version of the viewer has ever been used by the user, then this stage involves downloading the viewer itself. ViewONE is a client side viewer; which means that it performs its functions on the user's machine. This carries with it many advantages over 'server side only' systems; not least that there is no need for powerful and expensive servers. Client side viewers do not place rendering loads on servers (for any aspect of the viewer, including maintaining its basic user-interface graphics, menus etc) and so do not suffer from slowness of servers to handle rendering operations.

However, in order to be able to perform these operations client-side, the viewer itself must be downloaded and started by the users machine. This is an automatic process for applets and one of the key reasons for using Java technology - users do not need to be skilled in IT.



THE STAGES OF CLICK-TO-VIEW (cont.)

The download process will take some time, varying from seconds to potentially a few minutes, depending on the ability of the server to send files (in much the same way as it send web pages to the browser) and principally the speed of the connection to the server. ViewONE is around 1MB (much smaller than most other viewers), yet on a slow connection (say a slow modem) this could take a few minutes to download. On a typical LAN (high speed network) however this can however even be sub-second.



Once the viewer has been downloaded it is then 'cached' (temporarily saved) by the browser (or the Java engine if you are using the Sun Java Engine – 'JRE') so that this process is not required for future uses of the viewer. The viewer usually remains in the cache until, for example, a new version of the viewer is available (usually as a feature or maintenance update from Daeja). Under normal circumstances once cached for the user, this is the only time they'll notice this stage (see Cache note below).

The most important exception to this rule however is where a 'secure connection' is used with the server. This can most easily be seen by the presence of the letters 'https' within the browsers address bar, or by a padlock icon in the browser status bar. This indicates that all communication between the browser/Java engine/viewer and the server is encrypted (in order to hide communications from unauthorized eyes).

A condition of such secure links is also that any data (web graphics, text, objects etc) are removed from the client's machine once the web session has ended. This includes temporarily cached items such as applets (the viewer). If this is the case with your particular application then you'll be interested in how you can resolve this issue by using ViewONE Pro (discussed later in this document).

6. Once the viewer has been downloaded the Java engine then takes over. The first operation it will perform is to verify the integrity of the viewer itself (the program or 'archive file' that contains the program).

THE STAGES OF CLICK-TO-VIEW (cont.)

Any applet that wishes to perform I/O (file and server access) and/or printing operations must be 'signed' with a digital certificate that is provided by the applet vendor.

This digital certificate is used to verify the integrity through a complex process of encrypted checksums and keys. If there is any problem with the integrity (for example if it had been maliciously hacked or edited) then the applet will not be permitted to run.

This is a good safety net and one of the benefits of 'signed applets'.

This process again however can take a short time, usually sub-second but potentially longer depending on machine capacity and resources.

A NOTE ABOUT APPLLET CACHING AND THE START-UP STAGE

As mentioned above, applets are temporarily cached either by the browser or the Java engine. The very nature of caching means that from time to time caches will be purged. This is the process of removing unused or large cache files, usually when the cache reaches a predefined limit. Such a limit can usually be easily modified (through the browser preferences or Java preferences). However such modifications are rarely done as it requires a modicum of IT knowledge on the part of the user and that knowledge is usually beyond the typical user. Additionally, even if a user is able to increase the limit for purging, it can be impractical to expect this for users of large sites (e.g. those that run to 100s or 1000s of users).

ViewONE Standard already tries to address this issue in part, by maintaining its own caching area (to keep its own resource files and



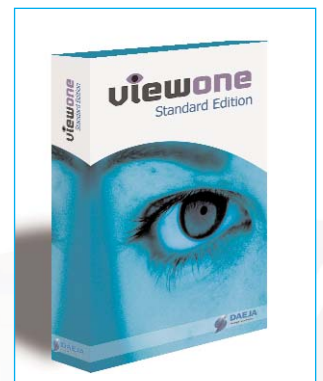
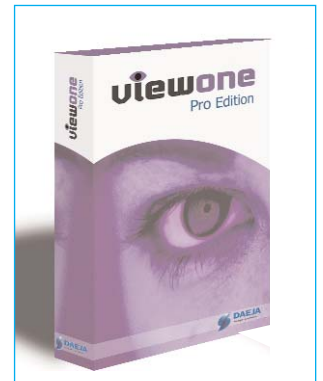
THE STAGES OF CLICK-TO-VIEW (cont.)

working temporary files). However, the applet itself may still be subject to the purging process, and as mentioned previously, where secure (https) links are used, that purging process forcibly takes place at the end of the secure session (usually when the user logs off the web site or a manual termination of the secure session). In such secure environments, any applets downloaded during the secure session are always removed.

The only solution to this is some form or pre-installation of the viewer. Now this is an area that is in direct conflict with the desire for a 'lightweight installation free viewer' (the essence of ViewONE). Until recently this has posed a real problem, especially where secure sessions are used. However, ViewONE Pro was specifically designed to tackle this head-on.

Unlike ViewONE Standard (and any other applet for that matter) the majority of ViewONE Pro 'sits' outside of these constraints. ViewONE Pro is structured so that the applet itself is only around 60k bytes (as opposed to the 1MB of ViewONE Standard). The applet is therefore very quick to download and check (in terms of the integrity checks above) and is of little consequence if purged from the cache. The major part of ViewONE Pro is actually cached outside of the browser's or Java engine's temporary storage area. This is performed automatically when the viewer is first used. This process also provides the user with a progress bar (which is not possible for normal applets) that informs the user when the download and caching process is occurring. This can be very useful in deterring the 'happy clickers' or impatient users (they are a reality that all software must try to deal with if possible).

In summary therefore, by resolving the caching issues, which ViewONE Pro does successfully, the entire start-up process and therefore related performance issues can be tackled head-on (see the 'How to' section for further details).



THE STAGES OF CLICK-TO-VIEW (cont.)

Document Retrieval

7. Most of the aforementioned processes occur independently of the viewer itself. Up until the latter phase of step 5, the viewer hadn't even been started. Now that it is started, the viewer takes over control of the remaining stages.



The first of these stages is for the viewer to perform its own integrity and start-up process. With ViewONE this is a pretty quick phase (usually sub-second) and involves verification of the license (to ensure that the viewer is only being used within licensed web-sites and servers) and the download of a few 'resource files'. These files amount to around 25K bytes of data, most of which is the 'localized language translation'.

Uniquely ViewONE offers automatic support for 20+ languages (with more being added all the time). During start-up, ViewONE auto-detects the user's particular machine setup and downloads the corresponding language files. These files are cached by the viewer itself so, again, this process only occurs once (unless a new viewer version is being downloaded). These files are cached 'outside' the browser's or Java engine's own storage area so that the viewer can ensure they remain cached even between web sessions.

8. At this stage, the viewer will now be visible to the user (with the user-interface showing its usual buttons, status bars etc).

The viewer will have also interpreted the HTML configuration parameters (generated during stage 3). These parameters will include the URL(s) for the document and annotations (if the annotations module is used). Many more parameters are available for ViewONE (for further information please see the "ViewONE Applet HTML and Installation Manual"). The number of parameters you use will not generally cause any noticeable delays but if you have any concerns please contact your support representative.

THE STAGES OF CLICK-TO-VIEW (cont.)

If the document being viewed does have annotations then the viewer will begin by retrieving the 'annotation definitions'. With ViewONE, this is a small amount of text based data and is retrieved before the document files themselves. If this process runs slowly then the viewer's inbuilt logging features will provide additional information which will help with diagnosis (please see diagnosis section below). For those technically minded readers, this is a simple "HTTP(S):GET" with retrieval of the full URL data.



If multiple URLs are defined for the document (one for each page), then the viewer will now issue a call back to the server to retrieve the file behind the first of those URLs. If only one URL is used for the document, then only one call back to the server will be made for the document files. For those readers who are still technically minded, this is another "HTTP(S):GET" with retrieval of the full URL data. This retrieval must take place before the process of 'rendering' the first page for viewing can take place.

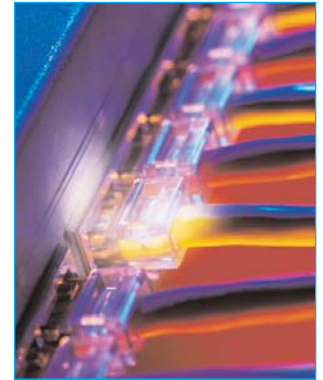
Ordinarily, ViewONE will display a progress bar at this stage (and associated help text) so that the user is aware the document is being retrieved. However, it should be noted that a progress bar can only be displayed (with percentage of completion), if the server had responded with the 'expected file size/length'. Daeja has encountered some cases where servers do not respond with this data, and so the user is often left unaware as to how this retrieval stage is progressing. This is easily dealt with and your support representative will be more than happy to provide specific advice on how to cure this issue.

