ViewONE Pro Streamer Manual

Version 1.1

Last Updated: 22nd March 2013

Email: info@daeja.com
Web site: http://www.daeja.com

Contents

Version 1.1 ..... *i*

Email: info@daeja.com ..... *i*

Contents ..... *ii*

Introduction *4*
Document Streaming 4
Thumbnail Generation 4
Combining Streaming with Prefetching 4
Performance and Server Resources 5

Installing the ViewONE Pro Streamer *6*

Configuring the ViewONE Pro viewer for use with the Streamer *7*
Modifying the applet definition 7
StreamerEnabled 7
setStreamerEnabled(true\false) 7
StreamerURL 8
setStreamerURL(url) 8
prefetchPages 8
prefetchThumbs 9
obfuscateUV 9

Advanced Filename configuration *10*
filename 10
Simple filename configuration 10
Advanced filename configuration 12
cookieAllowedDomainList 13

Configuring the ViewONE Pro Streamer servlet *14*

The streamer.properties file **14**
logFile 15
logFileSize 15
logFileCount 16
resourcePath 16
workingPath 16
docURLPrototype 17
docIdPrototype 18
sessionCookieList 19
sessionAuthenticationTimeout 19
signInResource 20
serverSupportsHeadRequestMethod 20
modifiedDocumentCheckMethod 21
modifiedDocumentMinimumCheckTime 21
documentExpiryTime 22
poolInitialEntryCount 22
poolMaxEntryCount 22
cacheRoot 23
cacheEmptyOnRestart 23
cacheSizeLimit 23

cacheAgeLimit ... 24
openDocumentWaitTime ... 24

Using the ViewONE Pro Streamer servlet for Pre-caching. ... 25

Pre-cache Server Configuration ... 25
maxPreCacheWorkers ... 25
maxPreCacheWorkerAge ... 25

Pre-cache Authentication and Authorization ... 26
authentication ... 26
allowedIPAddresses ... 27
authentication.handler ... 27

Pre-cache : 'default' Authentication Handler ... 28
authentication.postParameters ... 28
authentication.signInResource ... 28
authentication.loginURL ... 28
authentication.loggedInURL ... 29

Pre-cache : 'Workplace' Authentication Handler ... 29
authentication.userId ... 29
authentication.password ... 29
authentication.base ... 30
authentication.originPort ... 30
authentication.appId ... 30
authentication.signInResource ... 30

Pre-cache Client Parameters ... 31
precache ... 31
pages ... 31
thumb ... 32
thumbsize ... 32

Pre-cache: Example Request ... 32

Troubleshooting ... 33
Clearing temporary working files ... 33
Clearing the Streamer's document cache ... 33
Logging ... 33
Access denied error ... 33
User logged out error ... 33
Internal server error ... 34
No cookies supplied on request ... 34
java.lang.InternalError ... 34
Server Response Code 401 ... 35
IO error: Connection timed out ... 35
Platform compatibility notes for server side code ... 35

Appendices ... 36

Installing the ViewONE Pro Streamer trial ... 36
Apache Tomcat installation ... 36
Installation on other Web Application servers ... 36

web.xml file ... 38
Sample web.xml file ... 38

streamer.properties file ... 39

Introduction

Document Streaming

The ViewONE Pro Streamer processes both multi-page TIFFs and multi-page PDFs on the server and with version 4.0.2, this functionality was extended to include MS Word and MS Powerpoint files.

The Streamer sends only the data for the pages requested by the viewer. By only sending pages on viewing/printing demand, this significantly reduces the amount of network data transferred, which in turn speeds up viewing considerably and lowers network traffic for all users.

In the case of PDF files, the Streamer further reduces the amount of network traffic by performing (patent-pending) optimisation of PDF information. This can have a significant impact by saving up to 50% in network activity.

The Streamer is designed to minimise load on the server in order to be truly scaleable for even the largest of user groups.

The Streamer also encrypts all data it sends to the client machine so adds an additional level of security for any documents sent to the client machine via the Streamer.

Note: Whilst the Streamer can currently only stream TIFF, PDF, MS Word and MS Powerpoint documents, it can also act as a pass-through proxy for other file types that cannot be streamed.

It should also be noted that MS Office streaming will ONLY work where the streamerURL approach has been used. The advanced filename configuration approach will NOT work for streaming of MS Office documents.

Thumbnail Generation

The Streamer can generate separate (smaller) thumbnails images on-demand and then cache them for future use.

Note: Thumbnail generation does require more server CPU resources than for ordinary streaming. Please see the section on platform compatibility for support of this feature.

Combining Streaming with Prefetching

ViewONE already has an option to prefetch pages (see HTML manual, "PrefetchPages"). This feature is designed for use when pages of a document can be sent to the viewer separately, as in the case when this Streamer is used; ViewONE reads-ahead as a user browses a document.

Performance and Server Resources

Predictions for runtime resource usage during document streaming vary significantly based on the types of documents that are being processed. For example, large PDF documents with 10 or more pages require more CPU and disk usage than two page TIFF files.
The Document Streaming Server Module requires a minimum of 20 MB of memory for a single user. Per additional concurrent user, the memory requirement increases. The increase is dependent upon the type of documents that are being streamed. Typically, TIFF documents incur a 700 KB or more per-user increase and a complex PDF document could incur a 5 MB or more per-user increase. Real-world usage figures are likely to vary between, and occasionally greater than, these values. Concurrent user access can be limited by configuring some of the advanced options of the Document Streaming Server Module, options that are documented in the manual available at the
Daeja website location specified at the end of this section.

When you are sizing resources for streaming, remember that the Document Streaming Server Module does use caching. After a document's page is processed and is available from the cache of the Document Streaming Server Module, no additional CPU usage is required to serve the page to another client that requests the same cached page.

Disk space requirements are dependent on the size of the streamer cache and expiry options. The default size is 100 MB. This is a soft limit and it is dependent on the cache expiry check time that has a default of 1 hour. These defaults can be changed by editing the configuration options described later on in this manual.

Installing the ViewONE Pro Streamer

If you are installing the trial version of the ViewONE Pro Streamer, please see the "Installing the ViewONE Pro Streamer trial" section in the Appendices at the end of this document.

If you are upgrading the Streamer from a previous version or from the trial version, please refer to the section "Upgrading the ViewONE Pro Streamer".

Without the Streamer installed, the client will normally request a document directly from a document store via a URL. The document store will then send the requested document back to the client for viewing.

The Streamer requires a change to this behavior. It has to be configured to sit between the ViewONE Pro client and your source document store. Once this is done, the client will request a document by way of asking the Streamer. The Streamer will then contact the original document store, and request the source document and then stream the requested document back to the client.

