IBM Power Systems with IBM i performance and tuning tips for Oracle’s JD Edwards EnterpriseOne WebSphere-based HTML servers

John Brock
IBM i ERP Development Support
and
Clark Scholten
IBM Advanced Technical Skills – Oracle Applications

July 2013
# Table of contents

**Change history** ............................................................................................................................ 2
**Executive summary** ...................................................................................................................... 3
**Oracle support for WebSphere Application Server** ................................................................. 3
**IBM HTTP server performance** .................................................................................................. 4
  Measure and Analyze .......................................................................................................................... 5
  Adjust ................................................................................................................................................... 6
    1) Minimize the logging in the production environment ............................................................... 7
    2) Allow unlimited requests per connection .............................................................................. 9
    3) Set the number of threads for each job .................................................................................. 10
    4) Enable compression .................................................................................................................. 11
    5) Setup Fast Response Cache Accelerator (FRCA) ................................................................ 13
    6) Adjust HTTP Server subsystem settings in IBM i .................................................................. 19
**WebSphere Application Server tuning** ....................................................................................... 20
  Measure and analyze .......................................................................................................................... 20
  Adjust .................................................................................................................................................. 20
    1) Adjust the heap size for the JVM .......................................................................................... 21
    2) Adjust the number of threads running in the JVM .............................................................. 22
    3) Enable localhost name caching ......................................................................................... 23
    4) Adjust HTTP transport channel settings ........................................................................... 24
    5) Adjust load balancing settings for WAS ND ....................................................................... 25
    6) Adjust WebSphere subsystem settings in IBM i .................................................................. 27
**Known issues and optimizations** ............................................................................................... 29
  Out of memory condition when viewing large pdf’s ........................................................................ 29
  Contention on shareclasses cache ............................................................................................... 30
  Plugin scaling issue ....................................................................................................................... 32
  Additional garbage collection optimization with gencon ............................................................... 34
  Tuning garbage collection threads ............................................................................................... 35
  Changing to the Toolbox JDBC Driver ......................................................................................... 36
**TCP/IP settings** ........................................................................................................................... 37
**Appendix A – additional information** .......................................................................................... 38
  Contact information ....................................................................................................................... 38
  References .......................................................................................................................................... 38
    IBM i ............................................................................................................................................... 38
    WebSphere and Java ..................................................................................................................... 38
    Additional Resources .................................................................................................................. 38
**Trademarks and special notices** .................................................................................................. 39
# Table of figures

- Figure 1 - Starting IBM HTTP server administration for IBM i from IBM System i Navigator ........4
- Figure 2 - Real Time Server Statistics show current activity on the HTTP server instance ..........5
- Figure 3 - Select the HTTP server instance ..............................................................................6
- Figure 4 - Change the error log settings in HTTP server instance .............................................7
- Figure 5 - Eliminate access logging for the HTTP server instance ...........................................8
- Figure 6 - Allow unlimited requests per connection for the HTTP server instance .................10
- Figure 7 - Set the number of threads to handle requests in the HTTP server instance ..........11
- Figure 8 - Enable compression for the HTTP server instance ..................................................12
- Figure 9 - Set the output filter for text in the JAS directories ....................................................13
- Figure 10 - Enable FRCA ...........................................................................................................14
- Figure 11 - Enable FRCA File Cache .........................................................................................15
- Figure 12 - Display Configuration File .......................................................................................16
- Figure 13 - Configure FRCA file paths ......................................................................................16
- Figure 14 - FRCA File Cache Setup ...........................................................................................17
- Figure 15 - FRCA Logging .........................................................................................................18
- Figure 16 – QHTTPSVR routing entry. .......................................................................................19
- Figure 17 - Set the initial and maximum heap size for the JVM in WebSphere .......................22
- Figure 18 - Set the number of threads for the JVM in WebSphere 7.0 and 8.5 .......................23
- Figure 19 - Enable localhost name caching ...............................................................................24
- Figure 20 - Adjust maximum persistent requests for HTTP Transport Channel .....................25
- Figure 21 - Change the load balancing option to “Random” .......................................................26
- Figure 22 - Change the session affinity settings .........................................................................26
- Figure 23 - Generate and Propogate the Plug-in ......................................................................27
- Figure 24 - Garbage collection information from the WRKJVM command .............................29
- Figure 25 - Custom property for channelwritetype ..................................................................30
- Figure 26 - Java Virtual Machine configuration ........................................................................31
- Figure 27 - Generic JVM arguments for separate shareclasses caches ....................................32
- Figure 28 - Finding the plugin location from the HTTP configuration file ...............................33
- Figure 29 - Java Virtual Machine configuration .......................................................................34
- Figure 30 - Generic JVM arguments for gencon garbage collection .........................................35
- Figure 31 - Generic JVM arguments for garbage collection threads .........................................36
- Figure 32 - Change to the Toolbox JDBC driver ......................................................................37
**Change history**

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Editor</th>
<th>Editing description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>September 30, 2010</td>
<td>Jim Denton</td>
<td>Original version</td>
</tr>
<tr>
<td>2.0</td>
<td>September 30, 2011</td>
<td>Diane Webster</td>
<td>Add FRCA, remove WebSphere 6.0</td>
</tr>
<tr>
<td>3.0</td>
<td>August 10, 2012</td>
<td>Diane Webster</td>
<td>Update WebSphere tuning guidelines</td>
</tr>
<tr>
<td>4.0</td>
<td>May 10, 2013</td>
<td>Clark Scholten</td>
<td>Update WebSphere tuning guidelines, remove WebSphere 6.1, add WebSphere 8.5</td>
</tr>
<tr>
<td>5.0</td>
<td>July 2, 2013</td>
<td>Clark Scholten</td>
<td>Update WebSphere &amp; JDBC tuning guidelines</td>
</tr>
</tbody>
</table>
Executive summary

This white paper provides tuning techniques to optimize the performance of JD Edwards EnterpriseOne HTML servers implemented on IBM i using WebSphere® 7.0 and 8.5. JD Edwards EnterpriseOne is a comprehensive software package that provides Enterprise Resource Planning (ERP), Supply Chain Management (SCM), and Customer Relationship Management (CRM) capabilities using HTML-based user interfaces in a J2EE web server architecture. Previous versions of this white paper included information on WebSphere® 6.1, because support for WebSphere 6.1 ends September, 2013 that information has been removed.

This document is a companion to another whitepaper titled “IBM Power Systems with IBM i Performance and Tuning Tips for Oracle’s JD Edwards EnterpriseOne 9.0 and 9.1” which covers tuning recommendations for the JD Edwards EnterpriseOne application. It can be found at: http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101504.