The last item for this stage to consider is whether the document requires a background template to display. Such documents are classed under the umbrella of COLD documents (documents that have been split into two files, usually an image background - the form - and a foreground which may be text or image). The foreground will already have been retrieved and now the

THE STAGES OF CLICK-TO-VIEW (cont.)

background will need retrieving. This process is identical to the foreground retrieval, it's just another file required prior to view.

Now having already cached the viewer itself, this document-download stage is usually where most time is spent. This stage is dependent entirely on the ability of the server to send that file to the client's machine and the speed of the network connection. The more files that are required (i.e. annotations, COLD template) then the more opportunities there are for delays. If either of these areas causes delay, then the viewer can only wait until the document is available for rendering (if there are any problems here, the inbuilt logging system will help identify the cause).



A NOTE ABOUT ANNOTATIONS:

It is perhaps useful to understand how ViewONE differs with regard to annotations to other viewers on the market.

Usually annotation data is embedded with in the 'image file' itself (or in the case of PDF annotations within the PDF file itself). Daeja chose to keep annotation data separate from the image data itself, primarily to gain improved performance. If the annotation data is kept separate from image files then there is no need to 'send the entire image file' back to the server when annotations are either modified or created. The text data (which describes the annotations) is very small compared to the image data itself (and can be further compressed using zip algorithms – ViewONE will auto-decode zipped annotation data and re-compress prior to sending changes back to the server).

Additional benefits to keeping the annotation data separate from image data is a considerable saving on storage, since there is no longer any need to store image data with every change to annotation data. This in turn can lead to further performance improvements as well as significant costs savings.

THE STAGES OF CLICK-TO-VIEW (cont.)

Keeping annotation data separate also provides the ability to store such data in searchable databases, thus permitting uses to scan annotations without having to download images.

Finally, the separation of annotation data from image data, allows a much more generic approach to the issue of annotating documents as a whole. This effective abstraction of the 'annotation layer' from the 'image layer' permits ViewONE to offer annotation of any of the file formats which it supports. This list of formats currently covers the majority of the common formats out there (TIFF, JPEG, PDF, JPEG2000, BMP, GIF etc.).



However, Daeja is constantly adding support for more formats and, for example, is currently engaged in investing time and effort to adding Microsoft Office documents to that range. This is only truly possible by keeping annotation data separate from the original documents themselves and so the performance benefits described above will apply to any of these file formats.

Decompression and rendering

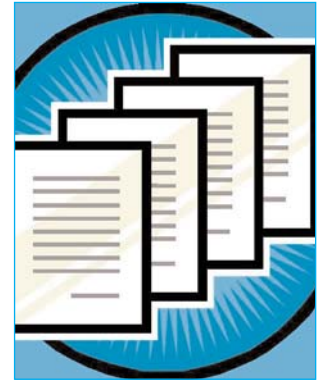
9. Once the document has been retrieved, the viewer sets about 'decompressing' the image data. Since mostly the viewer is used for viewing images, and those images are usually 'compressed' (so that their storage requirement is lower and transmission times to clients machines is kept to a minimum), they must first be decompressed before they can be viewed.

The viewer displays a progress bar for this process and is again usually subsecond. Where this may differ is for large color images (as they take longer to decompress) or where complex and high compression technologies are used (these require more CPU and memory resources to implement). If there are delays in this area, again the inbuilt logging will allow us to identify the cause here.

10. Once the image has been decompressed, the viewer will then 'render' the image for display. This process has been highly

THE STAGES OF CLICK-TO-VIEW (cont.)

optimized within ViewONE so that viewing and re-viewing (including rotating, panning, zooming) can be performed fluidly. When comparing client-side viewing software with server-side rendering software, this is where client-side viewers really demonstrate faster performance. This process should be transparent and the least incumbering on the whole process from click-to-view.



11. If the document only has one page, or the user chooses to view only one page, then this will be end of the viewing process for this particular document. If however, the viewer then wishes to view additional pages, the viewer will return either to stage 9 (retrieval) or 10 (decompression), depending on whether a separate URL (file) was specific for each page.

For example, for multi-page TIFFs (where all pages of a document are stored within a single file) then the viewer will return to stage 9 (decompression) for the next page selected by the user. There will be no need to return to the server to retrieve any more data and so one can expect a quick response while 'browsing' the document.

If however, a separate and different URL (file) was specified for each page, then the viewer will return to stage 9 (retrieval) in order to retrieve the (compressed) data for the next page.

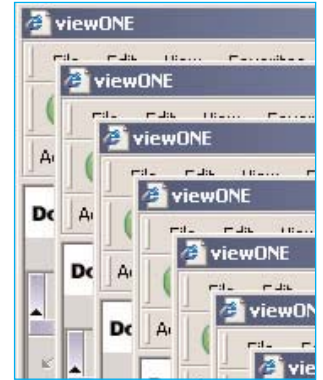
Which of these two systems you utilize will be application dependent and entirely your choice. The former (multi-page TIFF approach) yield better 'browsing' experiences for the user, whilst the latter (separate files for each page) yields quicker viewing of the first page.

It is generally accepted that the latter is the best approach for web-based systems, as it generally tends to place less load on the network (usually the slowest component of the system) and tends to generate the best perceived experience (in terms of performance) for the user.

It may be worth noting that if, for example, some or most of your

THE STAGES OF CLICK-TO-VIEW (cont.)

documents are of the multi-page TIFF variety, Daeja has made available (free of charge) source code for a servlet (that may be easily ported to other languages) that can be used to dynamically 'split' a multi-page file into its separate component pages. This makes it possible to obtain the benefits of page-separated documents whilst not having to physically separate/convert them in your document database (more on this will appear later in the document).



12. What happens next is highly dependent on your specific implementation. If the user returns to stage 3, to select another or new document to view, all the following stages may re-apply.

Assuming that the user returns to stage 3 only (and does not log out of the system or abort all browser sessions), then stages 4 and 5 (Java engine start-up and viewer caching) will not occur again.

It is typical however that your implementation will start a new instance of the viewer (for the new document), either in a new physical window, or by re-using the current viewer window. If the former approach is used (new window) then it will usually be left to the user to 'shutdown' any remaining and superfluous viewer windows. If the latter approach (re-use of the same viewer window) is used then the current viewer instance will be automatically shutdown and a new one started in its place.

It should be noted, that whereas in some cases the multiple-window approach is desired, it can have an affect on machine resources (especially when many windows are left open). Viewing images can be resource hungry (due to the need to hold image data in memory) and each window left open effectively locks-out resources, leaving less for new windows. This principle applies to all windows (not just viewer windows) but it can become intrusive, to the extent that performance can suffer, when leaving several image viewing windows open. The simplest and most common solution is to re-use the viewer window.

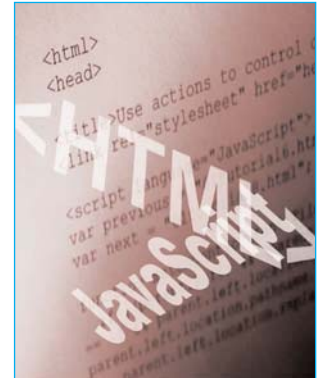
THE STAGES OF CLICK-TO-VIEW (cont.)

However, even that may not be ideal and can be further improved uniquely with ViewONE's comprehensive JavaScripting features. This is because the re-use of a viewer-window involves having to shutdown the current viewer instance and start a new one. This in turn involves re-checking the viewer instance (the start-up checks as described by stage 6 and the viewer start-up sequence described by stage 7).

There is a way to avoid both of these stages by implementing a combined HTML and JavaScript approach for subsequent viewings that removes the need to shut down the current instance but instead uses the same instance. This area is discussed in more detail in the diagnosis section.