Before you begin, you will need to make sure that a web application has been set up for the Streamer. For details on how to set up a new web application, please refer to your web application server's manual. Briefly, this consists of creating a web application root directory, (that might be called **"**streamer**"** for example) which in turn contains a "**WEB-INF**" directory which should house a "**web.xml**" deployment descriptor file. Alternatively, you can install the ViewONE Pro Streamer trial which is supplied as a standard web application directory. For information on installing the trial, please see the "Installing the ViewONE Pro Streamer trial" section in the Appendices at the end of this document.

To install the ViewONE Pro Streamer servlet into your web application server, you must first download the 'Update' zip (v31update_server.zip) from the Daeja web site. This can be found by visiting www.daeja.com/support/login.asp

After unzipping the downloaded file you should have a "**readme.txt**" file providing a high level overview of the files supplied. You should also find a "**WEB-INF**" directory (containing a "**lib**" directory).

1. Locate the **WEB-INF/lib** directory in your target web application server.

2. Copy the **jiserver.jar** file found in the downloaded "**WEB-INF/lib**" directory to the same place in your target web application server.

3. Copy your license file (**lic-server.v1**) into the web application server's **WEB-INF** directory.

4. Now create or modify the web.xml and streamer.properties configuration files (see the information on these files below).

Configuring the ViewONE Pro viewer for use with the Streamer

The changes required in your viewer HTML are minimal although note that older versions of the Streamer (Pre X.1.184) required mandatory changes to be made to the `filename` and `cookieAllowedDomainList` parameters (detailed below).

Since version X.1.184 of the Streamer, these changes are no longer necessary since the `StreamerURL` parameter now takes care of the construction of the URL to include the streamer and assembles all available cookies for inclusion with the request.

Further performance enhancements can be gained by using the prefetching features of ViewONE (see below).

Modifying the applet definition

It is important that the "mayscript" applet element is present in the `<applet>` definition so that the client can correctly obtain the browser's cookies. If it is not, you will see errors in the client's Java console and you will find that the Streamer cannot stream documents from a document repository that requires a login. The correct way to specify the `mayscript` parameter is as follows:

```
<applet mayscript="true" name="ViewONE" ...
```

It is not sufficient to use the `<param name="mayscript" value="true">` method.

Parameter: **StreamerEnabled**

(Version Standard 3.1.184+, Pro 1.1.184+)

<PARAM NAME="StreamerEnabled" value="true">

This parameter, when set to 'true', will cause all HTTP requests for documents\images sent from the viewer using either HTML parameters or Java Script methods to be directed via the specified streamer URL (see below).

The default for this parameter is 'true' where the "StreamerURL parameter has been defined (see below).

Method: **setStreamerEnabled(true\false)**

(Version Standard 3.1.184+, Pro 1.1.184+)

E,g, ViewONE.setStreamerEnabled(true);

The setStreamerEnabled() is the Java Script equivalent of the "StreamerEnabled" parameter (above) and this method allows the streamer to be enabled and disabled dynamically.

Parameter: **StreamerURL**

(Version Standard 3.1.184+, Pro 1.1.184+)

<PARAM NAME="StreamerURL" value="http://www.myWebApp.com/Streamer">

This parameter, when set to the location of the installed streamer will automatically append the URL of any file requests to ensure that they are directed via the streamer and that any authentication cookies supplied with the request are passed onto the streamer and then the image repository in turn.

Method: **setStreamerURL(url)**

(Version Standard 3.1.184+, Pro 1.1.184+)

E,g, ViewONE.setStreamerURL(http://myStreamerHost/Streamer);

The setStreamerURL() is the Java Script equivalent of the "StreamerURL" parameter (above) and this method allows the streamer location to be set dynamically.

Important Note: When opening a document via JS, the setStreamerURL(String url) call will need to be made before each call to open a document in the same way as the setAnnotationSource(String file) method.

Parameter: **prefetchPages**

<PARAM NAME="prefetchPages" value="2">

When this parameter is set to any value above zero, ViewONE will read-ahead the number of pages specified after the current page has been viewed. Depending on the specific needs of the user and the network setup, this read-ahead mechanism can help improve document browsing speed.

The value corresponds to the number of read-ahead pages, in the direction that the user browses a document. So, for example, if the user views page 1, ViewONE will read-ahead pages 2 and 3 with the above value.

If the user then views page 2, ViewONE will read-ahead page 4 (it has already read-ahead page 3). Page 3 will appear quickly since it had already been read-ahead while viewing page 1.

If the user then views, say, page 10, ViewONE will read ahead pages 11 and 12, on so on.

If then, finally, the user views page 8 (from say page 10), ViewONE will read-ahead pages 7 then 6, i.e. backwards because the user has changed their “direction” of browsing the document.

This process can improve viewing for browsing but it must be noted that it may also increase the network and server load. The benefit of pre-fetching depends on the type of

browsing usually performed by users, the network speed and the ability for the server to keep pace with such read-ahead mechanisms.

If the value of the parameter is set to “-1”, then all pages in the document will be pre-fetched. Note that this setting should be used with caution as it will certainly increase network and server load (especially on large documents with many pages).

Parameter: **prefetchThumbs**

<PARAM NAME="prefetchThumbs" value="2">

When this parameter is set to any value above zero, ViewONE will read-ahead the number of thumbnail pages specified after the current thumbnail has been viewed. Depending on the specific needs of the user and the network setup, this read-ahead mechanism can help improve document browsing speed.

See description of “prefetchPages” for further information.

Parameter: **obfuscateUV**

<PARAM NAME="obfuscateUV" value="false">

`(Version Pro 1.1.52+)`

Note that when using streamer component to deliver documents that are rendered by the Universal Viewing module then you should set this parameter to "false" since the streaming servlet obfuscates all documents that are passed through it.

The Universal Viewing Module cannot deal with the cached file format used when documents are obfuscated - so. to get around this set this parameter to the value of "false".

Please note that setting this parameter to false means that documents displayed using the Universal Viewing Module will be cached using their native format. This could be a problem where access to images is a particularly sensitive issue or users are charged for image viewing/printing.

The default value for this parameter is "true".

Advanced Filename configuration

Note that prior to version 1.1.184 of the viewer, it was necessary to construct the full URL (including streamer URL) from within the viewer itself and to also ensure that any authentication cookies were forcibly passed through to the image repository from the client browser and streamer.

Note also that if you wish to stream MS Office documents, you cannot use this approach so if this functionality is required you wil need to revert to use of the simpler StreamrURL approach described in the previous section

Parameter:

filename

Locate the filename parameter in your HTML which may look something like the following (specifying an absolute URL):

```
<param name="filename" value="http://host/fileserver?fileid=29">
```

Please note that if the path is specified as a relative URL, eg.

```
<param name="filename" value="../docs/fileserver?fileid=29">
```

Then you should first convert it to an absolute URL.

