

# IBM TIVOLI SERVICE MANAGEMENT PRODUCTS VERSION 7 BEST PRACTICES FOR SYSTEM PERFORMANCE WHITE PAPER

Version 1.6

**Note:**

Before using this information and the product it supports, read the information in [“Notices”](#) on page 45.

This paper applies to the following IBM Tivoli service management products:

- IBM Maximo Asset Management 7.1
- Tivoli Asset Management for IT 7.1 and 7.2
- IBM Tivoli Change And Configuration Management Database 7.1 and 7.2
- IBM Tivoli Provisioning Manager 7.1 and 7.2
- IBM Tivoli Service Automation Manager 7.1 and 7.2
- IBM Tivoli Service Request Manager 7.1 and 7.2

© Copyright IBM Corporation 2011.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Table of Contents:

Chapter 1: Introduction.....	1
Best Practices Is a Cooperative Effort.....	1
Quality Assurance and Testing .....	1
Factors in System Performance .....	2
The IBM Tivoli service management products Infrastructure.....	3
Building Performance Improvements into IBM Tivoli service management products.....	4
New performance features .....	4
Disable doclink on load.....	5
Integration and Batch Processing.....	5
Chapter 2: System Architecture and Application Server Configuration .....	6
Terminology.....	6
Basic System Configuration .....	6
Advanced Enterprise Configuration.....	7
Deploying IBM Tivoli service management products in Multiple Clusters .....	8
Isolating User Applications.....	8
Isolating Reporting .....	9
IBM WebSphere Application Server Version 6.1 Tuning .....	9
IBM HTTP Server Version 6.x Tuning Suggestions .....	10
Managing Integration Load .....	11
Load Balancing.....	11
Chapter 3: Scheduled Tasks (cron Tasks).....	13
Configuring Cron Tasks in Version 7 .....	13
Chapter 4: Integration .....	14
Integration and Batch Processing, version 7.....	14
Performance Issues in Integrated Systems.....	15
Clustering with the Integration Framework.....	15
Recommended Approach to Clustering.....	15
Using Exception-Handling Queues .....	15
Using a Loop-Back Exception Queue Design .....	16
XML/HTTP and SOAP-based Web services for Inbound processing .....	16
Multi-record inbound XML message processing .....	16
Using the Sync operation with Action Add.....	17
Using Message tracking.....	17
Chapter 5: Reporting.....	18
Chapter 6: Tuning the Database and Tuning SQL .....	19
The Database .....	19
Database Connection Pool.....	19
Database Tuning.....	20
Indexing.....	20
Special Index Types.....	20
Statistics .....	20
Customization.....	23
Moving a Collocated Database.....	23
Database Compression .....	23
Oracle Database Notes.....	23
IBM DB2 Notes .....	23
SQL Server Database Notes .....	25
User Queries .....	25
Setting the Appropriate Search Type.....	26
EXACT Search Type.....	26
TEXT Search Type.....	26
WILDCARD Search Type.....	26
NONE Search Type.....	26
Limit query times to improve performance.....	27
Resetting the DATETIME Data Type of a Column .....	27
Determining What Users Will Query .....	27
List Panel Order by Queries .....	28

Restricting Querying in Applications .....	28
Database Maintenance.....	28
Key Performance Indicators .....	28
Live KPIs on Start Centers .....	28
KPI Queries .....	28
KPI cron Task Frequency.....	28
KPI Best Practices.....	29
Escalations.....	29
Efficiency and Frequency.....	29
Reports .....	29
Properties file (7.1.1.6 and later) .....	29
Load Testing .....	29
Chapter 7: Network and Bandwidth .....	30
IBM Tivoli service management products on a Network.....	30
Using Citrix or Windows Terminal Server.....	30
Using compression techniques to improve performance .....	30
Hardware compression using network appliances .....	30
HTTP compression .....	30
Runtime gzip compression .....	31
Image and JavaScript Browser Caching .....	31
Use Quality-of-Service Guarantees for Network Traffic.....	31
Chapter 8: Server OS Configuration .....	32
AIX Configuration.....	32
RedHat Linux Configuration.....	32
Windows Configuration.....	32
Chapter 8: Client Workstation Configuration.....	33
Chapter 9: Performance Improvement Tips and Customer Suggestions .....	34
Apply the Latest Patch.....	34
Staged Roll-out for New Users .....	34
Start Center Portlets.....	34
Start Center Result Set and Pre-defined Queries .....	34
Client Workstation Suggestions .....	34
Limit or Prevent Some Workstation Activities and Processes.....	34
Monitor the Network for Streaming Audio and Video.....	34
Have Only One Network Link Active .....	34
Monitor for Hung Processes and Applications .....	34
Garbage Collection Parameters for WebSphere Application Server .....	35
Limiting Use of E-Audit.....	35
Archive and delete historical data .....	35
Chapter 10: Troubleshooting .....	36
Documents or Files to Collect for Troubleshooting .....	36
Troubleshooting Performance Problems.....	36
Setting Up a Stand-alone Application Server for Debugging.....	36
Debugging Parameters.....	37
Loggers:.....	37
Displaying Garbage Collection Statistics on the Server .....	37
Troubleshooting Performance Issues in Application Server Configuration.....	38
Monitoring the Issues.....	38
IBM Tivoli Monitoring Agent .....	39
Addressing the Issues.....	39
Appendix A: Setting up Two Clusters .....	40
Overview.....	40
Example: Setting Up the UI Cluster and the Q Cluster .....	40
Steps to Set Up the UI Cluster and the Q Cluster .....	40
Result of Setup .....	42
Accessing the Sequential Queue.....	42
Example Deployment and Configuration .....	42
References.....	44
Notices .....	45
Trademarks .....	46

## Table of Figures:

Figure 1. IBM Tivoli service management products infrastructure pyramid.....	3
Figure 2. System Infrastructure Pyramid Tuning .....	4
Figure 3: Basic Configuration.....	7
Figure 4: Advanced Enterprise Configuration .....	8
Figure 5: Overview of integration .....	14
Figure 6: Garbage Collection Statistics.....	38
Figure 7: System Layout .....	42
Figure 8. Multiple Cluster Configuration.....	43

## Table of Tables:

Table 1. WebSphere Application Server V6 JVM Optimization .....	9
Table 2. WebSphere Application Server V6 Thread Pool Optimization .....	10
Table 3. IBM HTTP Server Version 6 Optimization .....	10
Table 4: Database Pool Properties .....	19
Table 5: Performance Problems .....	39



# Chapter 1: Introduction

IBM Tivoli service management products have a long and successful history in the world marketplace. Over the years, IBM Tivoli service management products have incorporated many new features and grown in complexity. These products also integrate with other complex software systems.

Small, medium, and large organizations implement IBM Tivoli service management products in increasingly complex ways. For many customers, IBM Tivoli service management products are now a global, enterprise-wide implementation that is in use by thousands of users.

This paper provides information to improve the performance of IBM Tivoli service management products when they are deployed in small, medium, or large scale customer environments. The service management products that are included in the scope of this paper include:

- IBM Maximo Asset Management 7.1
- IBM Tivoli Change And Configuration Management Database 7.1 and 7.2
- IBM Tivoli Provisioning Manager 7.1 and 7.2
- IBM Tivoli Service Automation Manager 7.1 and 7.2
- IBM Tivoli Service Request Manager 7.1 and 7.2

In the remainder of this document, these products are referred to as IBM Tivoli service management products. While this document was written specifically for the aforementioned products, this document may also apply to other products based on Tivoli's process automation engine.

The larger and more complex the deployment of IBM Tivoli service management products is, the more challenging it is for you to keep IBM Tivoli service management products performing well for your users. Because some of the greatest challenges are faced by those who deploy IBM Tivoli service management products across large, global enterprises, this document has a special focus on improving performance in advanced enterprise configurations. Note that the recommendations in this document are based on results from non-shared environments. Deployments in virtualized (shared memory/CPU) environments such as VMware® will likely not see the same performance benefits from the recommendations made in this document.

## ***Best Practices Is a Cooperative Effort***

This book is the result of a cooperative effort between IBM employees and a core group of IBM Tivoli service management products enterprise customers. This book was developed both in response to customers and in concert with them. The cooperative effort is ongoing.

System performance information that has been developed both within IBM and in actual deployments of IBM Tivoli service management products are presented here. The goal of this book is to provide you with information that can help you improve the experience of system performance for your users.

## ***Quality Assurance and Testing***

The IBM Quality Assurance team does extensive testing of IBM Tivoli service management products. The team tests during the development cycle, and again, before the product is released. Testing continues after IBM Tivoli service management products are released.

Application and system functionality testing is performed, as is system performance testing. The Quality Assurance team tries to emulate many of the kinds of system setups that you might use. However, a laboratory testing environment can never fully anticipate all of the scenarios that the product is put through in your environments.

For example, some of you have reported that testing with “only” 1,000,000 work orders is insufficient to model the work order load of some global implementations. Efforts are now underway to increase the number and complexity of work orders used in our test scenarios.