Oracle support for WebSphere Application Server

At the time this document is being written, Oracle supports the following editions and versions of WebSphere Application Server for use with JD Edwards EnterpriseOne:

- WebSphere Application Server Network Deployment and Express 7.0 (IBM Technology for Java 32 bit JVM and 64bit JVM)
- WebSphere Application Server Network Deployment and Express 8.5 (IBM Technology for Java 32 bit JVM and 64 bit JVM)

Note that the editions and versions supported are dependent on the operating system release and the release of the application. For complete information, please refer to the Minimum Technical Requirements (MTRs) at the following Oracle website which requires a user id and password for access: https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=705335.1

The tuning guidance in this document has been reviewed for all of these options and is expected to remain valid for future versions of WebSphere. References to WebSphere in this document which don’t specify an edition or release should be understood as any edition of WebSphere supported by Oracle for use with JD Edwards EnterpriseOne.

Note that Network Deployment includes additional functionality for clustering, high availability, and load balancing. It can be used to either configure multiple instances of the application server on the same system (“vertical clustering”) or multiple instances on multiple systems (“horizontal clustering”). Vertical clustering can help environments with high JVM memory requirements as it allows separate tuning and management for each JVM. Horizontal clustering is not generally used on IBM i because of the inherent scalability of IBM i and ease of management. WebSphere Base and Express editions do not support this functionality.
IBM HTTP server performance

The IBM HTTP Server for i (powered by Apache) is the product that renders web pages to WebSphere running the JD Edwards EnterpriseOne web client solution. The HTTP server can be started, analyzed, and tuned through a browser interface, the IBM Web Administration for IBM i, or by using IBM i system commands. The browser interface offers ease-of-use features such as selecting options from lists rather than typing in entries to the configuration file. It also offers extensive help text and is the recommended user interface. To start the browser, open IBM System i Navigator and click on network, then servers, then TCP/IP, and in the right pane right click on HTTP administration and click on Start. See Figure 1. The IBM i equivalent command is STRTCPsvR SERVER(*HTTP) HTTPSVR(*ADMIN). Both methods also start the subsystem QHTTPSVR if necessary.

![Figure 1 - Starting IBM HTTP server administration for IBM i from IBM System i Navigator](image)

To access the browser interface, start a Web browser, such as Microsoft® Internet Explorer, and enter the URL http://<server-name>:2001 where <server-name> is the name of the IBM i server. When prompted to sign on to the server, enter a valid IBM i user profile and password.

Configuration changes can be made through the IBM Web Administration for IBM i interface or by directly editing the HTTP configuration file (using the edtf command). There is a configuration file for each HTTP server instance located in the IFS directory /www/<http-server-name>/conf, where <http-server-name> is...
the name of the HTTP server instance. This directory contains a file named httpd.conf which holds all of the configuration information for that instance. The information below describes how to make changes to the configuration through the browser interface and also lists the HTTP configuration keywords and values that can be directly entered into the httpd.conf file.

**Measure and Analyze**

To measure the performance of the HTTP server instance, use IBM i performance tools such as Performance Monitor and Performance Explorer (PEX). To review what is happening in the HTTP server instance, examine the access log which is enabled by default. Because the access log impacts performance, one of the recommendations below is to disable this logging function during normal operations; access logging can always be re-enabled if there is a need to monitor the log or debug problems. The access log is located in the IFS directory /www/<http-server-name>/logs where <http-server-name> is the name of the HTTP server instance.

To more closely examine what is happening while the HTTP server instance is active, the IBM Web Administration for IBM i provides real time statistics for the HTTP Server on IBM i. To view these statistics, click on the “Real Time Server Statistics” link on the left hand pane, as shown in Figure 2.

![Figure 2 - Real Time Server Statistics show current activity on the HTTP server instance](image)

The server statistics screen shows information about the number of connections in use and the number of requests and responses. Monitor the number of active and idle threads during the time that the system is
running its peak load. Adjust the number of threads accordingly, as described in the section below "Set the number of threads for each job." Additional information such as the byte counts sent and received is available on the "Absolute" and "Delta" tabs. Use this information to look at the communications workload running on the system during peak time periods.

Adjust

The HTTP server instance has several settings that can impact the performance of the JD Edwards EnterpriseOne web client solution. Enable these settings one at a time to ensure that each recommended change provides a benefit. Changes can be made to the HTTP configuration in two ways:

- By selecting the HTTP server instance, through the IBM Web Administration for IBM i interface
- By directly editing the configuration file, httpd.conf, located in the IFS (using the edtf command)

To make the changes through the Web interface, select the HTTP server instance from the list as shown in Figure 3. In this example, select WASHTTPSVR from the list and make changes to the settings associated with this instance. (Note that this document uses the default names for Oracle installation. The name of the HTTP server instance(s) is dependent on the installation.)

![Figure 3 - Select the HTTP server instance](image_url)
1) Minimize the logging in the production environment

As discussed earlier, logging is useful during the initial startup of the web client environment as well as for debugging problems. When the production environment is running and stable, however, extraneous logging should be disabled as it consumes a large amount of processing power.

The HTTP server provides two different types of logging: error logging and access logging. The error log can be configured to capture various types of errors. By default it will capture all warnings, errors, and critical conditions. The recommendation is to change this setting so that only errors and critical conditions are captured in the error log. Also, since a separate error log will be created for each day that the HTTP server instance is running, change the configuration to have logs automatically removed after one week, as this reduces the disk requirements for the HTTP server instance.

One way to change the log settings is to select “Logging” from the left pane on the IBM Web Administration for IBM i, then select “Error Logs.” As seen in Figure 4 change the “Expiration” value to “1” week on the error logs page, then select the “Logging Level” pull-down list and select “Error” instead of “Warning.” Click the “Apply” button after making the changes; otherwise the changes are not saved.

![Figure 4 - Change the error log settings in HTTP server instance](image)

Alternatively, change the configuration file by editing it to add the following two lines:

```
LogLevel Error
```

IBM Power Systems with IBM i performance and tuning tips for Oracle’s JD Edwards EnterpriseOne WebSphere-based HTML servers

http://www.ibm.com/support/techdocs

© Copyright 2013, IBM Corporation
LogMaint logs/error_log 7 0

The access log, as the name indicates, captures all HTTP accesses to the system. Each browser that accesses the system via a URL will be noted by an entry in the access log. Depending on the number of users connected to the system, this log can become very large. As a result, the performance recommendation is to disable all HTTP access logging. Since the access log tracks all access to the system, however, it can be used to determine the location of a Web attack. Security considerations need to be balanced with the performance impact of access logging.