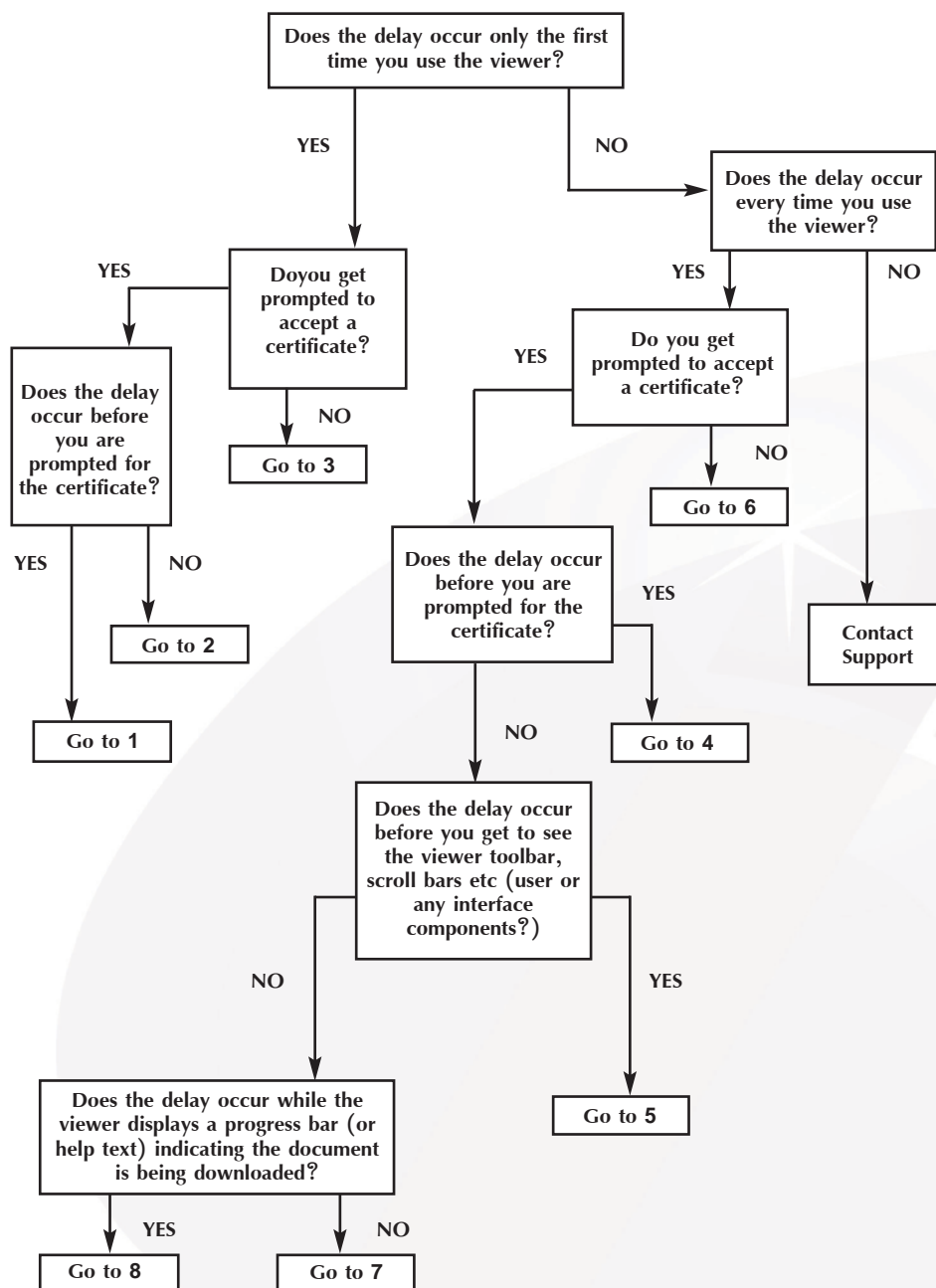
Summary

The above processes may look daunting at first glance, but do not be put off by the apparent complexities. Once a general appreciation is obtained that the click-to-view experience is predictive and re-occurring, then it becomes possible, especially using the viewers in-built logging to tune into where (if any) performance problems occur.



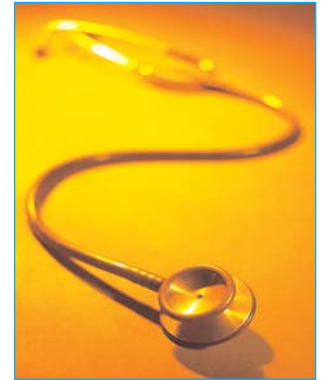
AVOIDING & DIAGNOSING PROBLEMS DURING CLICK-TO-VIEW

The previous section went through details descriptions of each stage of the click-to-view process. This section provides a method (by way of a flow chart) to help trace potential problems and makes suggestions to possible solutions. The basic premise is that there is a problem (otherwise you'd not need to use the flow chart!). The numbers in the chart refer to the numbered paragraphs overleaf.



AVOIDING & DIAGNOSING PROBLEMS DURING CLICK-TO-VIEW

1. This stage is the time required to download the viewer. This may take a few minutes on a modem, but would normally be a few seconds or up to 30 seconds on LAN or high speed LANs/T1/ASDL unloaded connections. You may be experiencing a slower connection or an over-loaded network problem, this may be a networking problem (you'll need to see your network specialist). If there is no solution to improving the network performance then you may benefit from a preinstallation of the viewer (see "ViewONE Pro to offer pre-installation"). You may also be experiencing a slow start-up of Java itself. This would normally be due to low resources (memory, CPU, disk, or fragmented disk, or slow I/O to disk). Once Java has started you should find this delay will not re-occur until next time you shutdown all browser instances and restart. This is normal, but it may indicate some tuning may be required of your machine.



We have since discovered, though rarely, some PCs are slow to process certificates. Once you see the certificate, make sure to tick the option to remember and accept the certificate. The next time you use it then you should not see such a long delay.

2. This stage is the time required to download the viewer. This may take a few minutes on a modem, but would normally be a few seconds to 30 seconds on LAN or high speed LANs/T1/ASDL unloaded connections.

You may be experiencing a slower connection or an over-loaded network problem.

3. It is probable that the viewer is not being started properly. This can be due to incorrect HTML (usually an incorrect codebase parameter) or a missing installation of Java. You can find HTML documentation in the ViewONE HTML installation manual at

AVOIDING & DIAGNOSING PROBLEMS DURING CLICK-TO-VIEW

<http://www.daeja.com/pub/start/manuals.html> and the Java installation software at <http://www.java.com/en/index.jsp>.

The easiest way to tell which problem you have is to try to open the 'Java console' (see instructions for "How to get Java console output" below). If the console will not open, it's likely you have an incorrectly installed or missing Java system. If it does start, then the console output will guide you to the problem (class-not-found errors indicate incorrect HTML codebase). See "Dealing with class-not-found startup error".



4. This stage is the time required to download the viewer. This may take a few minutes on a modem, but would normally be a few seconds to 30 seconds on LAN or high speed LANs/T1/ASDL unloaded connections. You may be experiencing a slower connection or an over-loaded network problem.

You may be experiencing repeat delays because the browser or Java caching areas are being purged or cleared (and therefore the viewer is forced to reload each time it is used). This may indicate you need to increase your cache size (see "Increasing you browser or java cache size").

It may also indicate that it is being cleared because you are using a secure connection (you'll see <https://> in your browser address bar and/or a padlock in the browser's status bar). The solution to this is a pre-installation of the viewer (see "Using ViewONE Pro to offer pre-installation").

You may also be experiencing a slow start-up of Java itself. This would normally be due to low resources (memory, CPU, disk, or fragmented disk, or slow I/O to disk). Once Java has started you should find this delay will not re-occur until next time you shutdown all browser instances and restart. This is normal, but it may indicate some tuning may be required of your machine.

AVOIDING & DIAGNOSING PROBLEMS DURING CLICK-TO-VIEW

We have since discovered, though rarely, some PCs slow to process certificates.

Once you see the certificate, make sure to tick the option to remember and accept the certificate. The next time you use it then you should see such a long delay.

5. This is the stage where the viewer needs to download its own resources. These resources are cached outside of the browser and Java engine temporary storage areas and so the delay should only occur once. If you are seeing repeat delays, then you need to switch on the viewer network trace logging. See “Switching on viewer network trace”. Send the trace output to your viewer support contact

6. It is probable that the viewer is not being started properly. This can be due to incorrect HTML (usually an incorrect codebase parameter) or a missing installation of Java. You find HTML documentation in the ViewONE HTML installation manual at <http://www.daeja.com/pub/start/manuals.html> and the Java installation software at <http://www.java.com/en/index.jsp>. The easiest way to tell which problem you have is to try to open the ‘Java console’ (see instructions for “How to get Java console output” below). If the console will not open, it’s likely you have an incorrectly installed or missing Java system. If it does start, then the console output will guide you to the problem (class-not-found errors indicate incorrect HTML codebase). See “Dealing with class-not-found startup error”.