The testing environment undergoes constant review to determine how best it can be scaled to meet the demands of large enterprise deployments. Customer involvement and feedback is vital to improving our quality assurance testing. You can provide direct feedback to this document by adding a comment to the wiki page where you downloaded this white paper:



- IBM Maximo Asset Management – <http://www.ibm.com/developerworks/wikis/display/maximo/Performance+and+Tuning>
- IBM Tivoli Change and Configuration Management Database – <http://www.ibm.com/developerworks/wikis/display/tivoliccmdb/Performance+and+Tuning>
- IBM Tivoli Provisioning Manager – <https://www.ibm.com/developerworks/wikis/display/tivoliprovisioningmanager/Performance+and+Scalability>
- IBM Tivoli Service Automation Manager – <http://www.ibm.com/developerworks/wikis/display/tivoli/TSAM+Performance+and+Scalability>
- IBM Tivoli Service Request Manager – <http://www.ibm.com/developerworks/wikis/display/tivolism/Performance+and+Tuning>

## ***Factors in System Performance***

System performance depends on more than the applications and the database. The network architecture affects performance. Application server configuration can hurt or improve performance. The way that you deploy IBM Tivoli service management products across servers affects the way they perform. Many other factors come into play in providing the end-user experience of system performance.

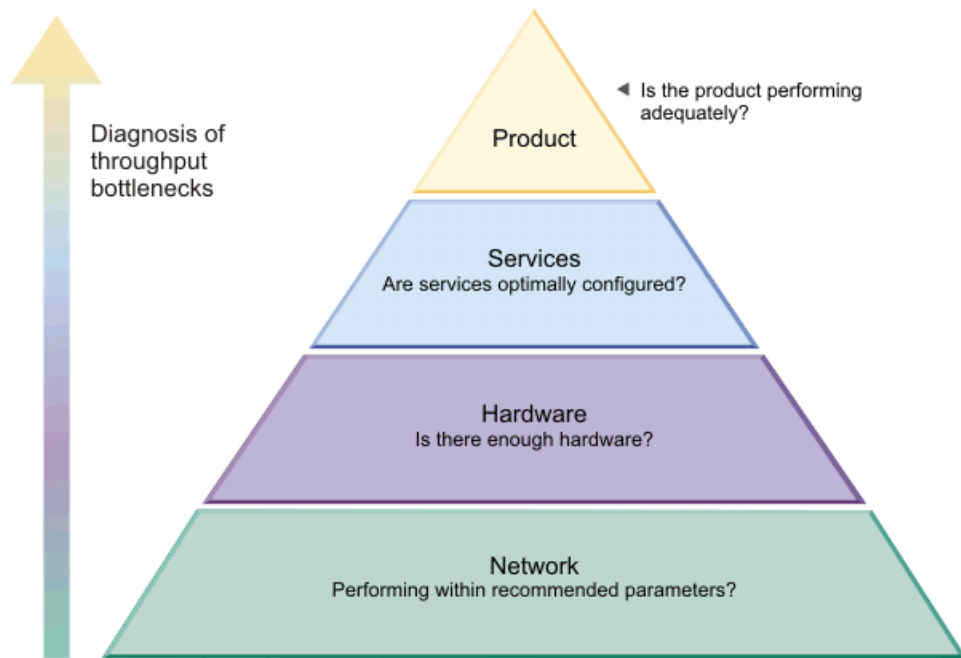
Subsequent sections in this paper address the following topics:

- System architecture setup
- Application server configuration
- Scheduled tasks (cron tasks)
- Reporting
- Integration with other systems using the integration framework
- Network issues
- Bandwidth
- Load balancing
- Database tuning
- SQL tuning
- Client workstation configuration
- Miscellaneous performance improvement tips
- Troubleshooting

# The IBM Tivoli service management products Infrastructure

The figures that follow illustrate the interdependence of IBM Tivoli service management products and other elements that make up the entire system infrastructure.

**Figure 1. IBM Tivoli service management products infrastructure pyramid**

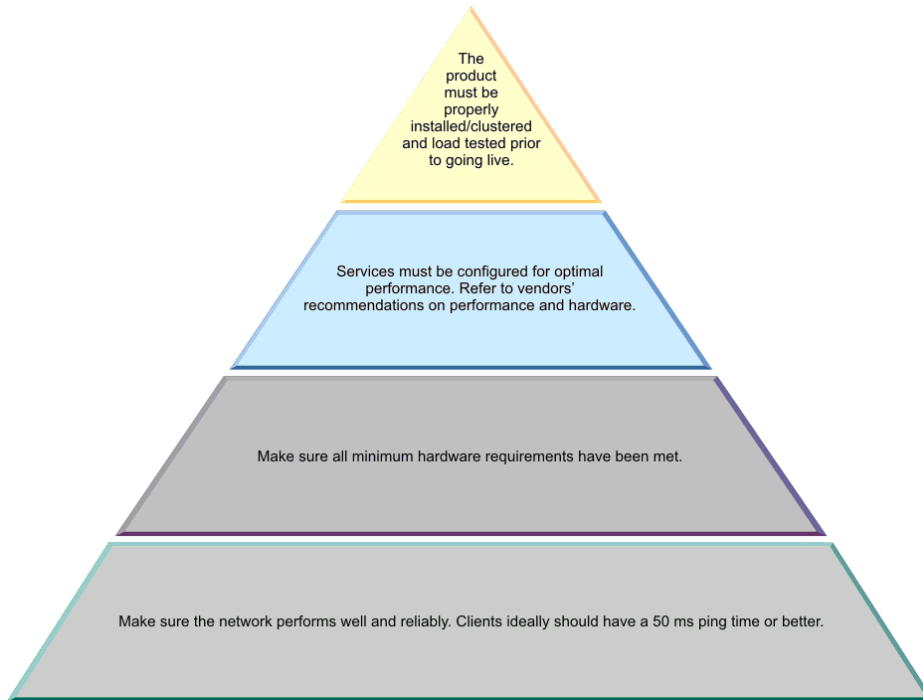


IBM Tivoli service management products can be seen as sitting at the apex of a pyramid that consists of the network, hardware, and software services. This pyramid comprises the infrastructure within which IBM Tivoli service management products operate.

- The entire system infrastructure relies on the network architecture.
- Hardware servers operate within the network.
- Software services such as application servers, report servers, and database servers, operate on the hardware servers.
- IBM Tivoli service management products run within the sphere of the software services.

For IBM Tivoli service management products to perform well, the entire system must be sufficiently robust and well tuned.

**Figure 2. System Infrastructure Pyramid Tuning**



While achieving a 50 millisecond ping time is a challenge in many environments, you can use this target as a guideline. You should also verify that your network bandwidth matches your expectations. For example, if you can ping a system that should be reachable via 1Gb network, but when you send data it transfers at 100Mb speeds instead of one gigabit per second, then you have an issue.

Improving system performance generally requires attention to all elements of the infrastructure. Your databases can be tuned perfectly, but if the network is slow, you may still have system performance issues. Or, you may have a fast local area network, but if client workstations lack memory and processing power, users will experience poor performance.

Customers report that there is normally more than a single issue that they must address. Improving performance involves adjusting many things. Together, these changes can combine to yield significant performance improvement.

But you can “cover up” some performance issues by making improvements in only one or two areas. For example, upgrading end-user workstations can often greatly improve the experience of system performance, even if other problems exist in the system.

## ***Building Performance Improvements into IBM Tivoli service management products***

With each new release of IBM Tivoli service management products, IBM provides new features for your end users. We also strive to increase performance, reliability, and speed with each new release. One way that we work to enhance performance is to provide new tools and features specifically designed to improve system performance.

This section briefly notes some of the performance improvements incorporated in version 7 of IBM Tivoli service management products. Subsequent chapters provide more information about how to take advantage of the enhancements in your own setup.

### **New performance features**

Support for software (HTTP) compression is available in version 7. HTTP compression is a capability that can be built into Web servers, such as IBM HTTP Server, and Web browsers to make better use of available bandwidth, and provide faster transmission speeds. HTTP compression, as well as other

compression techniques, is described in more detail in [Using compression techniques to improve performance](#) on page 30.

Additional performance features were made available in previous versions of IBM Tivoli service management products, such as image and JavaScript™ browser caching (described in Chapter 7) and configurable search options (described in Chapter 6).

## Disable doclink on load

Doclinks provide information about attachments that are linked to individual records in the database. To associate doclinks with records, the database must evaluate conditions to make sure that there are attachments available for a particular record. These evaluations consume resources and degrade performance of the data loading process. To improve performance, you can disable doclink processing so the evaluations are not performed when records are loaded into the database. A new property is added to the webclient.properties file called `enabledoclinkonload`.

With `enabledoclinkonload=false` the paperclip for every record appears as highlighted regardless of if an attached document record is associated to the record or not. Use this setting to enable the performance improvement.

With `enabledoclinkonload=true` the query is run to see if the record has an attached document associated to it or not and only those records with attached documents associated to them appear with a highlighted paperclip icon. Use this setting to maintain the previous doclink functionality.

After setting the property, you must save and rebuild and redeploy the `maximo.ear` file for the changes to take effect.