One way to disable the access log is to click on the “Custom Formats” tab on the “Logging” page. As seen in Figure 5, click the radio button on each of the log formats and then click the “Remove” button. After all of the log formats are removed, click the “Apply” button. Then click on the “Custom Logs” tab, select the entry for “logs/access_log”, click the “Remove” button, and then click the “Apply” button.

![Figure 5 - Eliminate access logging for the HTTP server instance](image)

An alternate way to disable the access log is to change the httpd.conf file for this instance. Comment out the following lines in the file by putting a number sign (#) in front of each line:

```
LogFormat "%h %l %u %t "%r" %>s %b "%{Referer}i" "%{User-Agent}i"" combined
LogFormat "%{Cookie}n "%{Referer}i" "%{User-Agent}i"" cookie
LogFormat "%{User-agent}i" agent
```

IBM Power Systems with IBM i performance and tuning tips for Oracle’s JD Edwards EnterpriseOne WebSphere-based HTML servers

http://www.ibm.com/support/techdocs

© Copyright 2013, IBM Corporation
LogFormat "%(Referer)i -> %U" referer
LogFormat "%h %l %u %t "%r" %>s %b" common
CustomLog logs/access_log combined

If the system has been running with access logging enabled, deleting the existing logs will clean up system storage. The logs are located in the directory /www/<http-server-name>/logs. Delete all of the files in this directory; the HTTP server automatically creates new files as needed for the instance.

2) Allow unlimited requests per connection

When a browser on a client machine makes a connection to the HTTP server instance, the server maintains (or keeps alive) the connection and allows multiple requests from many browsers to be made on the same connection. It is more efficient to keep the same TCP connection for as many requests as possible. By default, the HTTP server instance will use the same TCP connection for 100 requests. Other platforms sometimes recommend using this limit on the number of requests per connection based on memory concerns but this is not a significant concern for IBM i. The recommendation is to change the setting so that an unlimited number of requests can be sent on the same connection.

Note: This guideline is different on other platforms. Please use caution to ensure that the correct tuning is applied in an environment which utilizes multiple platforms.

One method of changing this setting is to select “System Resources” from the left pane of the IBM Web Administration for IBM i and select the “HTTP Connections” tab as shown in Figure 6. Set the number in the field, “Maximum requests per connection” to 0. Although this may seem counter-intuitive, “0” indicates that the server will allow an unlimited number of requests.
Another way to change this setting is to edit the httpd.conf configuration file by adding the following lines:

```
MaxKeepAliveRequests 0
KeepAlive On
```

### 3) Set the number of threads for each job

The HTTP server job maintains a pool of threads. The number of threads is determined by a configuration setting. The thread pool is used to service all the web browsers that connect to the server job. If all threads within the HTTP server job are in use and a new request arrives, an additional job will be started—with the same number of threads in its pool. The recommendation is to keep the proper number of threads available to avoid the extra overhead of starting another HTTP server job.

Adjust the number of threads that run in each of these HTTP server jobs by selecting “General Server Configuration” from the navigation pane on the left-hand side and then selecting the “General Settings” tab. Change the field for “Number of threads to process requests” and then click the “Apply” button. The default is 40, and lab tests have found that this number of threads is sufficient to adequately handle 400 users doing typical workloads through browser connections. If the system is running more users than this, change this setting accordingly. For example, if the system is being configured for more than 400 users, change the setting to 50 as shown in Figure 7.
Figure 7 - Set the number of threads to handle requests in the HTTP server instance

Alternatively, adjust the number of threads running in each Web server job by editing the httpd.conf file to add the following line:

```
ThreadsPerChild 50
```

Note: This guideline is different on other platforms. Please use caution to ensure that the correct tuning is applied in an environment which utilizes multiple platforms. Also note that there is little harm in specifying a larger number of threads than indicated by this guideline.

4) Enable compression

The HTTP server supports data compression of HTML files delivered to clients that support decompression. This compression of files to the client decreases the data volume sent over the network and improves response times. It is enabled via a DEFLATE filter, which can be configured to only compress files located in certain directories. Remote clients connected to the network over a wide area network (WAN) at slow speeds have the greatest potential for performance gains. Local clients connected at higher speeds will see smaller performance gains. Compression and decompression will utilize more processor resources and are mostly recommended when the network is relatively slow.

Compression must be supported by both the HTTP server and the web client’s browser to take advantage of this feature. HTTP Server for IBM i includes compression support in the base product. Likewise,
Internet Explorer and Netscape Navigator both support this function when running at the versions recommended by the Oracle MTRs.

There are two steps to enable compression:

1. As shown in Figure 8, select “Compression” from the left pane, and then click on the “Output Filters” tab. In the “Set output filter” section, click “Add”, enter “Deflate” as the filter name, and click “Apply”.

   Alternatively, edit the configuration file by adding the following lines:

   ```
   LoadModule deflate_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
   SetOutputFilter DEFLATE
   ```

   ![Figure 8 - Enable compression for the HTTP server instance](image)

2. Next, as shown in Figure 9, in the “Server area” section, select the directory path that ends with “webclient.war” that was added during the JD Edwards EnterpriseOne Java™ Application Server (JAS) configuration. Next select “Compression” from the left pane. Click the “Add” button in the “Add output filter by MIME type” section, then enter a MIME type of “text/html” or select it from the list. Finally, enter a filter name of “DEFLATE” and click the “Apply” button.
Figure 9 - Set the output filter for text in the JAS directories

Alternatively, change the directory directives to include the filter statement for text by adding the line shown in bold, below, to the httpd.conf file:

```
<Directory /QIBM/UserData/WebSphere/AppServer/V6/Base/profiles/default/installedApps/<sys>/<ent-app>/webclient.war>
  Order Deny,Allow
  Allow From all
  AddOutputFilterByType DEFLATE text/html
</Directory>
```

Note: Some implementations may contain a hardware compression device which may not be compatible with this software compression feature. For further information, contact the appropriate hardware service provider.

5) Setup Fast Response Cache Accelerator (FRCA)

Fast Response Cache Accelerator (FRCA) enables the caching of static and dynamic content in a memory-based cache. Requests that are cached by FRCA are processed by the operating system instead by the HTTP server. This can improve performance and scalability of the HTTP server.
Note: the HTTP Server does not check for authorization on content served from FRCA. Use FRCA to cache content that does not need to be secured.

Testing in the lab showed that using FRCA to cache the static content reduces the number of the requests that the HTTP server processed by roughly 30%. This will vary depending on the end user browser configuration, and usage patterns of the application.

There are three steps to setting up FRCA in the Administrative console

1. As shown in Figure 10, select “FRCA” from the left pane, and then click on the “General Setting” tab. In the “Server IP addresses and ports to listen on” section, click on the line with the Port Number for the EnterpriseOne website. Change the value in the selection box for “FRCA” to “Enabled” and click “Apply”.