Simple filename configuration

The simple configuration is intended for use with the special `docURLProtoype` configuration parameter in the `streamer.properties` file. The value of `docURLPrototype` should be `${originalDocURL}` in this case. You should note that if this property is not specified in the `streamer.properties` file then the following is the default value:

```
docURLPrototype=${originalDocURL}
```

The configuration change required is to point the client at the Streamer. In order to do this, some things are required: 1. the location of the Streamer, 2. to tell the Streamer where the original document is stored, 3. to provide runtime configuration parameters to the Streamer.

1. The location of the Streamer will be the host and path on which the Streamer is running eg.

   ```
   http://host/streamer
   ```

   You can determine if there is a Streamer running on the specified host and path by entering it into your web browser's address bar. If the Streamer is running, you will see the words "**Streamer servlet is running**" in your web browser. If you do not get this response, you should first correct the problem that is preventing the Streamer from running before continuing.

2. In order to tell the Streamer where the original document is stored, the simplest method is to use the special URL parameter `originalDocURL`. The value of this parameter should be an URI escaped value, eg.

   ```
   http://host/streamer?originalDocURL=http%3A%2F%2Fhost%2Fmyfile.jpg
   ```

   If the filename parameter is generated in JavaScript, this can be done using the `encodeURI(value)` JavaScript method, otherwise it should be done on the server before the HTML page is sent to the browser. Some of the more common escape values are:

Character	**URL encoding**
Plus (“+”)	%2B
Comma (“,”)	%2C
Forward slash (“/”)	%2F
Colon (“:”)	%3A
Semi-colon (“;”)	%3B
Equals (“=”)	%3D
Question mark (“?”)	%3F
Space (“ “)	%20
Quotation marks (“)	%22
Less than (“<”)	%3C
Greater than (“>”)	%3E
Percent (“%”)	%25
Ampersand ("&")	%26

   For example, for the following URL:

   ```
   http://host/fileserver?fileid=29
   ```

   The result would be:

   ```
   http%3A%2F%2Fhost%2Ffileserver%3Ffileid%3D29
   ```

3. Runtime configuration parameters supplied to the Streamer are done using the page URL parameter and the mode URL parameter.

   - `page`

This parameter is mandatory. If it is not specified then the Streamer will send the entire document because an individual page has not been specified.

The `page` parameter will tell the Streamer which page to send to the client first. For example, to send page one, append `&page=1` to the end of the `filename` parameter URL.

- `mode`

  The `mode` parameter can be used to turn on Streamer features – currently the only feature supported is thumbnail generation. To enable thumbnail generation, specify a value of 4. If you don't wish to use thumbnail generation, set this value to 0, or omit the `mode` URL parameter completely.

Given the steps above, the final URL will look like (any line break in this URL should be ignored):

```
http://host/streamer?originalDocURL=http%3A%2F%2Fhost%2Ffileserver%3Ffileid%3D29&
page=1&mode=4
```

This URL will tell the Streamer located at `http://host/streamer` to load a document from `http://host/fileserver/fileid=29` , it will send page 1 and turn on the thumbnail features for this document.

Advanced filename configuration

The advanced configuration is intended for use when the `docURLProtoype` configuration parameter in the `streamer.properties` file is made up from parts of the original URL. For example, if the value for the filename parameter on the client is set as:

```
<param name="filename"
 value="http://host/streamer?orighost=docserver&method=fileapp&docid=89">
```

And the value of `docURLProtoype` in `streamer.properties` is:

```
docURLPrototype=http://${orighost}/${method}?docId=${docid}
```

The final document repository URL will be:

```
http://docserver/fileapp?docId=89
```

Parameter: **cookieAllowedDomainList**

<PARAM NAME="cookieAllowedDomainList" value="streamerhost">

This parameter is required when the source document repository uses cookies to identify logged in users. The value of this parameter is a comma separated list of hosts that the client ViewONE Pro is allowed to send cookies to from its containing HTML page.

This comma separated list should contain the hostname(s) of the server that is running the Streamer server-side code; this will then allow cookies intended for the document repository to be forwarded correctly and allow authorization for requested documents.

Configuring the ViewONE Pro Streamer servlet

The streamer.properties file

This file contains the configuration parameters for the Streamer servlet. The Streamer servlet needs to be told where this configuration can be found. This is done using an `init-param` called `properties`. The value can either be an absolute path (for example `c:\burn\streamer.properties`) or a path that is relative to the web application's root directory (for example `WEB-INF/streamer.properties`).

At the very least, it is suggested that you specify your own value for the "cacheRoot" parameter. You should also review all the configuration options to ensure that they are optimal for your particular setup.

Our downloadable Streamer demo includes a pre-configured web application example to assist you.

If required, the default location of the `streamer.properties` file can be changed by modifying the `web.xml` file which is also in the `WEB-INF` directory. You will find inside the `web.xml` file an `<init-param>` called `properties`. The value of this initialization parameter can either take the form of an absolute path such as `c:\streamerdemo\streamer.properties` or (as is the default) a path that is relative from the streamer web application root directory such as `WEB-INF/streamer.properties`

What follows is a description of each parameter that can be added into the streamer.properties file and how it affects the Streamer's operation.

Parameter: **logFile**

```
logFile=c:\\streamer\\streamer.log

logFile=logs\\streamer.log

logFile=<stdout>
```

This parameter value can either be a filename or a special identifier `<stdout>`. In the case of a filename, logging information that is output during Streamer operation will be stored in the specified file. If using the special identifier, logging will be directed to standard out which in most cases means it will appear in the web application server's log files.

If left blank, then logging will be disabled. Please note that any backslashes (\) should be entered as a double backslash. It can be in the form of an absolute path or relative. If it is specified as relative, then it will be relative to the root of the web application (the deployed streamer directory).

The default value of this parameter is un-set (so no logging will occur).

Parameter: **logFileSize**

```
logFileSize=30mb
```

Available with version 4.0.32+ of the streamer module, this parameter is used to control how often the Streamer will "roll over" its cache file when it reaches the specified size (in Kb) at midnight each day. So if the streamer log file has reached the specified value it will create a new log file with a number included (starting at "1") that is the maximum set minus 1. See logFileCount for further details of how these files are cycled.

The valid size units for this parameter are b (bytes) kb (kilobytes) mb (megabytes) gb (gigabytes).

The default value is `5120kb`.

The minimum value that can be used is `512kb`.

Parameter: **logFileCount**

```
logFileCount=5
```

Available with version 4.0.32+ of the streamer module, this parameter is used to control how many log files the Streamer will keep when it "rolls over" its cache file based on the specified size set in the logFileSize property at midnight each day. So if the streamer log file has reached the specified value it will create a new log file with a number included (starting at "1") that is the maximum set minus 1.

For example, if you have set logFileCount to 10 and the log file name is "log.txt" then the oldest log file will be named "log_9.txt" and most recent wil be named "log_1.txt".