## Integration and Batch Processing

- **Multi-threading**  
IBM Tivoli service management products employ a multi-threaded integration model. This integration framework model is set up to process multiple inbound transactions at once.
- **Cluster Support**  
Servers and processes can be configured to run in a clustered environment. Distributing processing functions across multiple server instances [spanning one or many clusters], can increase performance. Using clusters also provides fail-over protection. If a server instance [a cluster member], that is running processes or an application server fails, other server instances can continue to service requests and deliver responses.

While setting up a clustered environment with IBM Tivoli service management products, configure the clusters so that the server that responds to customer requests, made via the user interface, runs in its own cluster. Configure other processes, such as JMS queue processing and other cron task operations, like `VMMSYNC/LDAPSYNC`, so these processes occur in a different cluster. An outage of the non-UI cluster does not immediately affect customers who are using the interface.

For more information, see [Chapter 2: System Architecture and Application Server Configuration](#), [Chapter 3: Scheduled Tasks \(cron Tasks\)](#), and [Chapter 4: Integration](#).

# Chapter 2: System Architecture and Application Server Configuration

This chapter addresses ways to help maximize system performance by configuring your system architecture to account for your resource needs.

## Terminology

This section provides a brief explanation of some terms used in this book.

An **application server** is software that maintains and provides an infrastructure to run applications (such as IBM Tivoli service management products). IBM WebSphere® Application Server and Oracle WebLogic are the two standard application servers that IBM Tivoli service management products use.

**IBM Tivoli service management products** maintain business objects and configuration files. Application servers are created in WebSphere Application Server Network Deployment or any of the supported WebLogic application servers.

A **cluster** is a collection of application servers that perform the same task. For example, you can set up a user-interface cluster dedicated to serving end users who are using browsers. Each application server in the cluster has the same setup and configuration parameters.

A **JVM (Java™ Virtual Machine)** is a platform-independent execution environment that converts Java code to machine language and executes it. Each application server runs on a different JVM. JVMs in a cluster can run on the same physical piece of hardware or on different hardware.

A **processor** is the central processing unit of the computer. A typical server has four processors, but can have more processors. Set up one JVM for each processor.

**Scaling** is the process of adding hardware, adding memory, or adding processors to an existing system. With the ability to deploy business rules or components on multiple servers, IBM Tivoli service management products provide virtually unlimited scalability.

**Vertical scaling** is the process of adding memory or processors to a computer to give it more processing power. **Horizontal scaling** is the process of adding hardware to the system.

**Front-end transactions** and **user load** are transaction requests initiated from a browser. **Back-end transactions** and **nonuser load** are processing requests that occur on a scheduled basis, or that come in by way of the integration components.

## Basic System Configuration

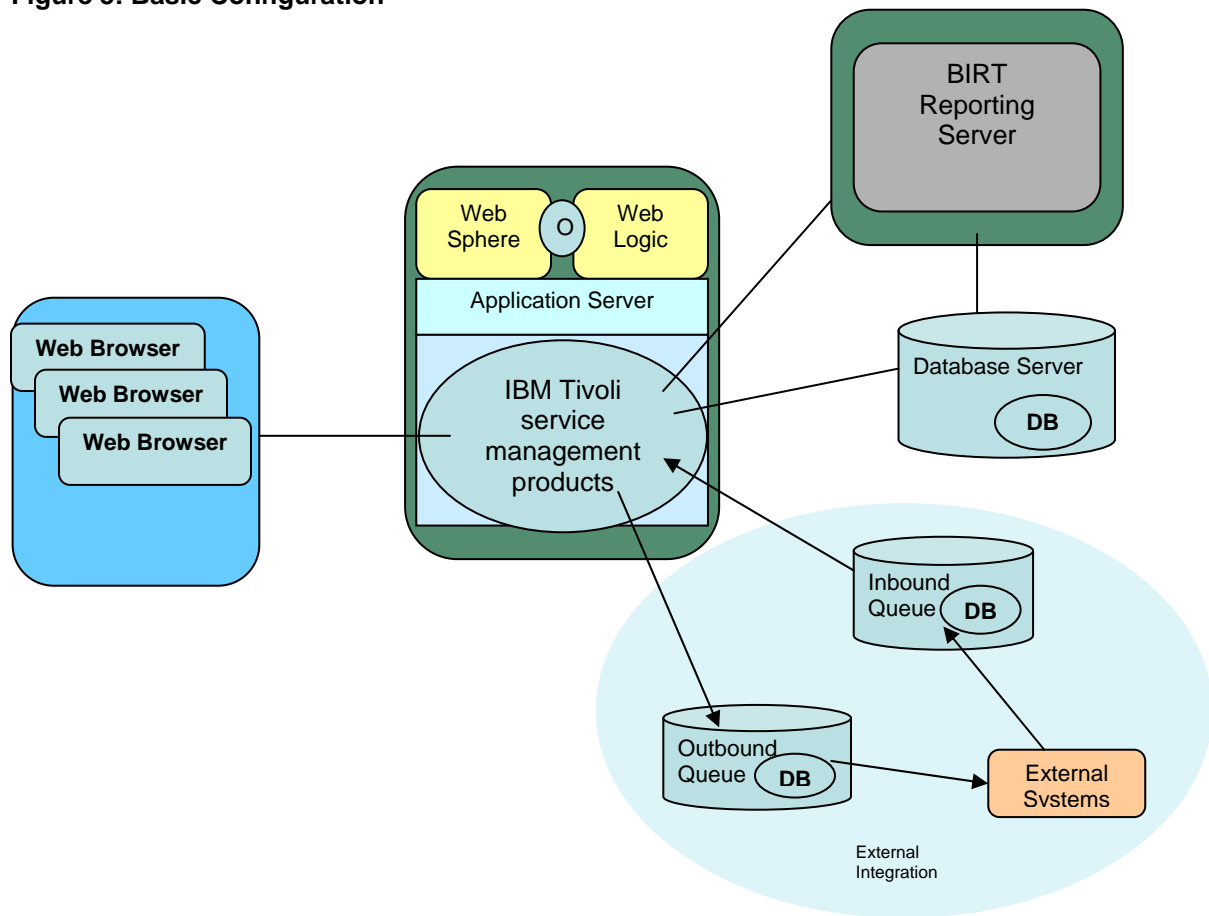
A basic configuration consists of IBM Tivoli service management products running on a single application server. The application server connects to a single instance of the database that is available on a database server. The application server can also connect to a report server.

If the integration framework is configured for deployment, then you must set up additional messaging queues. IBM Tivoli service management products use the additional queues to send data to external systems, and to receive data from external systems.

Your testing and QA configuration environment (to test the development work) should closely simulate your eventual production environment. The basic configuration is appropriate for development configuration and for QA configuration if your production system is expected to have a user load of 50 to 75 users, based on the version of you are running. This configuration is also suitable for a production system with an actual user load of 50 to 75 users. Refer to the [IBM WebSphere Application Server Version 6.1 Tuning](#) section for additional details.

Figure 3 illustrates the high-level components of the basic configuration.

**Figure 3: Basic Configuration**



Even with fewer than the recommended number of users per JVM, the basic configuration can become overloaded if a great deal of processing is done. For example, running scheduled jobs (cron tasks) and running reports both require considerable memory and processing power.

When processing power and memory are overtaxed, the end user who is using the application from a browser can experience unacceptable performance. If the basic configuration is performing poorly, you might need to deploy the advanced enterprise configuration.

In the basic configuration, even if the application server process runs on a 64-bit computer, performance can be slow. Although a 64-bit computer allows greater memory utilization, having a larger heap also increases the overhead of garbage collection. So you must consider both memory and the JVM heap size, when optimizing performance.

## ***Advanced Enterprise Configuration***

If you have a small to medium-sized implementation, follow the standard installation and setup instructions and the application server (WebSphere Application Server or WebLogic) configuration instructions. If your production system must support hundreds of users, the basic configuration is likely to be insufficient, due to memory and processing limitations. A single instance of the application server to support the application cannot handle the load that is required in enterprise deployments.

Because IBM Tivoli service management products are interactive applications, the end user who uses the application from a browser expects response from the server to be immediate, or nearly so.

Other processes in IBM Tivoli service management products, such as cron tasks and inbound messages from external systems, do not require user interaction. Response time for these processes does not need to be immediate. These processes can be configured to run in separate clusters or on separate hardware.

## Deploying IBM Tivoli service management products in Multiple Clusters

Use load distribution in a heavily used system that experiences unacceptable delays in applications and overall throughput. Load distribution separates the user-interface traffic from all other processing in IBM Tivoli service management products.

Deploying IBM Tivoli service management products on more than one server (or *clustering*) is an effective way to distribute user load and improve the end-user experience of system performance. When you use clustering, implement the cluster configuration so that the system can scale well.

You can set up multiple clusters, each consisting of multiple Java virtual machines (JVMs). The number of JVMs depends in part on the overall hardware and software limitations of the environment. You can tune the setup based on traffic and performance numbers.

A system of separate clusters, with multiple JVMs as cluster members, provides advantages in system administration. In such a setup, end users are not affected by any problems in the queue cluster or the cron task cluster that requires a cluster restart.