7. You may find that the delay is due to the repeat start-up and re-startup phase, which in turn is due to the use of a different web page (that contains the viewer) for each document. This is an area that can be improved upon by the use of Javascript that enables automatic re-use of viewer windows (negating the need to start a new web page or viewer instance for each document). Please see “Using Javascript to remove start-up delays”.



AVOIDING & DIAGNOSING PROBLEMS DURING CLICK-TO-VIEW

Alternatively, please contact your viewer representative. To help diagnose other issues, some detail information about how to re-create the problem will help, along with the output from the Java Console (see “How to get Java console output”).

Screen shots always help (please see “How to get a screen shot”). Please see “General information required...”.

8. This indicates that the document itself is taking time to download. This may be due to a number of potential causes, including network delays (overloaded or slow network/low bandwidth connection), network DNS setup issues, delays by the server in responding to the document retrieval request, incorrect document URLs.

The first step in isolating the cause is to enable the viewer’s network trace feature (see “Switching on viewer network trace”). Once this is done, repeat the test (after restarting your browser) until the problem is replicated. The trace output will provide details of response times and transmission times (see “Interpreting the viewer’s network trace”). If you have difficulty in understanding the trace output, please contact your viewer representative for assistance.

If you are viewing thumbnails on a regular basis, and the server does not already generate small thumbnail images to assist, then it may be worth considering the possibility of getting the server to generate small separate thumbnail images (dynamically or pre-prepared). For those applications that utilize thumbnails on a regular basis this approach can offer significant gains, see “Using separate thumbnail images”.

Additionally, you may find a text index approach also assists (this is where the viewer displays an index, much like what is found in normal paper based documents) to assist with navigating large documents. This feature is available for free with ViewONE Pro. Please see “Using ViewONE Pro’s text index feature”.



QUICKSTART FOR VIEWONE AND VIEWONE PRO

The most significant performance improvements at viewer startup time are delivered through the use of the QuickStart feature for ViewONE and ViewONE Pro.

Traditionally when using any applet viewer, the majority of the first few seconds that it takes for the user to view the image is the launching of the browser followed by the Java engine, both of which are normally out of the control of a Java applet. Thanks to QuickStart, we are able to control these actions and reduce this time delay significantly.



QuickStart is available as a free update for all customers subscribed to our maintenance service.

For full information about Quickstart, please view the QuickStart manual available through the Daeja website at www.daeja.com.

Deploying QuickStart


Here are the simple steps to setting up the QuickStart facility within your deployment of ViewONE or ViewONE Pro:

- 1) Check to see whether or not the 'quickstart.htm' file is included in your 'v1files' directory. If it isn't, it is likely that you will need to update your licensed copy of ViewONE or ViewONE Pro
- 2) Download the relevant product update from our website at www.daeja.com and update your viewer implementation
- 3) Check to make sure that the 'quickstart.htm' file is within the 'v1files' directory
- 4) Add the QuickStart parameter to your viewer install code. The parameter is listed within the QuickStart manual, and is:
`<param name="quickstartURL" value="quickstart.htm">`
- 5) Make sure that the 'codebase', 'archive' and 'cabbase' parameters defined in the new 'quickstart.htm' file match exactly those within your main viewer implementation code
- 6) Deploy!

Using QuickStart

QuickStart is a client-side process and is not enabled by default - users need to switch it on themselves. Once again, this process is very simple:

- 1) Open the viewer
- 2) Open the 'Preferences' menu (e.g. right-click the mouse and scroll down to 'Preferences')
- 3) Select 'QuickStart' to switch it on.

The only difference that users will notice immediately is that the QuickStart icon () appears in their system tray. This illustrates that QuickStart is running and has been added to the PC's startup procedure. If the user checks the Preferences menu within ViewONE or ViewONE Pro they will see that QuickStart is now ticked, also indicating that it is running.

Depending upon where the viewer has been embedded, either the next time the ECM system is started, or when the user next logs off and back onto their machine, or when the PC is restarted, QuickStart readies the viewer for launch, allowing significantly improving the startup time.

Switching Off QuickStart

Once enabled by the user, QuickStart will continue to prepare the browser and the Java engine at system startup until QuickStart is switched off. Even when the users closes the browser window, ViewONE will still be running in the background. The session will not actually end until the user logs off the machine, switches it off, or switches off QuickStart.

There are two ways to switch off QuickStart. The first is to uncheck the tick within the 'Preferences' menu by selecting it again. The second is to right-click on the QuickStart icon in the system tray and select 'Stop ViewONE QuickStart'.

Availability

QuickStart is available for both ViewONE Pro and ViewONE Standard edition, and is available to all new customers, and within the update download for all those existing customers subscribed to our maintenance service.

HOW TO...

General information required for all issues that you report to your viewer support representative...

Prior to submitting problems to your support representative it will help first to gather some basic information. Typical information required is as follows:

- Java Console output
- Screen shot of the problem
- Explanation of how to re-create the problem
- Viewer version
- Operating System version
- Java engine version

The following paragraphs explain how to obtain such information and how to perform the typical operations while diagnosing potential problems:

How to Install Java

Java can be installed by visiting:

<http://www.java.com/en/index.jsp>

How to Test Java itself

If you are having problems running the viewer (or any applet for that matter) we recommend you run a test by opening the following web page (using the machine that has the problem):

<http://java.sun.com/applets/other/Bubbles/index.html>

HOW TO...(cont)

If Java is correctly installed on your machine then you will see an applet displaying bubbles (rising to the top of the window). If the applet is not displayed then we recommend you re-install Java (see above “How to Install Java”).

If you have already installed Java (and re-installed it to make sure!) then the cause is probably security related. Some machines are locked-down to the extent that even applet are forbidden, in which case you'll need to make sure the viewer is permitted by contacting your Systems Administrator (this is usually the person responsible for lock-downs).

You may find it useful to explain that the viewer is neither malicious or performs any malicious operations. The viewer is a commercial product for use by professional users!

How to get Java console output

In order to see any Java Console output you first need to make sure the Java Console window is available.

If you use Internet Explorer, select the ‘Tools’ menu and then ‘Internet Options’. Then select the ‘Advanced’ tab and scroll down the list until you see a line with either “Java (Sun)” or “Microsoft VM” (you may have both).

If “Microsoft VM” is ticked, then make sure the menu marked “Java Console Enabled” is also ticked. If it is not, you need to tick it then restart the browser. You will then find a menu “Java Console” menu item under the Internet Explorers “View” menu.

If “Java (Sun)” is ticked will then find a menu “Sun Java Console” menu item under the Internet Explorers “Tool” menu. If you are using another browser, then it is likely that you are using the Sun java engine. Your browser should have a menu that allows you to display the Sun Java Console.



HOW TO...(cont)

How to get the Java Engine version

To ascertain which Java Engine and version you have:

If you use Internet Explorer, select the 'Tools' menu and then 'Internet Options'. Then select the 'Advanced' tab and scroll down the list until you see a line with either "Java (Sun)" or "Microsoft VM" (you may have both).

If "Microsoft VM" is ticked, then close the "Internet Option" window, then click on the 'View' menu and select the 'Console' option. The version is shown at the top of the console screen.

If "Java (Sun)" is ticked it should also show the version within the same line. If it does not, then close the "Internet Option" window, and click on the "Tools" menu, then select "Sun java Console". The version is shown at the top of the console screen. If you are using another browser, then it is likely that you are using the Sun java engine.

How to take a screen shot

To take a screenshot of an error message or screen that you wish to send to your viewer support representative, you first need to recreate the error or bring up the screen which you are interested in.

Your browser should have a menu that allows you to display the Sun Java Console (and therefore Java version), but if not you need to select the Java Icon within your Control Panel (see Windows Settings menu). This will also show the Java version.