The default value is `10`.

The minimum value that can be used is `2`.

Parameter: **resourcePath**
(deprecated since 1.1.106)

```
resourcePath=c:\\streamerdemo\\v1files
```

Please note that this parameter is no longer required with versions of the Streamer equal to or greater than 1.1.106. This information is provided for reference only.

This parameter should point to the server-side v1files directory. You should ensure that your server-side license file is placed into this directory. Please note that any backslashes (\) should be entered as a double backslash. It can be in the form of an absolute path or relative. If it is specified as relative, then it will be relative to the root of the web application (the deployed streamer directory).

Parameter: **workingPath**

```
workingPath=c:\\streamerdemo\\temp

workingPath=WEB-INF\\working\\temp
```

Please note that this parameter is not mandatory since version 1.1.106. Versions of the Streamer equal to or greater than 1.1.106 will use the web application's temporary area automatically. This behavior can be overridden with this parameter, but it is not advised unless you are having problems with the default functionality.

This parameter should point to a writable directory in which temporary working files will be placed.

Please note that any backslashes (\) should be entered as a double backslash. It can be in the form of an absolute path or relative. If it is specified as relative, then it will be relative to the root of the web application (the deployed streamer directory).

The default value is a subdirectory of the private temporary directory in the servlet context.

Parameter: **docURLPrototype**

```
docURLPrototype=${originalDocURL}

docURLPrototype=http://host/repository/${file}
```

This is the prototype for the URL that is passed to the HTTP repository store when a user requests a file through the Streamer. Tokens can be specified as part of this prototype using `${` to start the token and `}` to end it. The values for these tokens are taken from the URL that the Streamer was called with.

The simplest method of using this configuration parameter is to use the `originalDocURL` token name. See the section entitled “Simple Configuration” under “Configuring the ViewONE Pro viewer for use with the Streamer”.

Alternatively, a more complex method is available which provides greater control over the URL. You can substitute values from the original request URL eg. if your Streamer URL (which is specified in the filename parameter in the HTML page for the ViewONE Pro client) is:

```
http://localhost:8080/viewone/streamer?path=images&file=test
```

and your `docURLPrototype` is:

```
http://host/fileserver?type=${path}&name=${file}
```

then the resulting request will be to:

```
http://host/fileserver?type=image&name=test
```

The default value is `${originalDocURL}`.

Special tokens:
(since 1.1.172+)

Sometimes it is necessary to re-write or alter a document repository URL because the streamer's view of the network is different to that of the browser. This might be due to a firewall (such as port redirection, IP address remapping) or network configuration (such as DNS resolution differences). For example, to obtain a document from within the browser, a sample URL might read:

```
http://docstore.myhost.com/docserver?id=29
```

The same document request for the streamer in our fictitious intranet might read:

```
http://docstore:8080/docserver?id=29
```

There are a selection of special tokens that can be used to cater for this situation. These tokens allow you to extract discrete parts of the original document repository URL such as the protocol (eg. http/https) or the port (eg. 80, 8080, 443). Listed below is the complete list - each of the examples listed next to the tokens are based on the URL sample above:

`${originalDocURL.protocol}` eg. http

`${originalDocURL.host}` eg. docstore.myhost.com

`${originalDocURL.port}` eg. `80` (Note, if no port is supplied, 80 is the default)

`${originalDocURL.path}` eg. /docserver

`${originalDocURL.query}` eg. id=29

`${originalDocURL.authority}` eg. docstore.myhost.com

For the sample above, we need to remap both the hostname and the port, this would be achieved with the following (there should be no line breaks in the configuration file):

```
docURLPrototype=${originalDocURL.protocol}://docstore:8080${originalDocURL.path}?
${originalDocURL.query}
```

This value for `docURLPrototype` will take any hostname from the incoming document repository URL and replace it with "docstore:8080".

Parameter: **docIdPrototype**

`docIdPrototype=${originalDocURL}`

In order for the Streamer to work, it needs to be able to identify every document uniquely internally. If you are using a value of `${originalDocURL}` for `docURLPrototype`, then by its nature it represents a unique identifier for the document and as a result the `docIdPrototype` can be set to the same value. In most cases, this is an adequate setting, but if you access documents via a proxy which may result in differing URLs for the same document, then you should consider the more advanced usage of `docIdPrototype` below.

The unique identifier can alternatively be built up directly from your original Streamer URL in a similar fashion to the `docURLPrototype`; tokens can be specified as part of this prototype using `${` to start the token and `}` to end it. The values for these tokens are taken from the URL that the Streamer was called with. For example, if your Streamer URL (which is specified in the filename parameter in the HTML page for the ViewONE Pro client) is:

`http://hostname/streamer/streamer?docId=89&version=20`

and your `docIdPrototype` is:

`docIdPrototype=${docId}-${version}`

then the resulting unique id will be:

`89-20`

The default value is `${originalDocURL}`.

Parameter: **sessionCookieList**

```
sessionCookieList=JSESSIONID
```

A comma separated list of cookies that must be present on the request from the client if they have been specified with this configuration parameter.

An empty value indicates to the Streamer that the document repository does not use authentication and all authentication will be disabled.

To set up cookie forwarding within ViewONE, use the HTML parameter `cookieAllowedDomainList` (see page 13).

If your document repository requires a user to be logged in before they can retrieve documents then the value of this parameter is important. It indicates to the Streamer which cookies should be forwarded to the document repository for authentication purposes. It should consist of a comma separated list of cookie names that are used by the document repository to identify individual logged in users. An asterisk (*) wildcard is acceptable, so if this parameter was specified as:

```
sessionCookieList=ASPSESSION*
```

Then the following named cookies (for example) would be forwarded to the repository:

```
ASPSESSION1293
```

```
ASPSESSIONID
```

If your document repository does not use authentication, leave the value of this parameter empty and authentication will be disabled.

If this value is not specified in the properties file it defaults to `JSESSIONID`.

Note: If this parameter is specified and the client does not send all of the cookies, then the Streamer will consider the user to be unauthenticated and will not stream the document to the viewer.

Parameter: **sessionAuthenticationTimeout**

```
sessionAuthenticationTimeout=60s
```

After the time specified by this parameter, the user will be re-authenticated with the document repository. This is to cater for cases where a user is already viewing a document but access to it is revoked in the middle of viewing. The user will not be able to view any subsequent parts of the document after they have been revoked access.

The valid time units for this parameter are s (seconds) m (minutes) h (hours) d (days) w (weeks).

If this parameter is not specified then no session authentication caching will be performed.

Parameter: **signInResource**

```
signInResource=SignIn.jsp
```

If a user makes a request for a document and the session has expired, then the web server will usually return a redirect status (302 - moved temporarily). In this case, we will use extra logic to check if the user has been logged out by looking at the Location: header field for the string specified by `signInResource`. If this string is found, then the Streamer knows that user has been logged out.