For details of a two-cluster setup, see Appendix A: Setting up Two Clusters on page 40.

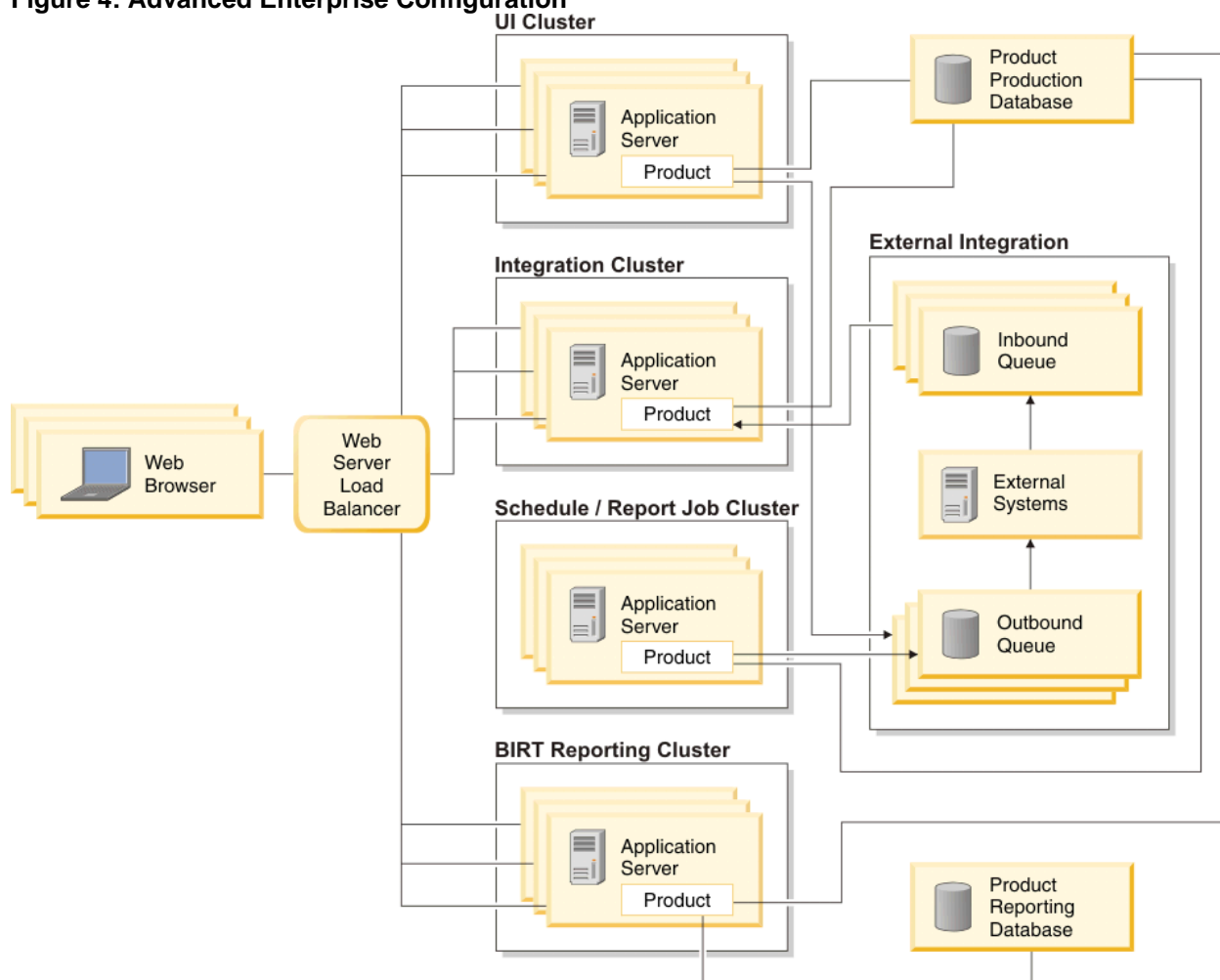
### Isolating User Applications

For your system to perform well, isolate the user-interactive applications to one cluster, and the asynchronous processes to a separate cluster. Such a configuration also helps the system to scale well.

You can further enhance each cluster simply by adding servers, depending on your needs.

Figure 4 illustrates the high-level components of an advanced configuration.

**Figure 4: Advanced Enterprise Configuration**



## Isolating Reporting

The use of application reports can also put a large burden on the user-interactive applications. To remove the impact of reports on the user-interface performance, run reports that require significant system resources on a separate report server, or schedule reports to execute from the cron cluster.

To help the overall system performance, run only simple reports from the user-interactive application.

For more information on managing the reporting function, see **Chapter 5: Reporting** on page 18.

## IBM WebSphere Application Server Version 6.1 Tuning

Tables 1 and 2 show the suggested settings to optimize WebSphere Application Server Version 6 for IBM Tivoli service management products. The settings provided are based on 50 users per JVM load for those products running on Tivoli's process automation engine version 7.1.1.6, and earlier versions, and on 75 users per JVM for Tivoli's process automation engine version 7.1.1.7.

While these settings provided optimal performance in the test environment, your environment may require different settings.

IBM Tivoli service management products recommend that the settings below are used as a guideline or starting point, and then are monitored and tuned to your specific environment.

**Table 1. WebSphere Application Server V6 JVM Optimization**

Web Sphere 6.x Application Server JVM Tuning recommendations			
Servers --> Application servers --> {ServerName} --> Process Definition --> Java Virtual Machine			
Settings	Default	32 Bit JVM	64 bit JVM
Java Virtual Machine --> Initial Heap Size	No Values	1280m to 1536m*	4096m
Java Virtual Machine --> Maximum Heap Size	No Values	1280m to 1536m*	4096m
Java Virtual Machine -->Generic JVM Arguments			
Settings	Default	32 Bit JVM	64 bit JVM
-Dsun.rmi.dgc.ackTimeout (IBM or Sun SDK)	Not included	-Dsun.rmi.dgc.ackTimeout=10000	
-Djava.net.preferIPv4Stack (IBM or Sun SDK)	Not included	-Djava.net.preferIPv4Stack=true	
Disable explicit garbage collection (IBM or Sun SDK)	None	-Xdisableexplicitgc (IBM SDK)  -XX:+DisableExplicitGC (Sun SDK)	
-Xgcpolicy (IBM SDK only)	-Xgcpolicy:optthruput	For general UI deployments: -Xgcpolicy:gencon or  -Xgcpolicy:subpools (when number of processors > 15)  For Integration Framework only deployments: -Xgcpolicy:optthruput	
-Xmn (Applicable only with gencon with IBM SDK)	None	-Xmn320m	-Xmn1024m

### \* Notes on 32-bit JVM settings:

For 32-bit environments, depending on type of operating system and the available physical memory, the upper limit for the maximum JVM heap size can vary. Physical memory needed by IBM Tivoli service management products is composed of the java heap and memory required for the process outside of the heap. The total memory size cannot exceed what the OS can provide to the java process. For more details on java heap size allocations and limits, see the link:

[http://publib.boulder.ibm.com/infocenter/javasdk/tools/index.jsp?topic=%2Fcom.ibm.java.doc.igaa%2F\\_1vg000121410cbe-1195c23a635-7ffa\\_1001.html](http://publib.boulder.ibm.com/infocenter/javasdk/tools/index.jsp?topic=%2Fcom.ibm.java.doc.igaa%2F_1vg000121410cbe-1195c23a635-7ffa_1001.html).

On the Windows Server operating system, 1280 megabytes is a safe maximum heap size for IBM Tivoli service management products. Given enough physical memory, you can specify a maximum heap size as high as 1536 megabytes, but this value is not always safe. There is a possibility of running out of physical memory depending on the workload. You should always monitor your heap usage and adjust the



minimum and maximum java heap sizes accordingly to accommodate your workload as well as the available physical memory. Control the load on each JVM by having a sufficient number of members in your application server cluster. Since the IBM JVM does not support the PAE switch, attempts to increase java heap size with PAE will not help. **For maximum scalability, IBM Tivoli service management products recommend to move to 64-bit platforms.**

**\*\* Notes on `sun.rmi.dgc.ackTimeout` property:**

The value of this property represents the length of time (in milliseconds) that the server-side Java RMI runtime will strongly refer to a remote object (or a reference to a remote object). This time is returned from the current virtual machine as part of the result of a remote method call. The default value is 300000 (five minutes). Setting the value too low can increase the risk of a remote object being prematurely garbage collected, when the only known reference to the remote object is the one in transit as part of the remote method call result. Setting the value too high will prevent the remote objects being garbage collected in time when they are not actually being referenced, which can cause a larger memory footprint on the server.