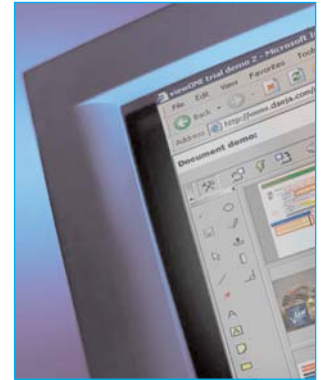
To take a screen-shot of the entire contents of the screen (includes all windows on your desktop), press the <Print Screen> (or <PrtScrn>) button on your keyboard.



HOW TO...(cont)

To take a screen-shot of just the relevant dialog box/window (displaying an error message for example), the first make sure that window as focus (by clicking on it), then hold down the <alt> key and press the <Print Screen> (or <PrtScrn>) button at the same time.

You will now have a screen-shot in your paste-buffer, which can be pasted into emails, Word documents, image editors, basically anything package than can display image contents of the past buffer.



How to get the Windows Operating System (O/S) version

Select the Windows “Start” Menu, then “Settings” menu, “Control panel” menu then “System”. The System dialog will display the O/S version and service pack details.

Dealing with “Class not found” startup error

If the viewer does not start at all, the first step you need to take is to look at the Java Console output (see instructions above).

The output generally gives some indication of the cause, one common cause is the message “Class not found”. This rather cryptic message indicates that the applet itself could not be located by the browser and/or Java engine.

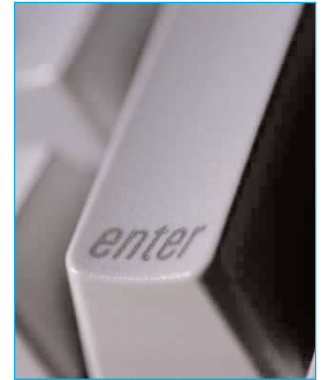
This will usually mean either the location that has been specified to the browser (see HTML Applet “codebase”, “archive” and “code” paramaters – described in the “ViewONE Applet HTML and Installation Manual”) are incorrect, or that the location specified does not actually contain the viewer.

The viewer itself is usually contained with a directory on the server named “v1files” (unless a different name was chosen when it was installed).

HOW TO...(cont)

To test whether the location has been specified correctly and whether the files are present and available at that location we recommend entering the full URL address for the directory and the file "ji.jar" to be entered into the browsers address bar.

Then when you press the Enter key the browser should prompt you to download this file. If it does not then some indication of the cause of the problem will be provided by the browser (such as file not found, invalid address etc).



Switching on viewer network trace

If you have access to the HTML source code, add the following line to the applet parameter list: `<param name="TraceNet" value="true">` (note that you should remove this once you have finished since the logging itself can affect system performance).

If you do not have access to the HTML source code, then with ViewONE running and showing an image, press `<Ctrl>` `<Shift>` and `<T>` together and then `<6>`.

All viewer network activity will not be logged to the Java Console.

Interpreting the viewer's network trace

When the viewers network tracing is switched on (see above) a series of logs will be sent to the Java console (see "How to get the Java console output). These logs will include time stamps and various summary descriptions of processes that take place. The key areas of interest will be the times at which the viewer attempts to connect to the server (say to retrieve an image file), the time it takes to obtain details about the file being retrieved (e.g the mime type and file length) and the time taken to download the file.

There will be other log entries for resource files downloads during start-up and various summary details that your support representative will use to get a better understanding of timings of

HOW TO...(cont)

events such as local file access as well as network access. On first glance the log often looks complex but it can be segmented into sections for easier understanding.

A typical output follows. This particular output starts after a user has viewed the first page of a 4 page document, and has chosen to view (for the first time) the second page:

User clicks next-page button...

```
-al> 28 Jun 2005, 13:06:08, GMT+01:00 (TIME: 0 / 0): -----Set page 2 withGen = true
-al> 28 Jun 2005, 13:06:08, GMT+01:00 (TIME: 16 / 16): -----Set page 2 done
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004437/000003594): 1119960368540> Net: SetPageA: 2
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004437/000000000): 1119960368540> Net: SetPageE: 2
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004437/000000000): 1119960368540> Net: SetPageF: 2
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004437/000000000): 1119960368540> Net: SetPageG: 2
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004437/000000000): 1119960368540> Net: SetPageH: 2
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004437/000000000): 1119960368540> Net: SetPageJ1: 2
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004437/000000000): 1119960368540> Net: THM SetPageA1: 2
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004437/000000000): 1119960368540> Net: THM SetPageA2: 2
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004453/000000016): 1119960368556> Net: THM SetPageA3: 2
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004453/000000000): 1119960368556> Net: THM SetPageA4: 2
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004453/000000000): File: Close :
C:\WINDOWS\Java\viewone\cache\x0e1003.tif
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004453/000000000): File: : Files open :
Tot=2/Read=2/Write=0
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004453/000000000): > Net: File is in cache:
C:\WINDOWS\Java\viewone\cache\x091006.jpg URL: http://myServer.com/docs/mixed/p2-t.jpg
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004453/000000000): File: : secRandomAccessFile r
:C:\WINDOWS\Java\viewone\cache\x091006.jpg....
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004453/000000000): File: : secRandomAccessFile r OK
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004453/000000000): File: : Files open :
Tot=3/Read=3/Write=0
-al> 28 Jun 2005, 13:06:08, GMT+01:00 (TIME: 32 / 16): -----IH: Set page 2 done
-al> 28 Jun 2005, 13:06:08, GMT+01:00 (TIME: 32 / 0): -----IH: Set page 2 tidy1..
-al> 28 Jun 2005, 13:06:08, GMT+01:00 (TIME: 32 / 0): -----IH: Set page 2 tidy done.
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004453/000000000): 1119960368556> Net: THM SetPageA5: 2
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004468/000000015): 1119960368571> Net: SetPageJ2: 2
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004468/000000000): File: Close :
C:\WINDOWS\Java\viewone\cache\x08b011.tif
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004468/000000000): File: : Files open :
Tot=2/Read=2/Write=0
```

The above logs summaries that steps the viewer takes to merely change to page two. This is prior to retrieving the image for that page. Most of the descriptions are here for the benefit of your support

HOW TO...(cont)

representative in helping to determine if any of the basic page changing features suffers delay. Ordinarily they would not be any delays here, and as can be seen from the time stamps (to the left of each log line) this process all takes place under one 10th of a second (it takes .004468 of a second = 4468 milliseconds).

Viewer checks its cache

```
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004468/000000000): > Net: File not in cache:  
http://myServer.com/docs/mixed/p2.jpg
```

Here the viewer checks its cache (to see if the image has already been retrieved for this document. It has not, and so it must retrieve the image from the server. The URL it uses is listed and one can see that it is a jpeg file.

Viewer connects to the server

```
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004468/000000000): > Net: Connecting to:  
http://myServer.com/docs/mixed/p2.jpg/true  
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004468/000000000): 1119960368571> Net: Waiting for connection...  
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004468/000000000): 1119960368571> Net: Waited for connection...  
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004500/000000016): 1119960368603> Net: Connection date:  
1078335936000  
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004500/000000000): 1119960368603> Net: Connection processed...  
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004500/000000000): > Net: Connection: retrieving content  
length...  
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004500/000000000): > Net: Connection: Length = 69185...  
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004500/000000000): > Net: Connection: retrieving content type...  
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004500/000000000): > Net: Connection state: true/image/jpeg -  
jpg/69185  
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004500/000000000): File: : secRandomAccessFile rw  
:C:\WINDOWS\Java\viewone\cache\x0db013.jpg....  
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004500/000000000): File: : secRandomAccessFile rw OK  
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004500/000000000): File: : Files open : Tot=3/Read=2/Write=1
```

The viewer issues a connection request to the server and retrieves content and length details. We can see that it is an image of type JPEG with a length of 69,185 bytes. If there are any delays during the connection phase then these logs will show the delay. Delays can occur if, for example, the network bandwidth is limited or the server is slow to respond. In this example this process takes 32 milliseconds (4500 - 4468).

The last few lines here show the local file location where the viewer will store (cache) the image (C:\WINDOWS\Java\viewone\cache\x0db013.jpg).