If this parameter is unspecified then no attempt will be made to see if a user has been logged out by a document repository. If the document repository doesn't support sessions or logins then this value can be left unspecified.

The default value is unspecified.

Parameter: **serverSupportsHeadRequestMethod**

```
serverSupportsHeadRequestMethod=true
```

When set to true (the default), this parameter will this use the HTTP HEAD method for collecting information about a document without retrieving the entire document over the network. Support of this feature allows for more efficient protocol dialog. If the document repository server does not support the HEAD method or it does not return meaningful values for HTTP header fields such as Content-Length, Content-Type, or Last-Modified, then this should be set to false.

Valid values are: true or false. The default value is true.

Parameter: **modifiedDocumentCheckMethod**

```
modifiedDocumentCheckMethod=lastmodified

modifiedDocumentCheckMethod=uniqueid
```

For optimum Streamer performance, documents are cached by the Streamer. It is therefore important that the Streamer does not send out of date documents to users if the document has changed since the Streamer cached the document.

lastmodified

Most document repositories will provide an HTTP header called “Last-Modified” to tell the client an idea of when a document was last change. The Streamer can use this information when the value of this parameter is set to `lastmodified`. If you are serving up files from an HTTP server, this will probably be the correct option.

uniqueid

An alternative to using modified dates, some document repository systems rely on the fact that an updated document has a different document identifier from the original document. In these situations, the modified date becomes irrelevant and you should set the value of the parameter to `uniqueid` so that the Streamer knows to take this into account. It is important when using this mode that the URL for one document version is not the same as the URL for a different version of the same document.

If a value for this property is not specified in the properties file, the default value is `lastmodified`.

Parameter: **modifiedDocumentMinimumCheckTime**

```
modifiedDocumentMinimumCheckTime=5m

modifiedDocumentMinimumCheckTime=1d
```

This parameter is used to control how often the Streamer will check back with the document repository to see if a particular document has been modified since it was last accessed. As the name implies, the Streamer will check no more often than the time specified in this parameter. The reason for this parameter is that it is more efficient at a network level to avoid re-querying the document repository every time a document is accessed. If you really want the repository to be queried with every document access, a value of 0 should be supplied, however this is not recommended.

The valid time units for this parameter are s (seconds) m (minutes) h (hours) d (days) w (weeks).

If a value for this property is not specified in the properties file, the default value is `120s`.

Parameter: **documentExpiryTime**

```
documentExpiryTime=120s
```

This parameter controls for how long that a document is kept in memory after it has been accessed. The internal counter is reset for every access and after no more accesses have occurred after the time specified in this parameter, the document is closed. The ideal value for this parameter will be the length of time that the average user browses pages in a single document. Setting this parameter to a high value is counter-intuitive as it will keep resources tied up on the server.

The valid time units for this parameter are s (seconds) m (minutes) h (hours).

If a value for this property is not specified in the properties file, the default value is `120s`.

Parameter: **poolInitialEntryCount**

```
poolInitialEntryCount=1
```

This parameter controls the number of concurrent document "holders" that will be initially created when the Streamer servlet is initialized. This value will grow automatically up to the limit of `poolMaxEntryCount`. It is provided as a fine-tuning option and can be safely left at its default value of 1.

If a value for this property is not specified in the properties file, the default value is `1`.

Parameter: **poolMaxEntryCount**

```
poolMaxEntryCount=10
```

This parameter controls the maximum number of concurrent documents that will be held open in memory. A low value (for example 1) will result in low server memory usage but will cause bottlenecks if you have more than one user viewing separate documents that have not been processed by the Streamer. A large number may be limited by the amount of memory your web application server has available but it may improve performance for the situation when many users are viewing un-cached documents.

If a value for this property is not specified in the properties file, the default value is `1000000`.

Parameter: **cacheRoot**

```
cacheRoot=c:\\streamerdemo\\cache
or
cacheRoot=cache (relative path to web application root directory)
```

It should be noted that in a production environment, this property value should be specified.

This parameter controls where the Streamer will keep its cached document store. There are size controls for this store (see configuration below).

Please note that any backslashes (\) should be entered as a double backslash. It can be in the form of an absolute path or relative. If it is specified as relative, then it will be relative to the root of the web application (the deployed streamer directory).

If a value for this property is not specified in the properties file, the default value is WEB-INF/data/<*servletname*>. Where <*servletname*> is the name of the servlet as defined in the web.xml.

Parameter: **cacheEmptyOnRestart**

```
cacheEmptyOnRestart=false
```

This parameter controls whether all the cached documents in the Streamer's document store are expired when the Streamer servlet has been restarted. It is advised to set this parameter to false as one of the advantages of the Streamer is that it will cache streamed documents ready for fast retrieval next time they are accessed.

Valid values for this parameter are true or false. The default value is `false`.

Parameter: **cacheSizeLimit**

```
cacheSizeLimit=100mb
```

This value is the size that the cache should be kept to. It is a soft limit meaning that the cache could exceed this value between the last time a cache check was performed and the time left before the next check is done (specified by `cacheCheckTime`). If the cache is found to exceed this size limit, cached document data will be deleted from it, oldest first, until the size is back within the limit.

The valid size units for this parameter are b (bytes) kb (kilobytes) mb (megabytes) gb (gigabytes).

The default value is `100mb`.

Parameter: **cacheAgeLimit**

```
cacheAgeLimit=1d
```

This parameter determines the maximum age of a document in the cache. It is a soft limit meaning that documents in the cache could exceed this value between the last time a cache check was performed and the time left before the next check is done (specified by `cacheCheckTime`). If data for a document inside the cache is found to exceed this age limit, the cached document data will be deleted from it.

The valid time units for this parameter are s (seconds) m (minutes) h (hours) d (days) w (weeks).

The default value is `1d`.

Parameter: **openDocumentWaitTime**

```
openDocumentWaitTime=2s
```

This parameter determines the amount of time to wait prior to opening a second instance of a document that is already in the streamer cache. This should only be used where it is likely that a large document that is being pre-cached will be viewed at the same time and the page to be viewed has not yet been generated.

The default value is `0s` which will block the viewing of a page yet to be generated until the pre-cache process has completed.

Using the ViewONE Pro Streamer servlet for Pre-caching.

Documents may be pre-cached ready for streaming, thus reducing wait time of users accessing large documents for the first time. Pre-caching is desirable for large documents where the first user request may incur a considerable delay while the document is fetched from the repository and the page(s) requested are separated and stored prior to being sent to the user.

Pre-caching is initiated by issuing an HTTP request to the streamer which then examines the request and sends a response to the caller. The response will indicate either that the request has been added to the queue or provide feedback on the reason for not adding the request to the queue. The document splitting is then in a separate process on the server independently of the request.