Since IBM Tivoli service management products uses RMI and it allocates a large quantity of short-lived remote objects, especially for products running on Tivoli's process automation engine prior to version 7.1.1.7, setting `sun.rmi.dgc.ackTimeout` to the default five minutes prevents garbage collection from collecting the objects on the server fast enough, which can cause out of memory problem. IBM Tivoli service management products recommend this value to be set to at most 10000 (10 seconds); however, a lower value may further reduce the memory footprint. Performance benchmark testing uses 1000 (1 second) and observes slightly lower CPU usage compared to 10000. However, in some cases due to network latency, a client may not be able to receive the remote object reference and acknowledge back to the server within 1 second, and the server may prematurely garbage collect the object. If you have an environment where latency can be an issue, use a value that is large enough to accommodate the roundtrip network traffic, e.g. 10000.

**Table 2. WebSphere Application Server V6 Thread Pool Optimization**

WebSphere Thread Pool Settings		
Servers --> Application servers --> {ServerName} --> Thread Pools		
Settings	Default	Recommended
Default --> Minimum Size	5	20
Default --> Maximum Size	20	50
Default --> Thread inactivity timeout	5000	30000
WebContainer --> Minimum Size	10	50
WebContainer --> Maximum Size	50	50
WebContainer --> Thread inactivity timeout	3500	30000

## IBM HTTP Server Version 6.x Tuning Suggestions

Table 3 shows suggested settings to optimize IBM HTTP Server Version 6 for IBM Tivoli service management products. The settings provided are based on 50 users per JVM load for those products running on Tivoli's process automation engine version 7.1.1.6, and earlier versions, and on 75 users per JVM for Tivoli's process automation engine version 7.1.1.7.

While these settings provided optimal performance in the test environment, your environment may require different settings. IBM Tivoli service management products recommends that the below settings be used as a guideline or starting point and then monitored and tuned to your specific environment.

**Table 3. IBM HTTP Server Version 6 Optimization**

Web Sphere IHS HTTP Server Configuration ( For clustered environment)		
Settings	Default	Recommended
TimeOut	300	900
MaxKeepAliveRequests	100	0
KeepAliveTimeout	10	10 (60 for low bandwidth situations)
ThreadLimit	25	2400

ServerLimit	64	Comment out
StartServers	2	Comment out
MaxClients	600	Comment out
MinSpareThreads	25	Comment out
MaxSpareThreads	75	Comment out
ThreadsPerChild	25	2400
MaxRequestsPerChild	0	0

Note: The above settings for IBM HTTP Server are optimized and tested on a Windows environment. While these settings were also tested on an AIX environment, they are not optimized for Unix/Linux. Additional testing is planned to determine optimal settings for these environments, which will be included in a future update of this document.

## Managing Integration Load

If you are using the integration framework to work with other systems, from IBM or other vendors, the activity generated by the integrated systems place an additional load on the system resources. This additional load is over and above the load imposed by users accessing IBM Tivoli service management products using the interface. To ensure that human users receive optimal performance, you can take steps to limit the resources consumed by the integrated systems.

To do this, you can limit the number of message-driven beans (MDBs) on the continuous queue of the integration framework. You can also improve performance by processing inbound integration traffic on a different server or cluster.

A default installation of IBM Tivoli service management products does not have a specific number of MDBs in the deployment descriptor file. The Enterprise Java Bean container manages the number of MDBs that are created. Depending on the load on your system, you might need to reset this number. It is generally best to start with a small number, such as two MDBs.

For a small or medium-sized implementation, two or three MDBs usually provide acceptable performance. As the system load increases, you might need to gradually increase the number of MDBs. You generally should not exceed ten MDBs. More than ten MDBs does not necessarily yield higher throughput and it can significantly increase resource usage (CPU cycles in particular).

Test different numbers of MDBs in a development setting to determine an appropriate number before you establish the number of MDBs for your production system.

## Load Balancing

Load balancing is the distribution of the task load across multiple instances of an application. User load comes from users who are logged-in to the system and using the interface to perform tasks. Non-user load comes from things such as scheduled jobs (cron tasks) and integration framework incoming transactions.

Distribute user load and non-user load across different application servers or clusters using either a hardware or a software load balancer. A software load balancer is included with IBM Tivoli service management products.

Use any commercially available hardware device for hardware load balancing.



## Chapter 3: Scheduled Tasks (cron Tasks)

A cron task is a task that is scheduled to run at a specified frequency, such as every 10 minutes or on a certain day or date at a certain time. The Reorder routine and Preventative Maintenance record generations are two examples of activities that use cron tasks. Cron tasks run in the background and require no direct end-user action after they have been configured.

If there are large numbers of cron tasks running at the same time that there is significant end-user load, response time for the end user can decline. To improve response time for the end user, you can set up cron tasks to run outside of the end-user environment (in a different cluster to the cluster that the user interface server runs in).

You can also run cron tasks on several nodes within a cluster or even on several nodes in different clusters. If any one of the nodes fails, the surviving nodes can still run the cron job.

### Configuring Cron Tasks in Version 7

To set up cron tasks to run outside the end-user environment in Version 7, complete the following steps:

1. Use the file `maximo.properties` to contain the correct property `mxm.crontask.donotrun` in the ear file for the UI server cluster.
2. Use the file `maximo.properties` and property `mxm.crontask.donotrun` in the EAR file for the cron task server.
3. If you want the different cron tasks to run on different JVM in the cron task cluster:
  - a. Give each JVM a different name in the cron task cluster by adding `-Dmxm.name=<server name>` to the JVM generic argument list in each JVM.
  - b. Use system properties application to set different `mxm.crontask.donotrun` properties for each server in the cron task cluster.

If the non-user clustered environment consists of more than one server, each cron task runs on only one server. If any server shuts down, the cron task starts on another running server.

For the same cron task definition, you can create multiple cron task instances to increase performance, but make sure that the instances do not interfere with each other. For example, you can set up reorder cron task instances so each instance is used to handle a different site.

When you define multiple instances on the same server, set them up so they do not all start and run at the same time. A simultaneous start or run of multiple cron tasks can adversely affect the server's performance.

# Chapter 4: Integration

This chapter provides information on improving system performance in a setting where the integration framework is used to integrate IBM Tivoli service management products with other systems.

## Integration and Batch Processing, version 7

IBM Tivoli service management products version 6 introduced multi-threaded integration. Using multiple threads allows the integration framework to process multiple inbound transactions at the same time.

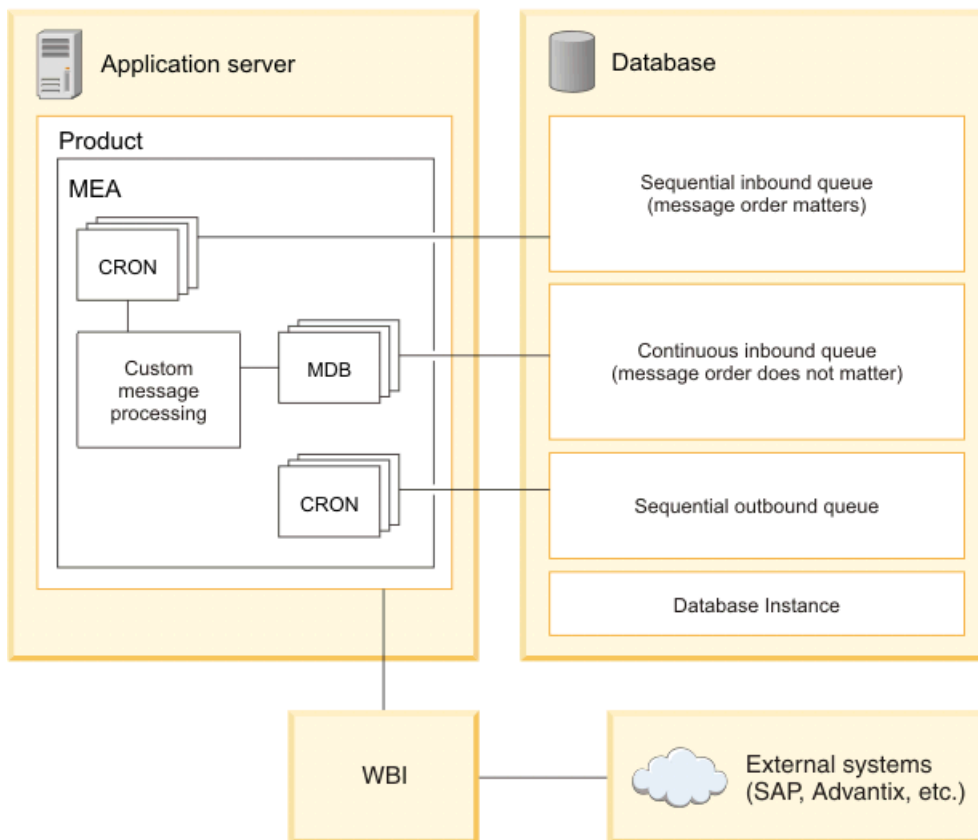
IBM Tivoli service management products version 7 also support clustering. Transactions can be clustered across multiple physical computers.

Cron tasks can also take advantage of clustering. You can run cron tasks on multiple separate computers. Running cron tasks on separate computers improves throughput and can help improve fault tolerance.

Incorporation of both multi-threading and clustering greatly increases throughput in an integrated environment.