![Figure 10 - Enable FRCA](image)
2. As shown in Figure 11, select “FRCA File Cache” tab. In the “FRCA file cache capabilities” drop box, select “Enabled” and click “Apply”.

3. The third step is to setup the files that are to be cached during server runtime. The majority of the static content for EnterpriseOne is contained in three sub-directories.
   - `webclient.war\img`
   - `webclient.war\share\images`
   - `webclient.war\share\images\login`

   To configure this, the full path to these locations must be determined. The easiest way to determine the full path is to look at the definition of the jde alias in the configuration file.
a. Figure 12 shows the format of the jde alias in the Configuration File. This will vary for each installation depending on the version, edition and configuration of the WebSphere install.

   - In this example, the path for jde is
     `/QIBM/UserData/WebSphere/AppServer/V7/Express/profiles/E1JAS/installedApps/JDE_E1JAS/AS_JS_80_PD.ear/webclient.war`
b. To configure the file paths, click “Add” under the “Files to cache during server runtime” section as shown in Figure 13.

c. Enter: (substitute your file location as appropriate)
/QIBM/UserData/WebSphere/AppServer/V7/Express/profiles/E1JAS/installedApps/JDE_E1JAS/AS_JS_80_PD.ear/webclient.war/img/*.gif

d. Click “Apply”

e. Do this for the other two file locations:
   * /QIBM/UserData/WebSphere/AppServer/V7/Express/profiles/E1JAS/installedApps/JDE_E1JAS/AS_JS_80_PD.ear/webclient.war/share/images/*.gif
   * /QIBM/UserData/WebSphere/AppServer/V7/Express/profiles/E1JAS/installedApps/JDE_E1JAS/AS_JS_80_PD.ear/webclient.war/share/images/login/*.jpg

f. When finished the configuration will look similar to that shown in Figure 14. Note the HTTP Server needs to be restarted for this to take effect.

Figure 14 - FRCA File Cache Setup

NOTE: When you view the httpd.conf file, ensure that the directive to turn FRCA on is located before the FRCACacheLocalFileRunTime directive as shown in the example below. If FRCAEnableFileCache ON directive is not located in the correct place, FRCA may not function correctly.
To validate the setup, you can configure FRCA logging

4. Navigate to the “FRCA Logs” tab, as shown in Figure 15. Click “Add” in the “FRCA Logs” section. Enter “logs/frca.log” in the “Log” field, and “%t,%r” in the “Attributes” field. Click “Apply”. This will create a log file in /root/www/<httpserver name>/logs. The log file will contain an entry with the e time and url for each item that was cached by FRCA. You should only use this logging to ensure that FRCA is working properly, as it can generate a large amount of data. The HTTP Server needs to be restarted for logging to take effect.
6) Adjust HTTP Server subsystem settings in IBM i

As shipped, the HTTP subsystem jobs run at priority 25. For smaller systems, this may work fine. However, in very busy systems, with lots of database activity and CallObject kernel activity; it is possible the HTTP jobs can be starved for CPU. A good reference to help determine this is published in IBM tips TIPS0051. See Appendix A for the reference. The following are the steps to change the priority:

1. Use the Display Subsystem Description (DSPSBS) command to display the subsystem where QHTTPSVR is running. The command is: DSPSBS SBSD(QHTTPSVR/QHTTPSVR)
2. Take option 7 – Routing Entries. This will display the routing entries for the subsystem. By default, there is only one routing entry with sequence number 10. The remaining instructions assume a default configuration.
3. Take option 5 – Display to display the routing entry. This will display the details for the routing entry as shown in Figure 16 below.

![Figure 16 – QHTTPSVR routing entry](image)

4. The class for the routing entry determines the priority for the jobs running in the subsystem. Therefore, the class QZHBHTTP in library QHTTPSVR should be changed.
5. Create a backup of the QZHBHTTP subsystem with the CRTDUPOBJ command: CRTDUPOBJ OBJ(QZHBHTTP) FROMLIB(QHTTPSVR) OBJTYPE(*CLS) TOLIB(QHTTPSVR) NEWOBJ(QZHBHTTPBK)
6. The following command will change the priority for the HTTP Server jobs to be equal to the priority of the WebSphere and Enterprise Server jobs (priority 20): CHGCLS CLS(QHTTPSVR/QZHBHTTP) RUNPTY(20). Some environments may benefit using a priority higher than 20, so you should adjust the priority accordingly.
7. After restarting the subsystem for these changes to take effect, use the command WRKACTJOB SBS(QHTTPSVR) to look at the running jobs. Validate the jobs are running at the desired priority.

**WebSphere Application Server tuning**

The WebSphere Application Server product runs the JD Edwards EnterpriseOne HTML server support in a server instance, created at install time, which is named AS_JS_xx, where xx is the port number specified as part of the installation process. The WebSphere Application Server is a general-purpose J2EE environment capable of running many different types of Java-based applications with JD Edwards EnterpriseOne HTML server code being only one of them. The HTML server is the application code which is a set of Java servlets and Java Server Pages (JSPs) that is used to serve as the JD Edwards EnterpriseOne application interface to web users. Each WebSphere Application Server instance runs in its own Java Virtual Machine (JVM). This document contains tuning considerations for WebSphere Application Server 7.0 and 8.5. It is expected that these guidelines will remain appropriate for future releases of WebSphere.

**Measure and analyze**

The WebSphere Application Server, in general, and the JAS server code, specifically, can be analyzed in a variety of ways. First, the traditional performance tools can be used to understand utilization of CPU, disk, memory, and other resources. These include Performance Monitor, Performance Explorer, and IBM System i Navigator Collection Services. Secondly, IBM i provides command interfaces such as Work JVM Jobs (WRKJVMJOB) which can be used to look at heap usage, garbage collection, thread stacks, and object statistics. Thirdly, JAS and WebSphere both have built-in tools to support performance analysis.

Prior to the availability of Server Manager for JD Edwards EnterpriseOne as described below, JAS includes the System Administration Workbench (SAW) which can be used to analyze JAS performance including response times and statistics for business functions. See the JD Edwards EnterpriseOne Application Tuning section for details. WebSphere Application Server 7.0 and 8.5 includes the Tivoli® Performance Viewer which can be used to analyze thread and heap resources for JAS using WebSphere ND with Deployment Manager.

Oracle support includes the Server Manager for JD Edwards EnterpriseOne product starting with JD Edwards EnterpriseOne Tools 8.97. It also replaces SAW and provides new interfaces for configuration and administration. This document includes some examples of how it can be used to manage some settings. For more information, contact Oracle or the appropriate service provider.