HOW TO...(cont)

Viewer retrieves image

```
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004500/000000000): > Net: Reading...
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004500/000000000): > Net: Read 2048 bytes (total 2048)
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004500/000000000): > Net: Read 2048 bytes (total 4096)
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004500/000000000): > Net: Read 2048 bytes (total 6144)
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004500/000000000): > Net: Read 2048 bytes (total 8192)
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004500/000000000): > Net: Read 4096 bytes (total 12288)
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004500/000000000): > Net: Read 8192 bytes (total 20480)
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004515/000000015): > Net: Read 16384 bytes (total 36864)
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004515/000000000): > Net: Read 32321 bytes (total 69185)
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004515/000000000): > Net: Read complete (69185 bytes)
```

The viewer begins retrieval of the image in small chunks (starting with 2048 bytes). The viewer monitors the time taken to retrieve each chunk. Over a period of time it gathers enough information to make an educated guess on whether the connection to the server is a good one (if it performs quickly), and if so it will increase the size of the next chunk, until it reaches a point where the time to retrieve the chunk exceeds a pre-configured time limit. Currently this is set to 500 milliseconds (please note this is subject to change).

Daeja has discovered that this approach to retrieval (using variable chunk sizes) offers a more flexible system that offers better performance for faster connections yet self adjusts for slower connections. If there are any network/transmission or server delays we will see the delays within these log lines.

In this example, we see the chunk size started out as 2048 bytes and grew to 32321 bytes. The image file was retrieved in 15 milliseconds (4515 – 4500). This indicates that the file probably came from the a server on a high speed connection (probably a LAN). This rate works out at over 43M bits per second which requires at least a 100M bit connection, so this network is operating quickly.

Viewer performs various read/write operations on the cached file

```
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004515/000000000): File: Close
:C:\WINDOWS\Java\viewone\cache\x0db013.jpg
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004515/000000000): File: : Files open :
Tot=2/Read=2/Write=0
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004515/000000000): Network read checksum: -7792
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004515/000000000): File: : secRandomAccessFile r
:C:\WINDOWS\Java\viewone\cache\x0db013.jpg....
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004515/000000000): File: : secRandomAccessFile r OK
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004515/000000000): File: : Files open :
```

HOW TO...(cont)

```
Tot=3/Read=3/Write=0
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004515/000000000): File: Close
:C:\WINDOWS\Java\viewone\cache\x0db013.jpg
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004515/000000000): File: : Files open :
Tot=2/Read=2/Write=0
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004515/000000000): Cached file checksum (69185 bytes) : -
7792 >
C:\WINDOWS\Java\viewone\cache\x0db013.jpg
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004515/000000000): > Net: Read finalized:
C:\WINDOWS\Java\viewone\cache\x0db013.jpg
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004515/000000000): File: : secRandomAccessFile r
:C:\WINDOWS\Java\viewone\cache\x0db013.jpg....
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004515/000000000): File: : secRandomAccessFile r OK
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004515/000000000): File: : Files open :
Tot=3/Read=3/Write=0
-al> 28 Jun 2005, 13:06:08, GMT+01:00 (TIME: 125 / 93): -----IH: Set page 2 done
-al> 28 Jun 2005, 13:06:08, GMT+01:00 (TIME: 125 / 0): -----IH: Set page 2 tidy1..
-al> 28 Jun 2005, 13:06:08, GMT+01:00 (TIME: 125 / 0): -----IH: Set page 2 tidy done.
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004546/000000031): 1119960368649> Net: SetPageK: 2
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004546/000000000): 1119960368649> Net: SetPageL: 2
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004546/000000000): 1119960368649> Net: SetPageO: 2
-al> ji.image.cy 28 Jun 2005, 13:06:08, GMT+01:00 (000004546/000000000): File: : secRandomAccessFile r
:C:\WINDOWS\Java\viewone\cache\x0db013.jpg....
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004546/000000000): 1119960368649> Net: SetPageP: 2
-al> ji.document.ic 28 Jun 2005, 13:06:08, GMT+01:00 (000004546/000000000): 1119960368649> Net: SetPageQ: 2
-al> ji.image.cy 28 Jun 2005, 13:06:08, GMT+01:00 (000004546/000000000): File: : secRandomAccessFile r OK
-al> ji.image.cy 28 Jun 2005, 13:06:08, GMT+01:00 (000004546/000000000): File: : Files open : Tot=4/Read=4/Write=0
-al> ji.image.cy 28 Jun 2005, 13:06:08, GMT+01:00 (000004546/000000000): File: Close :
C:\WINDOWS\Java\viewone\cache\x0db013.jpg
-al> ji.image.cy 28 Jun 2005, 13:06:08, GMT+01:00 (000004546/000000000): File: : Files open : Tot=3/Read=3/Write=0
-al> ji.image.cy 28 Jun 2005, 13:06:08, GMT+01:00 (000004546/000000000): Decompress file checksum (69185 bytes) :
-7792 >
C:\WINDOWS\Java\viewone\cache\x0db013.jpg
-al> ji.image.cy 28 Jun 2005, 13:06:08, GMT+01:00 (000004546/000000000): File: : secRandomAccessFile rw
:C:\WINDOWS\Java\viewone\cache\x009015.tmp....
-al> ji.image.cy 28 Jun 2005, 13:06:08, GMT+01:00 (000004546/000000000): File: : secRandomAccessFile rw OK
-al> ji.image.cy 28 Jun 2005, 13:06:08, GMT+01:00 (000004546/000000000): File: : Files open : Tot=4/Read=3/Write=1
-al> ji.image.cy 28 Jun 2005, 13:06:08, GMT+01:00 (000004671/000000125): File: Close :
C:\WINDOWS\Java\viewone\cache\x009015.tmp
-al> ji.image.cy 28 Jun 2005, 13:06:08, GMT+01:00 (000004671/000000000): File: : Files open : Tot=3/Read=3/Write=0
-al> ji.image.cy 28 Jun 2005, 13:06:08, GMT+01:00 (000004671/000000000): File: : secRandomAccessFile r
:C:\WINDOWS\Java\viewone\cache\x009015.tmp....
-al> ji.image.cy 28 Jun 2005, 13:06:08, GMT+01:00 (000004671/000000000): File: : secRandomAccessFile r OK
-al> ji.image.cy 28 Jun 2005, 13:06:08, GMT+01:00 (000004671/000000000): File: : Files open : Tot=4/Read=4/Write=0
-al> ji.image.cy 28 Jun 2005, 13:06:08, GMT+01:00 (000004687/000000016): false:Gen1 complete
-al> ji.image.cy 28 Jun 2005, 13:06:08, GMT+01:00 (000004687/000000000): false:Gen2 complete
```

HOW TO...(cont)

Here the viewer performs various operations on the cached file (C:\WINDOWS\Java\viewone\cache\x0a2013.jpg) and creates a temporary working file (C:\WINDOWS\Java\viewone\cache\x009015.tmp) during the process.

These log lines are essentially for your viewer support representative in helping to determine if there are obvious delays with the users PC file operations. The viewer often creates temporary working files during processing of images files, rendering and printing operations. Applets have limited access to memory and so in order to perform the kind of intensive operations demanded by imaging viewing and printing it often resorts to 'swapping-out' memory data to local files (to free up memory).

This is a normal operation and we can see the entire process takes 156 milliseconds (9312-9156), or a little over one 10th of a second. The image will now be displayed by the viewer indicated by the last two lines "Gen1 complete" (this is programmer speak for image rendering 'gen'eration complete).

Using separate thumbnail images

The viewer's thumbnail pane is usually used to obtain a quick overview of a document and to aid in selecting pages more quickly. In order for the viewer to be able to display thumbnails it must either create them using the original image files or it can retrieve image files already pre-prepared for the purpose.

If the thumbnail pane is used frequently and you are seeing delays because the viewer must retrieve the main image files first, then it is recommended that the thumbnail files either be generated dynamically on the server or are pre-generated on the server for such uses. Pre-generated thumbnail files are usually much smaller than the source image file, for example a typical thumbnail file might be only 1-15K compared to 30-70K for the full sized equivalent (depending on quality and image resolution). Unfortunately Daeja does not supply software for server side generation of thumbnail files (though it may be considered in the future) and so at present we recommend the use of one of the many commercially available imaging toolkit packages.

In order to use separate thumbnail files you will need to modify the HTML to include the "Thumb<N>" paramater. See "ViewONE Applet HTML and Installation Manual" section "Opening documents and images".

HOW TO...(cont.)

How and Why to Pre-install ViewONE Pro

Why pre-install?

Applets are contained in 'java archive files', also known as Jar's (Java Archives). These are basically compressed versions of the applet program. Applets that require access to files, servers and printing devices need to be 'signed'. This involves a process of including a digital certificate within the Jar file in such a way that it is tied to the integrity of the Jar file.