**Figure 5: Overview of integration**

### Logical Overview - Integration



MDB - Message driven bean  
CRON - Cron / scheduled task

## ***Performance Issues in Integrated Systems***

In an integrated IBM Tivoli service management products environment, one of the main causes of falling performance numbers is the demand of non-user-load processes via the integration framework.

The continuous queue functionality that the integration framework uses is resource-intensive if there is a large amount of queued data. Queuing can require significant memory and many CPU cycles.

In a large, integrated system, you need to tune the application parameters and the overall environment parameters, including the parameters for the application server.

## ***Clustering with the Integration Framework***

When you integrate IBM Tivoli service management products with external systems, it is a best practice to set up multiple clusters to provide failover and load sharing. In a typical implementation, IBM Tivoli service management products are connected with multiple external systems through JMS queues. You configure the queues using the integration framework.

The purpose of clustering is to separate the inbound queue processing to a different cluster from the interactive users. Separating the queue this way prevents inbound traffic from affecting the user interface performance.

Set up separate sequential inbound and sequential outbound queues for every system that IBM Tivoli service management products communicate with. Using multiple queues prevents a problem in a system from affecting traffic in other systems. If you also set up each sequential queue with its own cron task, administering the queues is easier.

In addition to setting up multiple clusters, you should determine and set appropriate cron task frequencies. Cron tasks can consume considerable resources; running a job too often can put a drain on system resources and affect performance.

## **Recommended Approach to Clustering**

A best practice for clustering is to use at least three clusters. In the text that follows the clusters are named UI, Q, and Cron.

The UI (user interface) cluster is where users generate transactions when they save work in the database. This cluster should also host the sequential outbound queues that send user-generated transactions out to external systems.

The Q (queue) cluster is intended for processing inbound transactions. However, you might need sequential outbound queues in this cluster if outbound data is generated as a result of inbound data. For example, a data loop-back condition may be initiated when an incoming receipt generates an inventory transaction that needs to be sent out from IBM Tivoli service management products. This loop-back behavior is not standard in IBM Tivoli service management products. You can customize the system to work this way.

The optional Cron cluster is used to run other cron tasks:

- If those cron tasks generate outbound transactions, then the Cron cluster also needs its own set of sequential outbound queues. The integration framework cron tasks [File, Interface table, JMS sequential inbound queues] should be run on the Q cluster.
- The integration framework cron tasks for outbound sequential queues need to run on all the clusters that host the outbound sequential queues. So if the outbound queues are created on the Q and Cron clusters, the corresponding outbound JMS cron tasks need to be run on those clusters too [along with the UI cluster].

For details on setting up such a clustered environment, see Appendix A: Setting up Two Clusters on page 40.

## ***Using Exception-Handling Queues***

Use exception-handling queues to prevent system performance degradation caused by bad data.

Bad data in a queue can come in two forms.

- The data needs manual intervention to correct.
- The “bad” data becomes “good” data in a matter of time or sequence.

An example of bad data that can become good data is a PO that specifies a vendor that is not yet saved in the database. After the vendor record is saved, the PO can be processed.

An effective way to deal with a high volume of this type of bad data is to establish an exception-handling queue. You can direct the bad messages from the continuous queue to the exception handling queue after a preset number of retries in the continuous queue. Sidetracking bad messages prevents the continuous queue from consuming system resources to process bad data. The bad data can be processed in the exception handling queue in a way that suits your needs.

The integration framework provides an out-of-the-box message driven bean (MDB) that can be used for processing messages from the error queues. When error queues are implemented, the MDB section for the error queue should be uncommented in the file `ejb-jar.xml` as well as the file `ibm-ejb-jar-bnd.xmi`, for the module `mboejb`.

## Using a Loop-Back Exception Queue Design

One approach to addressing a high volume of continuous queue errors is to take advantage of the exception queue concept of WebSphere Application Server. One such design option is described below:

- Create an exception destination named “A” for the continuous queue.
- For exception destination A, set A as its own exception destination,
- Then specify that A is the exception queue for the continuous queue.

This queue configuration causes error messages to loop-back to the end of the error queue (A). A bad message is reinserted into a queue behind the message that allows the bad message to become a good message. This setup allows most errors that are based on bad timing to be resolved without manual error correction.

For products running on the process automation engine version 7.1.1.5, and later versions, set the property `mxe.int.mdbdelay`. For earlier versions, set the environment variable `MDBDELAY` in the file `ejb-jar.xml` in the module `mboejb` to a value that suits your processing requirements.

Setting `mxe.int.mdbdelay` or `MDBDELAY` to a value greater than N [where  $N > 0$ ] implies that each error message is being delayed by N milliseconds before being processed. For those N milliseconds, the MDB is holding the messages and is not doing any other work. Setting the delay to a high value can decrease error-message-processing performance. Delaying the error message processing using the property `mdbdelay` or environment variable `MDBDELAY` helps good messages to be processed faster. This also prevents the system from being tied up in processing error messages that continue to arrive because the system is still waiting for good data from either a self-correcting sequence or manual correction.

If you set up an exception queue to handle errors, the error XML files are saved in the continuous queue folder, not in the exception queue folder hierarchy.

## XML/HTTP and SOAP-based Web services for Inbound processing

An important thing to consider while integrating your external system with IBM Tivoli service management products is whether you can use XML/HTTP mechanisms instead of the Web services-based mechanism (SOAP). XML/HTTP has less overhead and performs better under load conditions compared to SOAP. So if there is an option for your integration to use XML/HTTP or SOAP you know which service scales better.

## Multi-record inbound XML message processing

Consider using a file import when importing XML documents containing multiple records. When an HTTP POST or a Web service mechanism is used, the whole XML message is placed in the queue (if using the asynchronous path). The message is eventually processed by only one MDB (assuming that the continuous queue is used), which implies single-threaded processing of this large message. If, however, the file is loaded through the file import mechanism, the multi record message would be split into smaller XML messages, with each containing one record. Then smaller messages are placed into the queue. This

is more efficient because it allows multiple MDBs to pick up individual records and process them in parallel.

## **Using the Sync operation with Action Add**

Consider providing the action attribute value "Add" (for the Sync operation) while sending inbound data into IBM Tivoli service management products. If the Add action is not used, the framework tries to find the record in the database and assume it is an Update or Add, based on whether the record is found, or not. If, the sender provides the action as "Add", the framework tries to create the objects directly, instead of searching for them, and that improves performance. Keep in mind that providing the action attribute value as "Add" can only be done if the sender is confident this data does not exist in the database.

## **Using Message tracking**

Message tracking is performed to track a queue-based Integration Framework message. As with any reliable tracking mechanism, this comes with a performance overhead because all tracking points are logged in a database. Be aware of this performance impact before you implement message tracking.



## Chapter 5: Reporting

IBM Tivoli service management products 7 provide an ad hoc reporting feature. This feature is available in many applications by clicking **Select Action ->Run Reports**. Examine the ad hoc reports and options. Ad hoc reports allow you to create unique reports by selecting columns and choosing sorting and grouping options. You can save report queries so you can use them again, without having to re-specify the query criteria. You can run reports on demand, or the reports can be scheduled. You can have the results emailed to any user and you can secure reports so only authorized users can run them. When you create a report, you choose the format of the output, by selecting from a list of popular file formats (for example, PDF and XLS).

Some customers customize IBM Tivoli service management products to have applications display more information, in the user interface, than a standard installation displays. For example, by default, the List tab in the Assets application provides the following information for each asset:

- Asset (name)
- Description
- Location
- Loop Location
- Parent
- Rotating Item
- Is M&TE
- Linear
- Site

The customizations are performed by using Application Designer, or by directly modifying the application's XML. The usual intent of the customizations is to provide a convenient set of information that can easily be downloaded as an XLS file, by using the "Download" feature.



However, these customizations can create an unnecessary drain on system performance because every time the List tab is displayed, the added information must be retrieved from the database. Instead of customizing applications to display additional information, use ad hoc reports to create the information you need. Using the ad hoc reports reduces the impact on system performance because the additional information is retrieved, via an ad hoc report, only when it is needed; not each time the application is opened.

Best practices for BIRT report performance and design are available from IBM Support as downloadable documents. Use the following URL to locate and download these documents:

<http://www.ibm.com/support/search.wss?cc=us&cs=utf-8&rs=0&ics=iso-8859-1&loc=en-US&from=tss&lang=en+en&q=1305031&ibm-search.x=12&ibm-search.y=3&dc=&dtm>

## Chapter 6: Tuning the Database and Tuning SQL

This chapter provides information on improving system performance by focusing on two related areas:

- Database tuning
- User queries and SQL tuning

### *The Database*

The database is central to the functionality of IBM Tivoli service management products. This database stores all data that is collected and calculated by the applications. This database also stores metadata for configuring and maintaining the environment.

The database server processes all transactions from the applications. The integrated reporting function accesses the data in the database to generate documents such as work orders and purchase orders. Reporting also generates resource-intensive management reports. See *Isolating Reporting* on page 9 for information that is used to reduce the impact of reporting activities on the user interface.

Because all functionality is based on database performance, the database should be a key focus for performance tuning.

### *Database Connection Pool*

IBM Tivoli service management products use proprietary connection pooling. If necessary, you can tune the connection pool settings for better management of connections. Table 4 describes the database pool properties and their default values.