Pre-cache Server Configuration

Pre-caching is a function of the Streamer Servlet and requires a valid configuration as per the previous section 'Configuring the ViewONE Pro Streamer servlet'.

Pre-caching uses a set of worker objects that access a queue of pre-cache requests. The number of these objects and how long they remain active when there are no requests may be specified.

Parameter: **maxPreCacheWorkers**
(Pro 1.1.160+)

```
maxPreCacheWorkers=5
```

This parameter determines the maximum number of threads that will be created to manage the handling of document pre-caching requests. This limit will not be exceeded and where there are not enough workers available for the number of requests made the requests are queued. This queue is not persisted and will be lost if the Streamer servlet is shutdown.

The value set should be less than the `poolMaxEntryCount` as the process uses the same document pool as online users.

The default value is `5`.

Parameter: **maxPreCacheWorkerAge**
(Pro 1.1.160+)

```
maxPreCacheWorkerAge=60s
```

This parameter determines the maximum unused time of a pre-cache worker thread. When a worker thread has been unused for more than this time it will be destroyed to minimize server resource usage.

The default value is `60s`.

Pre-cache Authentication and Authorization

Pre-caching has the same authentication and authorization criteria applied to it as the standard streamer. A user must be logged into the document repository and have access rights to read the document. With pre-caching, however, it may be desirable to automate the pre-caching process and server-side authentication parameters can be configured to facilitate this.

When using server side authentication the document access authorization applied by the repository is exactly the same as for normal streamer functionality.

Any server side login parameters used by the streamer are only applied where pre-caching of documents and thumbnails is requested and are not permitted for any other streamer functions.

Pre-caching using server side authentication may also be restricted to a limited set of computers via specifying permitted IP addresses. Ordinarily this should be restricted to within the local network, but may be setup to allow any address.

Parameter:

authentication

(Pro 1.1.160+)

```
authentication=system
```

This parameter is used to specify the whether user initiated or server side login is used to create the valid HTTP session required to access the document repository. Only HTTP login is supported.

Valid values are `default`, `system` and `parameter`.

- `default` will use the current user session. In this mode pre-caching may only be made by a user already logged into the document repository. This requires that the Streamer servlet is in the same web-application as the repository.
- `system` will use the parameters configured in the streamer.properties file (see below).
- `parameter` indicates that authentication values may be passed in on the request and where they are not present it will use the streamer.properties entries.

The default value is `default`.

Server-side authentication is only available where `system` or `parameter` are set as the authentication value. Setting `parameter` and not passing the authentication criteria on the request is the same as setting system. Use `parameter` where you need different automated systems to access the same repository with different authentication criteria. The repository information may then be set in the properties file.

Note: where the default mechanism is used the user session on the repository could expire prior to the streamer request on the repository being made – this would cause the pre-cache to fail. This could happen where there is a long list of documents already queued for pre-caching, the user explicitly logs out of their repository session, the session expiry is very short or the user session has been dormant on the repository prior to the pre-cache request.

Parameter: **allowedIPAddresses**
(Pro 1.1.160+)

```
allowedIPAddresses=192.168.0.*
```

This parameter indicates where requests for server based authentication to the repository are permitted from. Valid values may include * to indicate any value.

This should be setup to restrict requests as otherwise it would be possible for anyone, or any process, to make requests that could impact the document repository using the authentication parameters set on the server. Although the impact would be restricted by the number or workers configured and the speed of the server that the pre-caching is performed on, not restricting access could enable an unauthorized process to put unnecessary load on the server and repository.

If unauthorized access restriction by IP address is not practical or desirable, then a Streamer servlet should be setup separate from that used by the ViewOne client application and additional security added.

This parameter is mandatory when `authentication` parameter is set to `system` or `parameter` and failure to provide a value will cause the Streamer startup to fail.

The default value is unspecified.

Parameter: **authentication.handler**
(Pro 1.1.160+)

```
authentication.handler=default
```

This parameter indicates which login mechanism to use for server authentication on the repository. Valid values are `default, Workplace`, or any fully qualified Java classname for a subclass of `ji.streamer.servlet.jiServerLoginHandler`.

This parameter is mandatory when `authentication` parameter is set to `system` or `parameter` and failure to provide a value will cause the Streamer startup to fail.

The default value is unspecified.

Pre-cache : 'default' Authentication Handler

The `default` setting provides an HTTP login to a document repository using parameters specified in the streamer configuration file.

This configuration may be used for any repository where the user login requires the posting of their credentials.

The parameters required for login will be either taken from the request or the streamer configuration as per the value specified in the `authentication` parameter (above).

Note: where a user is already logged in their authentication criteria will be ignored and the parameters posted and/or server-side properties configured for authentication will be used.

Parameter: **authentication.postParameters**
(Pro 1.1.160+)

```
authentication.postParameters=userId,password,[ param1,param2,…]
```

This parameter is used to specify the names of the parameters that will be posted to the login URL specified by the parameter `authentication.loginURL.`

This parameter does not have any default values.

The `userId` and `password` parameter should normally be specified in this list (actual parameter names will vary) and then added to the properties with the prefix `authentication.`. Parameter names must be composed ASCII characters and may not include `=`, `:`, or whitespace (For more detail please refer to Java documentation for java.util.Properties class).

For example, where a username and password are required the configuration file would contain the following:

```
authentication.postParameters=userId,password
authentication.userId=joe blogs
authentication.password=secret
```

Parameter: **authentication.signInResource**
(Pro 1.1.160+)

```
authentication.signInResource=WcmSignIn.jsp
```

This is used to build the login address.

This parameter is mandatory when `authentication.handler` is set to `default` and failure to provide one will cause the Streamer startup to fail.

The default value is unspecified.

Parameter: **authentication.loginURL**
(Pro 1.1.160+)

```
authentication.loginURL=http://yoursite.com/repository/login.jsp
```

This parameter specifies the URL where the repository login credentials are posted.

This parameter is mandatory when `authentication.handler` is set to `default` and failure to provide one will cause the Streamer startup to fail.

The default value is unspecified.

Parameter: **authentication.loggedInURL**
(Pro 1.1.160+)

```
authentication.loggedInURL= http://yoursite.com/repository/browse.jsp
```

This parameter is used to verify an existing session

This parameter is mandatory when `authentication.handler` is set to `default` and failure to provide one will cause the Streamer startup to fail.

The default value is unspecified.

Pre-cache : 'Workplace' Authentication Handler

The `Workplace` implementation may be used to connect to an IBM FileNet Workplace repository. It uses HTTP login in the same way as the default handler, but constructs a number of the parameters from the properties values.

The parameters required for login will be either taken from the request or the streamer configuration as per the value specified in the `authentication` parameter (above).

Note: where a user is already logged in their authentication criteria will be ignored and the parameters posted and/or server-side properties configured for authentication will be used.