**Adjust**

Changes to the WebSphere configuration are made through the WebSphere Administrative Console using a web based interface. The default instance of WebSphere, server1, is accessed through the following URL:

http://<sys>:9060/ibm/console

where <sys> is the name of the IBM i server that is running WebSphere.
The administrative console presents a user id prompt, which is used to track the changes entered into the administrative console. It does not have to match an IBM i user profile.

1) Adjust the heap size for the JVM

On IBM i servers, Java performs memory management for applications running within the JVM. It allocates memory on behalf of the application in a single heap space and cleans up unused memory via the garbage collection process. WebSphere, as a Java application, has its memory managed by the JVM. When the application server running JAS is started through WebSphere, Java allocates memory and manages it on behalf of JAS. Setting the initial and maximum heap sizes for the JVM helps manage the amount of memory used by WebSphere.

The initial heap size setting for JAS is based on the number of active users. The guideline derived from lab tests is 1.5 megabytes of memory per active user. The active user count is defined as the number of concurrent users who are clicking OK or other button two to three times per minute. The appropriate setting may vary depending on the type of applications being run as well as the number of users.

For WebSphere 7.0 and 8.5, the recommended values for initial and maximum heap size are influenced by three factors. First, the JD Edwards EnterpriseOne HTML Server running on both WebSphere 7.0 and 8.5 supports a 32-bit Java implementation. Because 32-bit pointers are smaller than 64-bit pointers, most applications see a reduction in main storage requirements. Secondly, because of the 32-bit Java implementation, it can address a maximum of 4 GB of memory. Some of this is reserved for use by the operating system and the JVM implementation. Many applications make use of additional heap memory which further affects how much can be allocated to the WebSphere JVM. IBM evaluations indicate that the effective maximum heap size value for JD Edwards EnterpriseOne is 1744 MB. Note that the maximum value may have to be reduced based on the workload. Thirdly, the general guideline for WebSphere 7.0 and 8.5 is to specify a maximum heap size that is at least four times the initial heap size. This leads to the following recommendations:

- If the environment is not memory constrained, specify a maximum heap size of 1744 MB and an initial heap size of 436 MB (e.g. one fourth of the maximum heap size).
- If the environment is memory constrained, specify an initial heap size of 1.5 MB per user or 256 MB, whichever is larger. Specify a maximum heap size of four times the initial heap size.
- In any environment, do NOT run with the default heap size values which are 32 MB for initial heap size and 768 MB for maximum heap size. This small initial heap size would lead to sub-optimal performance due to garbage collection in a JD Edwards EnterpriseOne environment.

Note: This guideline is different on other platforms. Please use caution to ensure that the correct tuning is applied in an environment which utilizes multiple platforms. This parameter is especially important as most platforms recommend setting a maximum heap size which can potentially hurt the performance of WebSphere on IBM i.

As always, the appropriate settings may depend on the type of applications being run and the system environment.

To change the JVM settings for the application server, follow these steps: expand Servers > click Application Servers > click the desired Server Name > click Java and Process Management > click Process Definition> click Java Virtual Machine. Figure 177 shows the values recommended.
Figure 17 - Set the initial and maximum heap size for the JVM in WebSphere

Heap size and garbage collection information may be checked with the PRTJVMJOB command. This command produces a spooled file that contains information on the current heap size, the number of garbage collections performed, and the duration of the garbage collections. At times, it may be helpful to collect additional data about the garbage collector as it is running. This can be accomplished with verbose garbage collection which records several pieces of information into System.out each time the collector runs. This includes the current heap size, as well as the number and size of objects collected, number of objects in the heap, amount of time the collector ran and other information. The PRTJVMJOB command can be intrusive and impact JVM performance. The WRKJVMJOB command provides higher level information and is less intrusive.

The number of users that can be supported by a single JVM is difficult to quantify. In controlled benchmark environments, JD Edwards EnterpriseOne has been run with up to 1600 users in a single JVM but that is not a realistic assumption for a production environment. In practice, feedback from customers using web-based clients indicates a maximum of 150 to 200 users.

2) Adjust the number of threads running in the JVM

WebSphere maintains a pool of threads that processes requests from web clients. Two things limit the pool: the minimum number of threads started and the maximum number of threads allowed in the pool.
In WebSphere 7.0 and 8.5, testing in the lab and benchmark tests show that using a thread pool equal to 10-15% of the number of users supported in the JVM is generally sufficient.

To navigate to the panel for thread settings, use the following path: expand Servers > click Application Servers > click the desired Server Name > click Additional Properties > click Thread Pools> click WebContainer.

Figure 18 shows the recommended values for a maximum workload of 200 users for WebSphere 7.0 and 8.5. The minimum and maximum size is set to 10% of the total number of users (or 20 threads). The checkbox “Allow thread allocation beyond maximum size” is checked.

![Figure 18 - Set the number of threads for the JVM in WebSphere 7.0 and 8.5](image)

3) Enable localhost name caching

The versions of the IBM SDK for Java used by WebSphere 7.0 and 8.5 do not cache the result of the static method java/net/InetAddress.getLocalHost. This method is used throughout the JD Edwards EnterpriseOne HTML Server code, WebSphere and the JDBC driver code. If the localhost address of a process will not change while it is running, then it is advised to use a built-in cache for the localhost lookup by setting the com.ibm.cacheLocalHost system property to the value “true”. Upon first call to the getLocalHost() method, the ip address of the hostname is stored in a static string. Later calls will use this string if it exists.

Setting this system property is particularly helpful if both IPv4 and IPv6 traffic is being used on your network. In these environments a full network DNS lookup is performed every time the getLocalHost() method is called. This has been demonstrated to significantly improve the performance of the JD Edwards EnterpriseOne HTML Server code.

Note: The address for servers configured using DHCP change over time. Do not set this property unless you are using statically assigned IP addresses for your server.
Figure 19 shows how you can set the JVM argument for this system property in the WebSphere Administrative Console. To navigate to the panel for custom properties for the JVM use the following path: expand Servers > Server Types > Application Servers > click the desired server name > Java and Process Management > Process Definition > Java Virtual Machine > Custom Properties. Press the New push button and enter in com.ibm.cacheLocalHost as the name and true as the value.

**Figure 19 - Enable localhost name caching**

4) **Adjust HTTP transport channel settings**

The HTTP transport channel within WebSphere manages incoming requests from web clients and I/O processing. Most JD Edwards EnterpriseOne webclients will issue many requests across the same connection. Tuning the HTTP transport to handle many requests prevents the connection from being closed too soon and requiring a new connection be established.