If you only have a small number of cluster members, or if your database resources are abundant, you can increase the number of free connections to provide faster access to the database for additional clients. To reduce the overhead associated with allocating and freeing database connections, you can increase the value in the properties `mxe.db.initialConnections` and `mxe.db.maxFreeConnections`. These properties are set in the System Properties application.

**Table 4: Database Pool Properties**

Setting	Default Value	Description
Initial connections ( <code>mxe.db.initialConnections</code> )	6	When the server starts up, it immediately creates this many database connections and puts them in the connection pool, ready to be used. You can opt for a larger number only if large number of users log in immediately after server starts up. Otherwise, this number should not exceed the maximum free connection, because extra connections are freed if not used.
Maximum free connections ( <code>mxe.db.maxFreeConnections</code> )	8	As transactions are finished for the user, connections are freed back to the pool. If the connections put back are greater in number than the connections being requested, the number of free connections increases. Once server detects that the number of free connections exceeds this setting, the server closes the extra connections to keep the number of free connections within the range.

Setting	Default Value	Description
Minimum free connections (mxe.db.minFreeConnections)	4	When a connection is requested by a user operation, a free connection in the pool is assigned to this user. If more free connections are taken from the pool than the connections freed back to the pool, the free connection number drops. Once the server detects that the free connection number is below this setting, it creates new connection(s) to fill the pool so that they are ready for the new requests.
New connections (mxe.db.newConnectionCount)	3	When the connection number drops below the threshold of minimum free connections, new connections are created in the batch of this setting to fill the pool.

## Database Tuning

You can apply standard database-tuning techniques to IBM Tivoli service management products. Periodically monitor a production database during peak load. You can use any appropriate monitoring tools or techniques. If necessary, adjust parameters to resolve the bottlenecks reported by the monitoring tools.

### *Indexing*

Indexing a database requires a good understanding of the data, user functions, and of how databases use indexes. Indexes use key parts of data from a table in a binary structure to enhance searching capability. Each record of data in the table must have associated data in the index.

Indexing can greatly increase search speeds. However, a drawback of indexes is that for each insert, update, or delete, the index must also be updated. Database administrators often apply many indexes to a table to enhance searching, and then find that other activities have slowed. You should review indexes to ensure that you have the right balance for searching and updating tables.

Indexes should be separated from the data and placed into a different tablespace. Use the Database Configuration Application to move the indexes. Also, you should separate large tables, like Assets and Work orders, into their own tablespace.

### *Special Index Types*

Some special index types are available on each database platform that are not available in the Database Configuration application. These index types can be created and maintained from the back end, and they can improve performance in specific cases. For example, on Oracle you might create a bitmap index or a function-based index if you determine that these indexes would improve certain queries.

If you use special index types, the system administrator must remember to remove any special indexes before configuring database changes. After the database is configured, the special indexes must be replaced.

### *Statistics*

Statistics should be periodically generated and updated on all the tables and indexes, including text search indexes.

In the Database Configure application, you can use a Select Action menu action to do this. You can also update index statistics from the back end.

The Update Statistics action is enhanced to function on Oracle by running the procedure MAXIMO\_GATHER\_TABLE\_STATS.

### *Sequences*

Sequence caching in Oracle and IBM DB2® platforms is highly recommended for improved performance. The suggested cache size for all sequences is 20 with the exception of the maxseq sequence, which is

used for rowstamps. The recommended size for this sequence is 200. To enable or modify sequence caching, perform the following steps:

- Run the following command to generate a script file containing the SQL statements to set the cache size:  
db2 "select 'alter sequence <db\_name>.' || sequencename || ' cache 20 ;' from <db\_name>.maxsequence" > change\_seq\_cache.sql
- Edit the `change_seq_cache.sql` file to modify the value for `maxseq` as described above. In addition, if you have the IBM Tivoli Asset Manager for IT product installed, you must remove any entries that match the following sequence names as these sequences are handled internally by that product and should not be altered manually:

```
ASSETATTRIBUTESEQ
CDMCITYPESSEQ
CLASSANCESTORSEQ
CLASSIFICATIONSEQ
CLASSSPECSEQ
CLASSSPECUSEWITHSEQ
CLASSSTRUCTURESEQ
OMPSEQ
RELATIONRULESEQ
RELATIONSEQ
ACTCIRELATIONSEQ
ACTCISEQ
ACTCISPECSEQ
DEPLOYEDASSETSEQ
DPACOMMDEVICESEQ
DPACOMPUTERSEQ
DPACPUSEQ
DPADISKSEQ
DPADISPLAYSEQ
DPAFILESEQ
DPAIMAGEDEVICESEQ
DPAIPXSEQ
DPALOGICALDRIVESEQ
DPAMADAPTERSEQ
DPAMADPTVARIANTSEQ
DPAMEDIAADAPTERSEQ
DPAMMANUFACTURERSEQ
DPAMMANUVARIANTSEQ
DPAMOSSEQ
DPAMOSVARIANTSEQ
DPAMPROCESSORSEQ
DPAMPROCVARIANTSEQ
DPAMSOFTWARESEQ
DPAMSWSUITECOMPSEQ
DPAMSWSUITESEQ
DPAMSWUSAGERANGESEQ
DPAMSWUSAGESEQ
DPAMSWVARIANTSEQ
DPANETADAPTERSEQ
DPANETDEVCARDSEQ
DPANETDEVICESEQ
DPANETPRINTERSEQ
DPAOSSEQ
DPASOFTWARESEQ
DPASWSUITESEQ
DPATCPIPSEQ
DPAUSERINFOSEQ
OMPCIRLNSEQ
```

- Execute the SQL script to enable or change the sequence cache. Note that the example above is for DB2. The steps for Oracle are identical with the exception that you use the `sqlplus` command instead of the `db2` command.

### ***Reorganize Tables and Indexes***

Use the `reorgchk` and `reorg` tools, which are provided with IBM DB2 and Microsoft® SQL server, to optimize tablespaces and indexes.

You may want to perform reorganizations to reclaim space in a table. When a large number of updates are made on a table, the space can become fragmented. If no further growth is expected in the table then

the space can be reclaimed. However, if further inserts are expected, then DB2 should be effectively reusing the space within the table. Reorganization should not be done soon after a previous reorganization as additional inserts will be done to the table. The action of shrinking and growing the table is detrimental to performance.

Another case where reorganizations can be helpful is to improve performance. When the data is not aligned on an index, performance can become degraded.

Some questions to consider before doing reorganization are:

- Is fragmented space causing a performance problem?
  - Will the records in the table be updated or new records inserted within a short period of time?
  - Is fragmented space resulting in inefficient storage utilization?
- Is the order of the table data aligned with a particular index?
- Has performance degraded over time or has it happened at a specific time?
- What changes could have caused the incident?
- How does the performance correlate to system CPU and I/O utilization?
- Have access plans changed?
- What does the reorganization check say?
- If table reorganization is done, what is the expected benefit? Running reorganization affects data availability, consumes system resources and takes time to complete.

General recommendations for reorganizations are:

- Only run a reorganization against a table when careful analysis has been done to determine that a reorg corrects the performance problem
- Only reorg tables to reclaim space which have had many deletes and updates when it has been determined that no further inserts will be done in the near term
  - Meta data tables should not be reorganized
- A reorg check should be run on the table to determine if the table needs to be reorganized before performing the reorg; however, this should not be the only indicator of whether a reorg needs to be done. For example, reorgchk flags a table for reorg if the table size in bytes is not more than 70% of the total space allocated for the table. Running a reorg in this case does not resolve the issue. For instance a table using 56% of the space allocated could be improved to 66%, but that still does not meet the 70% requirement, so it is again flagged for reorganization
- If it has been determined that a reorg improves a performance issue with the table/index, then careful consideration should be given on whether the reorg should be done online or offline while the system is down. System resources are required to do reorgs and this can slow down the system during the reorg. As well, consider that users will have to wait for locks during an online reorg.
- Carefully maintain the size of the database by archiving and trimming tables where appropriate to reduce the time needed for reorgs.
- If excessively locking is occurring, verify the following db2 lock parameters:
  - Maxlocks (maximum percentage of lock list before escalation)
  - Locklist (amount of memory allocated for locks)

A high-level database administrator may want to consider the following guidelines for tablespace page sizes and move appropriate tables into tablespaces with these page sizes:

Page size	Row size	Column count limit	Maximum capacity (DMS tablespace)
4 KB	4005	500	64GB
8 KB	8101	1012	128 GB
16 KB	16293	1012	156 GB
32 KB	32677	1012	512 GB

Highly experienced DB2 database administrators may want to consider the following two types of reorgs, each with advantages and disadvantages:

- Inplace reorg – Run while the database is active and users are doing work. An inplace reorg requires an instant super exclusive (Z) lock. When a table holds a Z lock, no concurrent application can read or update data in that table. However, a Z lock cannot be obtained when a pre-existing intent share (IS) lock is active. If the IS lock is never released, the reorg process for the table never starts. If the IS lock is released, inplace reorg starts and

subsequent IS locks that are obtained do not halt the reorg until the truncation phase of the reorg. At the truncation phase, the inplace reorg acquires another Z lock that prevents users from setting IS locks on tables to do work. Any requests from the user have to wait for the release of the Z lock, resulting in users seeing a slowness in the system. Inplace reorgs also require log space to store temporary data, which can grow quite large, when run against an index.