Parameter: **authentication.userId**
(Pro 1.1.160+)

```
authentication.userId=username
```

This parameter specifies the user identity to login to Workplace with.

This parameter is mandatory when `authentication.handler` is set to `Workplace` and failure to provide one will cause the Streamer startup to fail.

The default value is unspecified.

Parameter: **authentication.password**
(Pro 1.1.160+)

```
authentication.password=userpassword
```

This parameter specifies the password to login to Workplace with.

This parameter is mandatory when `authentication.handler` is set to `Workplace` and failure to provide one will cause the Streamer startup to fail.

The default value is unspecified.

Parameter: **authentication.base**
(Pro 1.1.160+)

```
authentication.base=http://192.168.255.21
```

This parameter is used to build the login address and the parameters required by Workplace to login.

This parameter is mandatory when `authentication.handler` is set to `Workplace` and failure to provide one will cause the Streamer startup to fail.

The default value is unspecified.

Parameter: **authentication.originPort**
(Pro 1.1.160+)

```
authentication.originPort=7001
```

This parameter is used to build the login address and the parameters required by Workplace to login.

This parameter is mandatory when `authentication.handler` is set to `Workplace` and failure to provide one will cause the Streamer startup to fail.

The default value is unspecified.

Parameter: **authentication.appId**
(Pro 1.1.160+)

```
authentication.appId=Workplace
```

This parameter is used to build the login address and the parameters required by Workplace to login.

This parameter is mandatory when `authentication.handler` is set to `Workplace` and failure to provide one will cause the Streamer startup to fail.

The default value is unspecified.

Parameter: **authentication.signInResource**
(Pro 1.1.160+)

```
authentication.signInResource=WcmSignIn.jsp
```

This is used to build the login address.

This parameter is mandatory when `authentication.handler` is set to `Workplace` and failure to provide one will cause the Streamer startup to fail.

The default value is unspecified.

Pre-cache Client Parameters

The process of initializing a pre-cache request is to post an HTTP request to the streamer. Any program capable of submitting an HTTP request may be used to achieve this either manually or automatically.

The pre-cache request is very quick as the job is queued and handled by a configured number of processes, but the caching itself may take a few minutes for very large documents.

While a document is being pre-cached it may not be viewed by another client unless the `openDocumentWaitTime` parameter has been set. This should be set where it is likely that a large document may be viewed whilst pre-caching is still in progress.

When pre-caching is requested a simple text response is returned indicating a success or failure. The success message will consist of: `REQUEST_QUEUED : <document id>`. The request will not fail if an invalid document id is requested as this is not validated at request time. A failure will indicate that the parameters were invalid. The failure message will consist of: `Error: <error message>`.

Pre-cache requests may be made for the main pages or for the thumbnail images.

Where the server has been setup with `authentication=parameter` the login credentials may also be added to the request parameters.

Parameter: **precache**

```
precache=true
```

This parameter signals to the streamer that the caching the document should be done without requiring the client to wait for a response. The value of the parameter is not used, but should be set to `true` for readability.

Parameter: **pages**

```
pages=1,7-10
```

This parameter allows you to select a single page or multiple pages to be cached. The single numbers may be entered in any order, but the ranges must specify the lower page number first. Page numbers that are outside the range in the document are ignored. The

parameter may only contain integers with commas to separate the page numbers and ranges and a single hyphen to denote a range of pages.

The default value is unspecified. An invalid or missing value will cause an error to be reported.

Where this parameter is not present the whole document is pre-cached.

Parameter: **thumb**

```
thumb=true
```

This parameter is used to get the thumbnail images pre-cached.

The default value is false.

Parameter: **thumbsize**

```
thumbsize=100
```

This parameter is used to specify the pixel width of the thumbnails generated. This should be set to the same value used by the viewer. If this parameter is not specified the value, if present, in streamer.properties is used.

The default value is 120.

Pre-cache: Example Request

If your filename parameter looks like this:

```
<param name="filename"
 value="http://host/streamer?orighost=docserver&method=fileapp&docid=89">
```

then a URL of:

```
http://host/streamer?orighost=docserver&method=fileapp&docid=89&precache=true
```

will pre-cache the main view for the whole document.

```
http://host/streamer?orighost=docserver&method=fileapp&docid=89&precache=true&thu
mb=true&thumbsize=100
```

will pre-cache the thumbnails for the whole document.

Troubleshooting

Once installed, the Streamer can be called with no URL parameters to ensure that it is running. For example, if your web application server is accessed on port 8080 and the Streamer has been installed under the default web application name on a host called `cohost`, then the following URL entered into a web browser should display "**Streamer servlet is running**":

```
http://cohost:8080/streamer/streamer
```

If you do not get this response then there could be a problem with the deployment of the streamer web application or there is a configuration problem.

Clearing temporary working files

Unless the behavior has been overridden with the "workingPath" parameter of the streamer.properties file, versions of the Streamer equal to or greater than 1.1.106 use the web application provided temporary area for servlets which should be cleared down automatically by the web application server. This is normally done when re-starting the server or re-loading servlets but you should refer to your web application server's documentation for exact details.

If a "workingPath" is in use, then the folder that this parameter points to should be emptied.

Clearing the Streamer's document cache

The "cacheRoot" parameter controls where the Streamer will keep its cached document store. If you wish to clear all the documents in the Streamer's cache then the folder that this parameter points to should be emptied. **Note:** emptying this folder will require regeneration of streamed documents.

Logging

The first step to diagnosing issues is to ensure that the `logFile` parameter in `streamer.properties` has been set up correctly. This will output information about the Streamer as it processes document requests.

If you find that after enabling logging that no log file is produced then you may need to check the logs of your web application server to see if the servlet has started at all or if it is unable to write to the log file location specified.

Access denied error

This error indicates that a user has tried to access a document that they do not have access to. It occurs when the Streamer tries to validate a logged in user with their cookies against the source document repository.

User logged out error

This error indicates that a user has tried to access a document but the source document repository has indicated to the Streamer that the user has been logged out. This condition is detected using the criteria under the `signInResource` configuration in `streamer.properties`.

Internal server error

Internal server errors (code 500) indicate that there has been a problem on the server-side and will require investigation of firstly the Streamer's log file and secondly your web application server's log file in order to investigate why a problem has occurred.

No cookies supplied on request

When accessing document repositories that require authentication, it is important that any session cookies which instruct the repository that the current user is logged on are passed from the client to the Streamer so that the Streamer can forward them on to the document repository. If you are having problems accessing documents and you see "No cookies supplied on request" in the error message, it may be due to the ViewONE client not sending cookies to the Streamer correctly. The ViewONE client may output information about any cookie errors in the Java console.