Digital certificates are then used (by the browser) to verify the integrity of the Jar file (at download time) and therefore the applet. A Jar file that fails that integrity test is deemed to be corrupt and potentially dangerous by the browser and the user will get some form of notification from the browser.

Additionally if the browser has not encountered a specific applet or certificate previously (certificates are supplied by the applet vendors), the certificate will be displayed to the user, so that the user can verify its origin and in so doing apply an additional level of protection. A user can always reject the certificate and in which case the applet will not be permitted to run.

For this process to work (and thus offer a good level of security for users against potentially malicious applets), the browser must download the Jar file. Unfortunately this requirement conflicts with another common requirement, which is to preinstall an applet, specifically to prevent it from being downloaded by the browser. Such a requirement usually stems from the need to keep network activity at a minimum, especially during the initial stages of rolling out applets to users on a web based network.

ViewONE Pro was specifically designed to deal with this issue by effectively separating out the majority of the product (and therefore out of the applet wrapper) so that nearly 95% of the viewer can be pre-installed. However, the approach that Daeja have taken has enabled the security provisions mentioned above to be retained at the same time.

ViewONE Pro retains an applet (of around 70K bytes) which is quick to download (compared to the ViewONE Standard 1MB applet) due to the much lower network overhead. This applet is solely responsible for maintaining security and for installing the remaining 95% of the applet. Additional benefits of this approach is that it also offers all the usual automatic updating features that applets are known for (by automatically updating installed versions

HOW TO...(cont.)

when they are released) while also adding the ability to control who and when such updates can be performed (unlike with typical applets).

It also resolves the issue of repeat downloads when using secure connections (HTTPS). When using secure connections, the browser will clear its temporary cache of any web objects (including applets) when the user ends the web session. This in turn forces the browser to re-download the applet upon next use. This in turn can cause unexpected delays due to the need to repeatedly download 1MB of applet every time prior to it being used. ViewONE Pro reduces this to 70K bytes.

How to pre-install?

ViewONE Pro offers two mechanisms.

Option 1:

If you are used to using ViewONE Standard, then the simplest method is to just swap ViewONE Pro in where ViewONE Standard is used. ViewONE Pro offers at least the same features as ViewONE Standard, and so users will not see much difference (unless you wish to open up additional ViewONE Pro modules to them), except that when they first use ViewONE Pro they will see a progress bar during start-up.

Unlike with typical applets, this progress bar is displayed to help notify the user that a download of an applet is occurring. In this case the download is the 95% of the applet mentioned above. The download will be installed into the installation path of the Java engine that the browser is using. The specific location will vary for different Java engines (and versions).

This entire process is automatic and does not require any changes to your system except a minor modification to the HTML that you had previously used for ViewONE Standard. This HTML is described in the "ViewONE Pro HTML and Installation manual".

Option 2:

In some cases it is not possible for users to be permitted to install programs (including applets). Users machine are sometimes 'locked-down' specifically to prevent such an occurrence, sometimes for security reasons and sometimes for administration reasons (where installs are controlled remotely and centrally).

HOW TO...(cont.)

Also it can sometimes be advantageous to perform installations during out-of-hours times or when the users machines is not being used.

To assist with both these cases ViewONE Pro comes with a mechanism to permit an installation of 95% of the applet prior to its use. This is achieved by using the 70K applet mentioned above but with specific parameters so that it just performs an installation rather than viewing a document. A sample web page (install-example.htm) is provided in the demo downloads (found at Daeja's web site www.daeja.com/pub/start/downloads.html - see Sample CGI scripts/"Example command line").

Simply open "install-example.htm" within your browser and the installation will take place, and once complete it will redirect the browser to your chosen 'completion' web page. In our example this completion web page displayed a message to the user and then closes the browser. This installation web page can also be run using any typical remote installation package (e.g Microsoft SMS system) using a simple command line which is executed on the users machine (using an account that has access to the "Program Files" and "Windows" directories).

An example of the command (if your users use Internet Explorer) is as follows:

```
"c:\program files\internet explorer\iexplore.exe" http://myserver.com/java/installexample.htm
```

If your users use a difference browser, then substitute the .exe file with the appropriate browser executable.

Note that this command instructs the browser to open a web page on your server. That's because the installation web page must be present with your viewer installation so that it can utilise the correct license and Jar files.

If you wish, instead, the installation to be perform by your users, perhaps during a set-up process prior to using the viewer then all you need to do is provide access to the installation web to those users.

If you want your users to perform the installation off line (e.g. by using a CD copy of the viewer) then please contact your viewer support representative so that a correct license file can be issued for that purpose. The HTML for the example installation web pages follow:

HOW TO...(cont.)

Install-example.html:

```
<html>
<head>
<META http-equiv="Content-Type" content="text/html">
<meta name="copyright" content="© 1997-2000 Daeja Image Systems Ltd">
<meta name="ROBOTS" content="NOINDEX, NOFOLLOW">
<base target="_top">
<title>viewONE trial demo 1</title>
</head>
<body bgcolor="#C0C0C0" text="#000000" topmargin="0" leftmargin="0">
<script LANGUAGE="JavaScript">
<!--      //this JavaScript sets the width and height of viewONE for all browsers
          //var archiveTag = 'ARCHIVE="viewone.jar, ji.jar, daejal.jar"';
          var archiveTag = 'ARCHIVE="viewone.jar"';
          var sizeTag;
          var appletHeight;
          var appletWidth;
          var leftToolbarWidth = 90;
          var topToolbarHeight = 30;
          var win = null;
          if (navigator.userAgent.toLowerCase().indexOf("mac_") > 0)
          {
              var archiveTag = 'ARCHIVE="viewonedsa.jar"';
              appletWidth = document.body.clientWidth - 40;
              appletHeight = document.body.clientHeight - 40;
          }
          else
          if (navigator.userAgent.toLowerCase().indexOf("msie") > 0)
          {
              //IE5.5 percents do not work on applets inside tables defined
              appletHeight = document.body.clientHeight - 30;
              appletWidth = document.body.clientWidth - 30;
          }
          else
          if ((navigator.userAgent.toLowerCase().indexOf("safari")> 0) &&
              (navigator.userAgent.toLowerCase().indexOf("ozilla")> 0))
          {
              //This is the only way we know how to set WIDTH and HEIGHT for Mac
              var archiveTag = 'ARCHIVE="viewone.jar"';
              appletWidth = window.innerWidth - 20;
              appletHeight = window.innerHeight - 20;
          }
          else
          if ((navigator.userAgent.toLowerCase().indexOf("macintosh")> 0) &&
              (navigator.userAgent.toLowerCase().indexOf("ozilla")> 0))
          {
              //This is the only way we know how to set WIDTH and HEIGHT for Mac
              archiveTag = 'ARCHIVE="viewonedsa.jar"';
              appletWidth = window.innerWidth - 20;
              appletHeight = window.innerHeight - 20;
          }
          }
```

HOW TO...(cont.)

```
else
if (navigator.userAgent.toLowerCase().indexOf("netscape6") > 0)
{
    appletWidth = window.innerWidth - 35;
    appletHeight = window.innerHeight - 35;
}
else
{
    //Netscape percents do not work on applets inside tables
    appletHeight = window.innerHeight - 35;
    appletWidth = window.innerWidth - 35;
}
sizeTag = 'WIDTH="' + (appletWidth - leftToolbarWidth) + '" HEIGHT="' + (appletHeight -
topToolbarHeight) + '"';
//-->
</script>
<script LANGUAGE="JavaScript1.1">
<!--
onerror=errorHandler;
function errorHandler()
{
//if we get here it is probably because a call has been made
//to the applet before the browser has had time to initialize it
//it can therefore be ignored
}
// -->
</script>
<div align="center"><center>
<script LANGUAGE="JavaScript">
<!--
    document.write('<applet CODEBASE="../v1files"');
    document.write(archiveTag);
    document.write('CODE="start.jiViewONE.class" NAME="viewONE" ID="viewONE"');
    document.write(sizeTag);
    document.write('HSPACE="0" VSPACE="0" ALIGN="middle">');
    document.write('<param name="cabbase" value="viewone.cab">');
    document.write('<param name="ACMPreInstall" value="true">');
    document.write('<param name="ACMredirect" value="../java/installed.htm">');
    document.write('</applet>');
-->
</script>
</center></div>
<align="center">
</body>
</html>
```

HOW TO...(cont.)