Two things control this behavior. The use persistent (keep-alive) connections setting controls whether connections are left open between requests or if connections will be closed immediately. Leaving the connections open can save setup and teardown costs of sockets. The default value for this setting is “true”, which is the optimal setting. The maximum persistent requests parameter controls the number of requests that can flow across the connection before it is reset. Increasing this parameter from the default setting also eliminates unnecessary setting up and tearing down of sockets.

To change this setting in the WebSphere administrative console, navigate to the correct panel: click Servers > Server Types > WebSphere application servers > click the desired server name > Ports. Then click View associated transports for the appropriate port (e.g. WC_defaulthost). Select the transport chain whose properties you are changing (e.g. WCInboundAdmin). Select the HTTP transport channel. As Figure 20 shows, ensure that the checkbox for “Use persistent keep alive connections” is checked and increase the “Maximum persistent requests per connection” from the default value of 100 to a value that handles the number of requests from your webclients without needing to be reset (e.g. 10,000 requests). In some cases using the unlimited persistent requests may be appropriate.
5) Adjust load balancing settings for WAS ND

When running WebSphere Application Server Network Deployment with multiple nodes in a cluster, the number of JD Edwards EnterpriseOne users may not be evenly distributed across all of the nodes. One node may have many more users than the other nodes in the cluster. Ensuring that all of the nodes in a cluster have as close to the same number of users as possible can help provide for consistent performance for all users.

As shipped, the algorithm used by WAS ND to distribute users is called Round Robin. This algorithm distributes users to each node in the cluster based on a weighting according to the number of requests that have been processed. This can become unbalanced. For example, if WAS ND running the JD Edwards EnterpriseOne web server code is started and a single user signs onto the system and does many transactions, WebSphere provides a much higher weighting to the other nodes in the cluster to balance the workload. If many users then log onto the system and each perform only a single request, these nodes will eventually end up with many users while the first node remains with a single user.

To overcome this issue and provide a more evenly distributed number of users per node, the following changes can be made to the WebSphere settings. The algorithm used is changed from Round Robin to Random and session affinity can be turned off.

To change the algorithm in the WebSphere administrative console, navigate to the correct panel: click Servers > Server Types > Web Servers > click the desired name > Plug-in Properties > Request Routing. Using the pull-down menu change the Load balancing option from Round Robin to Random as shown in Figure 21. After making the change, click Apply and then save the settings to the master configuration file.
Figure 21 - Change the load balancing option to “Random”

To change the session affinity setting in the WebSphere administrative console, navigate to the correct panel: click Servers > Server Types > Web Servers > click the desired name > Plug-in Properties > Custom Properties. Click the New button to add a new property. For the property name enter IgnoreAffinityRequests and for the value enter false as shown in Figure 22. Click apply, then save to the master configuration.

Figure 22 - Change the session affinity settings

Because these changes are placed into the plugin-cfg, the file will need to be regenerated. To do this in the WebSphere administrative console, navigate to the correct panel: click Servers > Server Types > Web...
IBM Power Systems with IBM i performance and tuning tips for Oracle’s JD Edwards EnterpriseOne WebSphere-based HTML servers

http://www.ibm.com/support/techdocs

6) Adjust WebSphere subsystem settings in IBM i

As shipped, the WebSphere subsystem runs in the *BASE memory pool on the system. For smaller systems or systems with limited memory, the system may run more efficiently using the *BASE memory pool. In many cases, however, it is helpful to divide workloads into separate memory pools. Putting WebSphere into a separate memory pool isolates it from other workloads and ensures it does not compete with other jobs for memory. This can be done either with a private pool belonging to the WebSphere subsystem or by using a shared memory pool which provides more flexibility as it can be managed using the combination of QPFRADJ and WRKSHRPPOOL as discussed above. Using a shared memory pool is the recommended technique and is described below. For information on how to use the alternative technique using a private pool, please refer to the IBM i Information Center website listed in Appendix A. Note that these pool configuration changes can sometimes be implemented on an active environment but it is recommended that this be done while the environment is offline to minimize performance disruptions during movement of the memory resources.

The following are the steps to configure WebSphere to run in a separate pool:

1. Use the Display Subsystem Description (DSPSBSD) command to display the subsystem where WebSphere is running. For WebSphere 7.0, this is normally the subsystem description QWAS7 in library QWAS7. The command is: DSPSBSD SBSD(QWAS7/QWAS7)
2. Take option 2 – Pool Definitions to display the pool configuration for the subsystem. By default, one subsystem pool with *BASE in the Storage Size column is shown.
3. The remaining instructions assume a default configuration. If unable to verify which subsystem description is being used for WebSphere or if the pool configuration is different than described above, stop here. The instructions for this environment would need to be identified by a service provider or someone with appropriate IBM i work management skills.
4. The next step is to identify the shared memory pool to be used via the WRKSHRPPOOL command. If running a default configuration, *SHRPOOL1 is available. If it is in use, any available shared pool can be used in its place.
5. The next step is to determine the initial amount of storage and the activity level for this pool. For any version of WebSphere, the recommendation is to allocate four times the initial heap size per JVM plus 1 to 1.5 GB for additional overhead. In the example environment for 200 production users described above, initial heap size was 300MB. Four times that value is 1.2GB plus approximately 1GB for additional WebSphere needs, which gives a total of 2.2GB. These values are additive and should reflect the needs of all JVMs in use. The activity level is the maximum...
number of processes or threads active in this memory pool at the same time. A good starting point for this value is 1.5 times the number of users. For this example, the recommended value for 200 users would be 300. This value is independent of which version of WebSphere is being used.

6. An additional recommendation is to set upper and lower bounds for the pool size using the WRKSHRPPOOL command. This provides the automatic performance adjustment enabled by the QPFRADJ system value with boundaries to ensure that the pool does not get too small or too large based on system activity. The values will depend on the configuration; a reasonable starting point is to specify minimum and maximum sizes that are 20 percent smaller and larger than the value determined in the previous step. This is done because JVM Garbage Collection requires sufficient memory to perform its processing. Allowing the pool to become too small can potentially cause performance problems.

7. The following command can be used to define the shared pool:
   CHGSHRPPOOL POOL(*SHRPOOLn) SIZE(x) ACTLVL(y) PAGING(*CALC) TEXT('WebSphere Pool') where n is the number of the shared pool, x is the amount of memory allocated, and y is the activity level determined in the previous steps. Note that this storage is not allocated until the pool is in use by an active subsystem. It will be taken from *BASE either when an active subsystem is changed to use it or when an inactive subsystem is started after being changed to use it. Note that both the size and the activity level can be adjusted automatically if the QPFRADJ system value is set to allow automatic adjustment or they can be set manually using WRKSYSSTS.