Note: When using the Firefox browser as a client, it is important that the MAYSCRIPT applet element is present in the <applet> definition so that the client can correctly obtain the browser's cookies. If it is not, you will see errors in the client's Java console. The correct way to specify the mayscript parameter is as follows:

```
<applet mayscript="true" name="ViewONE" ...
```

It is not sufficient to use the `<param name="mayscript" value="true">` method.

The most common cause of these not working is where 'HttpOnly' cookies are being used. These are special cookies that only the browser can see and as a result they are not passed on to any plugins inside the browser's HTML page (such as ViewONE).

Another approach to forcing cookies through to the is to pull the authentication cookie from the HTTP request and sets it on the viewer using the "documentCookies" HTML parameter. This is a comma separated list of cookies to set on any viewer HTTP requests. A limitation here is that the page holding the viewer needs to be dynamically generated so that the cookie can be retrieved. For security reasons, these cookies will only be sent to the hostnames as specified in the "cookieAllowedDomainList" HTML parameter. This needs to be set to the hostname or IP address of the server that the viewer to should send the cookies to when it makes an HTTP request. There is a further HTML parameter called "sendCookiesWithCookieHeader", this tells the viewer that the cookies set with the "documentCookies" HTML parameter should be sent to the server with the "Cookie:" HTTP header. If not, they are sent with a special HTTP header "CookieClone" for server side processes that might want to retrieve the viewer accessible cookies separately from the ones forcibly set.

java.lang.InternalError

If you see this error "java.lang.InternalError: Can't connect to X11 window server using ':0.0' as the value of the DISPLAY variable error" appearing in the log files then you need to ensure that you ***either*** point to an X11 server ***or*** make use of the -Djava.awt.headless=true Java runtime parameter

Server Response Code 401

If you see this error then the problem is that the streamer is seen as "unauthorized" to access the image repository. This can be caused by NTLM authentication. A solution is to disable NTLM authentication on requests from the streamer to the image repository.

IO error: Connection timed out

This error message may occur because the streamer is having difficulty connecting to the document repository. This could be related to a firewall or network configuration issue. Normally the document repository receives requests directly from the client browser but when the streamer is being used, it sits between the client and the server (usually behind the same firewall as the document repository). Depending on the configuration of the firewall, it may not be letting requests through from the streamer because they are being routed in a different way to client requests originating directly from the browser.

Resolving this issue can be done either by altering your firewall/DNS configuration or re-writing the URL for the requested document so that the streamer can contact the document store successfully. For more information on how to re-write the URL, please see the documentation for docURLPrototype

Platform compatibility notes for server side code

Streaming of TIFF and PDF files is platform independent, relying only on a Java 1.4+ based web application server (such as Tomcat, Weblogic or JBoss).

Generation of thumbnails is dependent upon the source file. TIFF thumbnail generation is platform independent and PDF thumbnail generation is currently limited to Windows 32bit servers.

Appendices

Installing the ViewONE Pro Streamer trial

The ViewONE Pro Streamer trial is supplied as a standard web application directory that can be deployed into the web application server of your choice. Once installed, it exposes a servlet called `ji.streamer.servlet.jiDefaultStreamerServlet` The web-app directory comes supplied with a default `web.xml` which is set up to call this servlet under a path called `/streamer`, it also specifies the location of the Streamer's configuration file (`streamer.properties`).

Apache Tomcat installation

1. Install Apache Tomcat 5.5 (http://tomcat.apache.org/ - if you are using a Windows OS, the EXE download is recommended)

2. Navigate to the Tomcat webapps directory (eg. C:\Program Files\Apache Software Foundation\Tomcat 5.5\webapps)

3. Copy the "viewone" folder supplied as part of the Streamer distribution to the webapps directory described in the previous step.

4. Restart Tomcat (use Start->Programs->Apache Tomcat 5.5->Configure Tomcat to stop and re-start the Tomcat service)

5. Launch your web browser

6. Navigate to http://localhost:8080/viewone/index.htm

7. Navigate to ViewONE Pro – Documetn Streaming Server Module Demo

You will be presented with a page that provides an introduction to the ViewONE Pro Streamer and some links that will demonstrate the Streamer running from your local machine.

Installation on other Web Application servers

There are no specific instructions for installing the Streamer servlet on other web application servers such as WebLogic, JBoss, WebSphere, etc. The instructions provided above are simply a sample installation using the free Apache Tomcat web application server to get you up and running quickly with the free trial - they are not a definitive guide to installing the streamer across all the different web application servers that are available.

The reason for this is that the streamer module is a servlet that needs to be installed on a web application server in order for it to work - but this can be ANY web application server of your choosing - and most customers will already have a preferred web app server they use and know about. Daeja are not (and cannot be) experts in every single web application server out there and so cannot provide specific instructions on how to deploy a servlet onto every single known web app server either.

So the minimum requirement for setting up the streamer server side component is that you have a web application server in place that you are familiar with and can deploy servlets on, gather logs files, etc so that you should be able to deploy the servlet and get to the point where you can enter the URL to the streamer and see that message:

“Streamer Servlet is running"

web.xml file

Inside your web application, locate the `WEB-INF/web.xml` configuration file which may need creating (you can copy and paste the sample fragment below should you require it). You will need to modify this configuration file to:

- include a servlet definition for the main Streamer class (`ji.streamer.servlet.jiDefaultStreamerServlet`)
- provide the Streamer servlet with an `init-param` called `properties` which defines the location of a properties file to control the Streamer's configuration (see below for further explanation)

In the following sample fragment, the name "streamer" has been chosen for this servlet and it will be called when a request is made to `/streamer` (as defined by the `servlet-mapping` tag):

Sample web.xml file

```
<?xml version="1.0" encoding="ISO-8859-1"?>

  <!DOCTYPE web-app PUBLIC
       "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
       "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>

  <servlet>
   <servlet-name>streamer</servlet-name>
   <servlet-class>ji.streamer.servlet.jiDefaultStreamerServlet</servlet-class>
     <init-param>
        <param-name>
           properties
        </param-name>
        <param-value>
           WEB-INF/streamer.properties
        </param-value>
     </init-param>
  </servlet>

  <servlet-mapping>
    <servlet-name>streamer</servlet-name>
    <url-pattern>/streamer</url-pattern>
  </servlet-mapping>

</web-app>
```

streamer.properties file

This file contains the configuration for the Streamer servlet. The Streamer servlet needs to be told where this configuration can be found. This is done using an `init-param` called `properties`. The value can either be an absolute path (for example `c:\burn\streamer.properties`) or a path that is relative to the web application's root directory (for example `WEB-INF/streamer.properties`).

At the very least, it is suggested that you specify your own value for the "cacheRoot" parameter. You should also review all the configuration options to ensure that they are optimal for your particular setup.

For a full list of all the configuration options, please see the section in this manual called "Configuring the ViewONE Pro Streamer servlet".

Our downloadable Streamer demo includes a pre-configured web application example to assist you.