Installed.html

```
<html>
<head>
<title>ViewONE Pro has been installed</title>
</head>
<body>
<script LANGUAGE="JavaScript">
onerror=localErrorHandler;
window.close();
function localErrorHandler() {}
</script>
<p align="center">&nbsp;</p>
<p align="center">&nbsp;</p>
<p align="center"><font face="Arial">ViewONE Pro has been pre-installed on yourmachine.</font></p>
</body>
</html>
```

Using ViewONE Pro's Text Index Feature

ViewONE Pro introduced the option to display a text index (for multi-page documents). This is as an alternative to the thumbnail pane, which requires thumbnail images to be retrieved or generated from the main document images.

Where separate thumbnail files are not available the viewer must retrieve the main image files for each page so that it can create a thumbnail version. The retrieval and creation of thumbnails can take time (mostly spent during retrieving the files) and users have a tendency to become impatient.

Usually the reason for using the thumbnails pane is so that the user can get a quick idea of which pages are of interest (by viewing a smaller albeit less details version). The same objective can be achieved by browsing a documents index. ViewONE can be configured to present an index, which is basically an array of text lines with page numbers, and which in terms of retrieval is usually a considerably smaller amount of data than thumbnail or image data. Therefore using the index feature as an alternative is usually much quicker. The process of setting up a text index is described in the "ViewONE Pro HTML and Installation manual".

Using Javascript to remove start-up delays

Starting applets and shutting them down can be resource hungry and time intensive especially when performing these operations repeatedly. In many cases where the viewer is used, the

HOW TO...(cont.)

simplest approach to installation was chosen to keep each viewer instance in a separate web page. Therefore, each time the user selects a document for viewing, a new browser window is created and a new instance of the viewer started.

In most of these cases, before the user views another document, the previous window is closed (which causes the applet to be shutdown). In such cases there can be significant performance gains by avoiding the repeat start-up and shutdown processes by using a little JavaScript to manage viewer instance(s). This is an approach that Daeja recommends for all installations (so as to avoid any problems in this area in the first place) but the necessary changes can also be applied retrospectively.

In order to use JavaScript in this way there will need to be a change to HTML (that is ordinarily generated for the viewer).

The basic principle is to replace the direct generation of HTML (upon selection of a document to view) instead with a call to a JavaScript method, which then opens a new window and that contains the HTML that was previously generated in the old system. The JavaScript, prior to opening a new window, will check to see if a window is already open, then, if one is already open, it will re-use that window (and the instance of the viewer that is running in the window). Rather than issuing a server call to create new HTML it will, instead, call one of the viewer JavaScript methods to instruct the viewer where to find the new document (and any associated annotations).

The instance (that is already running in that window) will immediately close and open the next document. No shutdown and no start-up involved.

We have prepared a demonstration (including HTML, JavaScript, the viewer and sample documents) which can be found at www.daeja.com/pub/start/downloads.html (see Sample CGI Scripts/"Demonstration using JavaScript.."). You will also find details on all available viewer JavaScript methods in the "ViewONE Applet JavaScript Manual".

Where to find viewer manuals and downloads

All demo downloads (and sample scripts) can be found at www.daeja.com/pub/start/downloads.html

Manuals can be found at www.daeja.com/pub/start/manuals.html

Summary of ViewONE Standard features that help with performance

- Separate image files (see tiff splitter)
- Separate thumbnails
- Javascript for changing documents (to avoid repeat applet shutdown/restart)

Summary of ViewONE Pro features that help with performance

- All the above
- Fully cached applet outside of the browser/Java engine temporary storage
- On demand download of major features such a PDF, DjVu, JPEG2000
- Text index alternative to thumbnails

OTHER AREAS TO CONSIDER

The areas that we have not specifically covered so far in this document are arguably the most difficult to consider and are those areas that are very application specific...

Viewing on-line

Are your users in the habit of simply printing off documents in order to view them off-line without first checking or browsing them on-line? If so, you may find that encouraging users to browse documents on-line could have significant benefits.

Page separated documents

Due to the way ViewONE has been designed, and especially if multi-page documents are being viewed where each page is separately stored (or dynamically split using the free TIFF-splitter Daeja provides), then on-line browsing can significantly reduce network traffic and server demands compared to printing the whole document. ViewONE will only retrieve those pages that are required for viewing and statistically most users do not need or wish to view all pages. Of course this is subjective to the types of documents being viewed but none-the-less a useful consideration.

Separate thumbnails

An area worth consideration is the use of separate 'thumbnails'. These are smaller images that represent the original document. In some applications it is helpful to provide thumbnails as a way to help the user to further fine-tune their selection. ViewONE has inbuilt a feature that allows display and selection of thumbnails. If separate thumbnails are not available, then the viewer must retrieve the original document and create a thumbnail (which it does automatically). However, this would not offer any performance advantage. The advantage to be made is if the system can supply separate (and therefore smaller than original) thumbnail files. ViewONE will then retrieve those files only during thumbnail viewing, and only when the user wishes to view the selected thumbnail at full resolution will it retrieve the original (full resolution) file. The use of separate thumbnails (if thumbnail browsing is useful to the application) can improve browsing performance, reduce network and server loads (and therefore the need to have an expensive-specification server) and enhance the overall viewing experience substantially.

OTHER AREAS TO CONSIDER (cont.)

Viewer controlled Pre-fetching

ViewONE has the capability to pre-fetch images before the user views or prints them. This is primarily designed to assist in perceived response times when the pages of a document are separated into individual files. The prefetching capability can be configured to pre-fetch any number of pages ahead of where the user is currently viewing. This feature is also designed to work even if the user is apparently randomly selecting pages to view by attempting to judge the direction which the user is stepping through pages and pre-fetch either ahead or backwards accordingly. This feature can be useful in providing improved performance for the user because it effectively utilizes 'dead-time' while the user is busy reading a particular page. However, the benefits may depend on the typical manner in which users read through documents and the typical number of pages in a document.

Compression technology and file format

An area that has been the subject of much discussion over the years (and continues to be) is the one of your chosen image compression. This is the technology used to compress image files into as small a space as possible, and thus reduce storage and network loads and improve retrieval performance. There are many different compression technologies out there, some use open-standards and some are propriety. Some claim benefits for specific types of documents and some claim general benefits. To date, it is Daeja's view that no one compression or standard suits all. Daeja finds that some companies have policies that guide them to one or another specific standard but generally there remains no consensus as to which is best.

The choice of technology or standard can also affect the choice of viewer. For example, the least used (but perhaps the offering best compression) techniques and standards often require very specific viewer features. Such features will usually focus entirely on the decompression phase of viewing (some focus on document retrieval as well).

Daeja has demonstrated over the years that these two issues alone are generally not adequate in solving performance and cost issues around the whole click-to-view experience. A viewer must also adequately support the many requirements specific applications place on a viewer, not least, licensing costs, flexibility in integration options, configuration options, user-interface options, ease of installation and roll-out issues and on-going support efforts and costs. All these issues are tied up with cost-of-ownership which embeds the issues of performance and cost of performance.

CONCLUSION

This document has attempted to provide an insight into the issues surrounding performance for viewing images/documents. This document also provides a system and set of suggestions as to how to tackle each potential issue. Just as QuickStart has offered a solution to the issue of minimising start-up time, so Daeja will continue to develop innovative solutions to assist you in your efforts to optimize your ViewONE and ViewONE Pro implementation.

Daeja tries to focus on the whole picture and hopefully you will see that ViewONE achieves many goals in each area. There will always be areas that can be improved upon and this is the reason that Daeja continuously researches and strives to make improvements. New technologies are also always around the corner, and through the modular design approach employed by Daeja with ViewONE Pro, Daeja is able to continue to incorporate them. ViewONE Pro is also an example of how a viewer can offer significant benefits such as universal annotations (of any of the supported file formats including PDF) whilst still keeping an eye on the issues of cost and performance.

Once again, Daeja believes there is rarely just one issue when considering performance, and we hope this document has helped you focus on what we see as the main issues.

Please feel free to contact Daeja to discuss any topic and any concerns or questions at [**support@daeja.com**](mailto:support@daeja.com).



Daeja Image Systems Ltd

London House
High Street
Stony Stratford
Milton Keynes
Buckinghamshire
MK11 1SY
United Kingdom

Tel: +44 (0)1908 563007
Email: info@daeja.com
Web: www.daeja.com