8. The WebSphere subsystem must be changed to make use of the shared memory pool. Note that each subsystem refers to its subsystem memory pools as 1, 2, etc. but this value does not correspond to the system pool numbers shown on WRKSYSSTS. The Work with Subsystems (WRKSBS) display shows the relationship between subsystem pools and system pools. The CHGSBSD command is used to indicate which pools are to be used by the subsystem. Although it is not strictly required, it is recommended that subsystem pool 1 should remain *BASE. The following Change Subsystem Definition (CHGSBSD) command will update the subsystem to reflect the new memory pool:
   CHGSBSD SBSD(QWAS7/QWAS7 POOLS((1 *BASE) (2 *SHRPOOLn)) where *SHRPOOLn is the shared pool identified above.

9. Next, each routing entry in the subsystem must be updated to reference the new memory pool. Again, enter DSPSBD to display the subsystem description. Take option 7 to Work with Routing Entries. For each routing entry, enter the following command:
   CHGRTGE SBSD(QWAS7/QWAS7 SEQNBR(9999) POOLID(2)

After restarting the subsystem for these changes to take effect, use the command WRKSYSSTS to look at the new pool. Validate the activity level setting for the shared memory pool by using the WRKSYSSTS command, pressing the F11 key, and verifying that the “wait to ineligible” value (Wait->Inel) remains at zero. If this value is significantly greater than zero, the activity level needs to be increased further.

The same general procedure can be used to allocate a separate memory pool for JD Edwards EnterpriseOne UBE jobs. In any case, it is not recommended to run WebSphere and UBE work in the
same memory pool. Like WebSphere, the subsystem running UBEs should be configured with a paging option to *CALC.

**Known issues and optimizations**

This section describes known issues with WebSphere Application Server and tuning opportunities that have been observed in customer and benchmark environments. Unless otherwise noted, these items are applicable to all supported levels of WebSphere Application Server.

**Out of memory condition when viewing large pdf's**

This problem was reported by customers who found that viewing a very large pdf through EnterpriseOne web client interfaces could trigger a JVM OutofMemory condition which brought down the JVM. This is most likely when the size of the pdf exceeds the available storage to the JVM. For example, an IBM Technology for Java 32 bit JVM can address a maximum of 4GB and would not be able to allocate enough storage to view a 10GB pdf. Internally, WebSphere Application Server uses asynchronous data transfer which may require an excessive number of buffers to send data over a TCP/IP connection. This condition can occur in other situations involving file transfers such as a dmgr instance updating the nodeagents and so on.

This growth in storage requirements can be observed using the Work with JVM Jobs (WRKJVMJOB) command and taking option 5 to view one of the HTML server JVM's to look at its garbage collection information. Figure 24 shows sample output from this command.

![Figure 24 - Garbage collection information from the WRKJVM command](image)

Before applying the change, you would see the internal memory size go up by the size of the pdf being viewed plus additional overhead. For example, when viewing a 100MB pdf the internal memory size would increase by 110 to 120MB. Note that it would also go down again when the pdf is closed. Attempting to view a 10GB pdf would result in an OutOfMemory condition and failure of the JVM.
This can be corrected by adding a Custom Property called com.ibm.ws.webcontainer.channelwritetype and setting its value to sync. This is shown in Figure 25. Note that the HTML server must be restarted to take effect.

![Figure 25 - Custom property for channelwritetype](image)

For additional information on this problem, please refer to the following URL:
http://www-01.ibm.com/support/docview.wss?uid=swg21317658

**Contestion on shareclasses cache**

In an effort to minimize storage requirements, WebSphere Application Server uses a shareclasses cache so that the class information can be shared by multiple JVM’s within the WebSphere version and release. In some cases, contention on this cache can result in severe performance problems. This was first observed in benchmark environments where large numbers of users attempt to sign on to EnterpriseOne at the same time. The first JVM to start signing on users performed normally while the second and subsequent JVM’s saw very slow performance resulting in timeouts and excessive response times. Internally, this was called the “winner/loser problem” as the first JVM was a winner in terms of achieving acceptable performance while the others were losers in that regard.

There are two possible solutions to this problem. The first was devised and tested during benchmark runs. It changes the JVM configuration to ensure that each JVM uses its own shareclasses cache. You do this by specifying Generic JVM options as part of the WebSphere Application Server Configuration. For example, -Xshareclasses:name=jde1cache,groupAccess,nonFatal -Xscmx50M indicates that this JVM should use a cache named jde1cache with a maximum size of 50MB. Note that a unique cache name must be specified for each JVM to avoid conflict. The name itself can be anything convenient, for example it could be the name of the JVM. The second option is to specify –Xshareclasses:none which completely
disables this caching. This option has not been tested in the benchmark environment. Figure 26 and Figure 27 show how to configure the first option.

Figure 26 - Java Virtual Machine configuration
Plugin scaling issue

This problem has only been observed internally in large scale testing of EnterpriseOne on IBM i. It occurs when more than 200 simultaneous threads are active on a single JVM and was observed on high end scaling measurements with 4000 users on 8 nodes in a clustered configuration with approximately 500 users per node. This is probably not typical for most EnterpriseOne installations which generally have fewer than 200 users per JVM.

This problem can be identified by looking at the WebSphere Application Server plugin log and checking for 3450 errors. To find the plugin log, first look at the HTTP configuration file as shown in Figure 28 to find the location of the plugin.
IBM Power Systems with IBM i performance and tuning tips for Oracle’s JD Edwards EnterpriseOne WebSphere-based HTML servers

http://www.ibm.com/support/techdocs

Figure 28 - Finding the plugin location from the HTTP configuration file

Note that the name of the plugin configuration file is too long to fit in the window display and it will be necessary to scroll to the right to see the entire name. In this example, the path for the configuration file was:

/QIBM/UserData/WebSphere/AppServer/V61/ND/profiles/st01/config/cells/ERP6MLX4_st01/nodes/ERP6MLX4_st01/servers/IHS_E1HTTP100/plugin-cfg.xml

“st01” was the name of the profile under which this HTML server instance was created. The plug in log will be found under the directory “logs” in a directory sharing the same name as the HTTP server plugin. In this example, the log can be found using the following path.

/QIBM/UserData/WebSphere/AppServer/V61/ND/profiles/st01/logs/IHS_E1HTTP100/http_plugin.log

Edit this log and look for 3450 errors. This can be done using a mapped network drive from Windows, using the Edit File (EDTF) command, etc.

If you do see this problem, contact IBM service to receive information on fixes. Reference APAR PM16591 which describes the issue as it occurred in WebSphere Application Server 7.0. The following url is a direct link to the APAR: http://www-01.ibm.com/support/docview.wss?uid=swg1PM16591

Note: This problem APAR has been fixed and the PTF is available in Fix Pack 13 for Websphere 7.0.

Before applying any fix packs, check Oracle’s MTRs to ensure that they are supported.
Additional garbage collection optimization with gencon

IBM Technology for Java JVM’s offers additional options than Classic JVM’s for managing garbage collection. One of these is called “generational concurrent” which is abbreviated as “gencon.” It divides the JVM heap storage between short lived and long lived (also known as tenured) objects. It also uses different garbage collection techniques. For applications that have many short lived objects, it can result in shorter garbage collection pauses while still providing good throughput. (Note: For WebSphere 8.5 gencon is the default garbage collection policy, therefore this tip applies specifically to WebSphere 7.)

For large scale benchmark tests, IBM specified the following gencon settings in the HTML server JVM’s:

-Xgcpolicy:gencon -Xmns256m -Xmnx1024m

The two parameters specify the minimum and maximum storage for the short lived portion of the JVM heap. This allows the JVM to determine the best fit for the amount of storage required for short lived objects. Specifying this value in internal benchmark testing showed a 4 percent reduction in processor utilization and 8 percent improvement in response time. Figure 29 and Figure 30 show how to configure this option.

Figure 29 - Java Virtual Machine configuration
Note that multiple options can be specific in the Generic JVM arguments. For example, to follow the guidance for both the shareclasses and gencon garbage collection, set the Generic JVM arguments to:

-Xshareclasses:name=jde1cache,groupAccess,nonFatal -Xscmx50M -Xgcpolicy:gencon -Xmns256m -Xmnx1024m

NOTE: Additional tuning of Java garbage collection may yield more benefit depending on your environment. Experiments with larger nursery sizes would be one possible area to explore.

**Tuning garbage collection threads**

By default the classic JVM used in WebSphere 7.0 will use multiple threads to perform garbage collection. The default for garbage collection will be the number of logical processors on the system. On systems which use simultaneous multi-threading (SMT), there can be multiple threads or logical processors per physical processor. For example, POWER7 hardware could have 8 physical cores and 4 threads per core. There would then be 32 logical processors. By default the number of threads used for garbage collection would then be 32. A best practice is set the garbage collection threads equal to the number of physical cores. In the POWER7 example above, you would set the number of threads for garbage collection equal to 8.

To set this, you will use the -Xgcthreads argument in the Generic JVM Arguments field.

Specify -Xgcthreads as follows:

-Xgcthreads<number of processors>

NOTE: Do not add a space between -Xgcthreads and the value for the number of processors.
Figure 31 shows how to configure the number of garbage collection threads for 8 processors. `-Xgcthreads8` specifies `-Xgcthreads` with 8 processors.

![Generic JVM arguments for garbage collection threads](image)

**Figure 31 - Generic JVM arguments for garbage collection threads**

### Changing to the Toolbox JDBC Driver

In previous versions of this guide, the Native JDBC driver has been recommended for customers running in an All-In-One environment to provide the best performance. In recent work done with a customer, it has been demonstrated that the Toolbox JDBC driver provides for greater stability. In particular if a database query is running for a very long period of time and a system administrator ends the QSQSRVR job or if a database job should end unexpectedly, the JDBC driver waits on a request for a job that has abnormally ended. The Toolbox JDBC driver has additional error checking capabilities and recovers from these types of errors with a more appropriate error message.

Additionally, performance tests have been run comparing the Native and Toolbox JDBC drivers and there is no longer any appreciable difference between overall response times or resource utilization between the two drivers. In an upcoming JD Edwards EnterpriseOne service pack the default setting for the JDBC driver will be changed from Native to Toolbox. Alternatively, the change can be made to your JD Edwards EnterpriseOne web server settings. To change this setting, sign on to the Server Manager, select your JD Edwards EnterpriseOne HTML Server, and then select JDBJ Database Configuration from the left-hand navigation panel. Scroll down to the JDBj Runtime Properties and uncheck the box for Native JDBC Driver on AS400 as shown in Figure 32.
TCP/IP settings

Depending on network, or line, speed, TCP/IP performance can often be improved by setting larger send/receive buffers. The TCP/IP attributes for the send and receive buffer sizes ship with the defaults set to 8192 bytes. These defaults correspond to an Ethernet line speed of 10 million bits per second. Set the TCP/IP attributes to 65536 if line speed is 100 million bits per second in full duplex mode to speed up performance, especially in a V3T environment. Similarly, change the buffers to 1048576 if line speed is 1 gigabit per second with duplex set to *auto detect. The changes take effect immediately. An example from the IBM i command line is: DSPLIND <Ethernet line name> to determine line speed and duplex characteristics. CHGTCPA TCPRCVBUF(65536) TCPSNDBUF(65536). As always, these changes need to be tested and evaluated for each environment.
Appendix A – additional information

Contact information

Please send questions or comments via email to IBM Oracle International Competency Center at: ibmoracl@us.ibm.com

References

IBM i

IBM i Information Center (IBM i 6.1, and i 7.1):
http://publib.boulder.ibm.com/infocenter/iseries/v6r1m0/index.jsp
http://publib.boulder.ibm.com/infocenter/iseries/v7r1m0/index.jsp

WebSphere and Java

WebSphere 7.0 Information Center:
http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp

WebSphere 8.5 Information Center:
http://publib.boulder.ibm.com/infocenter/wasinfo/v8r5/index.jsp

Additional Resources

IBM eServer iSeries Guidelines for Monitoring High Priority Job Performance (TIPS0051):

For the “IBM Power Systems with IBM i Performance and Tuning Tips for Oracle’s JD Edwards EnterpriseOne 9.0 and 9.1” whitepaper, please visit:
http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101504

For all other IBM whitepapers, please visit:
http://www-03.ibm.com/support/techdocs/atsmastr.nsf/Web/TechDocs

For the “IBM Technology for Java Virtual Machine in IBM i5/OS” redbook, please see:

For all other IBM Redbooks, please visit: http://ibm.com/redbooks

For specific information on JD Edwards Solutions from Oracle on IBM i, please visit:

For JD Edwards EnterpriseOne Tools 8.9x Minimum Technical Requirements for Web Servers, please visit:
https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=705335.1
(userid and password required)
Trademarks and special notices

© Copyright. IBM Corporation 2011. All rights reserved.

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

IBM, the IBM logo, Power Systems, Tivoli and WebSphere are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows Server, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

The information provided in this document is distributed “AS IS” without any warranty, either express or implied.

The information in this document may include technical inaccuracies or typographical errors.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.
Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.