- Offline reorg – Run when the database is not active.  
The length of time to run the reorg depends on the database size. Tables are readable up until the copy phase of the reorg. Monitoring can tell you what phase the reorg is in. Any reads started before the copy phase can continue and the reorg waits for the read to end before table access is blocked. New reads cannot start until the copy phase completes.

### *Customization*

Customizing IBM Tivoli service management products can change the way you select information from the database. Some customizations include additional tables and columns. If you have customized IBM Tivoli service management products, you should carefully compare indexes to the user functions that use them. Ensure that you implemented the right balance of indexes.

## Moving a Collocated Database

IBM Tivoli Provisioning Manager (TPM) customers commonly install TPM with the database collocated. By moving the local database to a remote node, a large performance gain may be seen. The following white paper provides an approach for doing this:

<http://www.ibm.com/software/ismlibrary?NavCode=1TW101082>.

## Database Compression

IBM Tivoli Provisioning Manager testing has shown improvements in large workloads when database compression is used. Database Compression test results for other IBM Tivoli service management products were unavailable at the time this document was published.

## Oracle Database Notes

On Oracle, you should specify the following initialization parameters:

- Set the **CURSOR\_SHARING** parameter to SIMILAR or FORCE so that the user-entered literal values are converted to bind variables.
- Set the **NLS\_LENGTH\_SEMANTICS** parameter to CHAR when the database character set is a double-byte or unicode character set.
- Use Program Global Area (PGA) **WORKAREA\_SIZE\_POLICY=AUTO** setting to automatically size work areas.
- Ensure Optimizers are set to your current Oracle version and not an older one.  
**OPTIMIZER\_FEATURES\_ENABLE**

Use System Global Area (SGA) management by setting the following on Oracle:

**SGA\_TARGET**= memory value (for both Oracle 10g and 11g)

**SGA\_MAX\_SIZE**= memory value (both Oracle 10g and 11g)

**MEMORY\_TARGET**= memory value (for Oracle 11g)

**MEMORY\_MAX\_TARGET**= memory value (for Oracle 11g)

- Increase the process number to handle more concurrent users by setting the **PROCESSES**= initialization parameter to the maximum number of users who can access Oracle concurrently.

## IBM DB2 Notes

For convenience, IBM DB2 Version 9.5 and above provides an environment variable that can be used to optimize DB2 instances for use with a variety of products, including IBM Tivoli service management products. Set **DB2\_WORKLOAD=MAXIMO**. This accomplishes the same thing as if you set the following registry keys individually, using db2set commands to set the registry keys:

```
db2set DB2_SKIPINSERTED=ON
```

```

db2set DB2_INLIST_TO_NLJN=YES
db2set DB2_MINIMIZE_LISTPREFETCH=YES
db2set DB2_EVALUNCOMMITTED=YES
db2set DB2_SKIPDELETED=ON

```

After setting DB2\_WORKLOAD=MAXIMO, stop and start the database to apply the changes.

In addition to the DB2\_WORKLOAD=MAXIMO settings, use the following DB2 registry setting:

```

db2set DB2_USE_ALTERNATE_PAGE_CLEANING=ON1
db2set DB2_FMP_COMM_HEAPSZ=65536

```

For optimal performance, use the following DB2 database configuration settings:

```

db2 update db cfg for <dbname> using CHNGPGS_THRESH 402
db2 update db cfg for <dbname> using DFT_QUERYOPT 5
db2 update db cfg for <dbname> using LOGBUFSZ 1024
db2 update db cfg for <dbname> using LOGFILSIZ 8096
db2 update db cfg for <dbname> using LOGPRIMARY 20
db2 update db cfg for <dbname> using LOGSECOND 100
db2 update db cfg for <dbname> using LOCKLIST AUTOMATIC
db2 update db cfg for <dbname> using LOCKTIMEOUT 300
db2 update db cfg for <dbname> using MAXFILOP 61440 (UNIX / Linux)
db2 update db cfg for <dbname> using MAXFILOP 65535 (Windows)
db2 update db cfg for <dbname> using NUM_IOCLEANERS AUTOMATIC
db2 update db cfg for <dbname> using NUM_IOSERVERS AUTOMATIC
db2 update db cfg for <dbname> using SOFTMAX 1000
db2 update db cfg for <dbname> using STMTHEAP 20000

```

Note that while the recommended setting for SOFTMAX has a positive impact on runtime performance, crash recovery time will increase. If you use HADR, this wouldn't be cause for concern. In addition, if you use online backup, there may be a spike in I/O while data is being flushed to disk when the backup starts.

In addition to the above database configuration settings, if you run version 9.7 of DB2, use the following settings:

```

db2 update db cfg for <dbname> using CUR_COMMIT ON
db2 update db cfg for <dbname> using AUTO_REVAL DEFERRED
db2 update db cfg for <dbname> using DEC_TO_CHAR_FMT NEW
db2 update db cfg for <dbname> using STMT_CONC LITERALS

```

While these settings are generally the default settings in version 9.7, you must ensure that they are set, especially if you upgraded from a previous version of DB2 to version 9.7.

Note that when using DB2 version 9.7 with the statement concentrator, IBM Tivoli service management products recommend applying fixpack level 5 or later to avoid a potential performance issue with the statement concentrator. In addition, IBM Tivoli service management products recommend not using VARGRAPHICS when enabling the statement concentrator.

Use the following database configuration changes to take advantage of the self-tuning aspects of the database, and to reduce the overall management cost for the installation. You must have enough system resources for automatic tuning to work properly. If you are constrained on system resources, tune the settings manually:

```

db2 update db cfg for <dbname> using DATABASE_MEMORY AUTOMATIC
db2 update db cfg for <dbname> using PCKCACHESZ AUTOMATIC
db2 update db cfg for <dbname> using DBHEAP AUTOMATIC

```

---

<sup>1</sup> When using DB2 version 9.7

<sup>2</sup> When using DB2 version 9.5

```
db2 update db cfg for <dbname> using STAT_HEAP_SZ AUTOMATIC
```

In some instances, setting PCKCACHESZ AUTOMATIC can result in an overly large package cache that can have adverse effects on performance. If you suspect that you are experiencing this problem, set PCKCACHESZ=524288.

In general, IBM Tivoli service management products do not recommend turning on automatic maintenance settings, such as AUTO\_MAINT, AUTO\_TBL\_MAINT, AUTO\_RUNSTATS, and AUTO\_STMT\_STATS. Instead, schedule database maintenance activities during regular maintenance windows to avoid adversely affecting end user performance.

For optimal performance, use the following DB2 database manager settings:

```
db2 update dbm cfg using AGENT_STACK_SZ 1024 (UNIX / Linux)
db2 update dbm cfg using AGENT_STACK_SZ 1000 (Windows)
db2 update dbm cfg using RQRIOBLK 65535
db2 update dbm cfg using HEALTH_MON OFF
```

Set the following database manager configuration parameter to avoid potentially encountering a DB2 heap exception (com.ibm.db2.jcc.c.SqlException: Not enough storage is available in the "MON\_HEAP\_SZ" heap to process the statement):

```
db2 update dbm cfg using MON_HEAP_SZ AUTOMATIC
```

In addition, DB2 fenced processes can cause excessive memory use in some cases. Set the following parameter:

```
db2 update dbm cfg using keepfenced NO
```

If you need fenced processes and decide to use this feature of DB2, monitor the db2fmp process closely to ensure that memory usage does not become excessive.

When running DB2 on AIX 5.2 or above, set the AIX network parameter lo\_perf=0 to avoid an issue where loopback traffic becomes slow, and ping responds with out-of-order packets. See [APAR IY63671](#) for more details.

Additional DB2 configuration and hygiene information for the IBM Tivoli Provisioning Manager and IBM Tivoli Service Automation Manager products can be found in the following white papers:

<http://www.ibm.com/software/ismlibrary?NavCode=1TW101088>  
<http://www.ibm.com/software/ismlibrary?NavCode=1TW101089>

While written for these products, the information in this document may also apply to other IBM Tivoli service management products, especially in the areas of maintenance schedules and activities to manage the database (including audit tables).

## SQL Server Database Notes

Microsoft® SQL Server poses some challenges in environments, like IBM Tivoli service management products, that have multiple concurrent-user transactions. When you select records, SQL Server escalates locks from "record" to "page" to "table", based on an internal algorithm. If user queries are not efficient, database locks can impede other users who are accessing the system.

Some customers running IBM Tivoli service management products on SQL Server have improved database performance by turning on the SQL Server row-level locking that is supported by Microsoft. Turning on row-level locking reduces the locked areas of the database. However, row-level locking also can result in a higher number of locks, because locks are based on record instead of memory page. A document that describes how to implement row-level locking for IBM Tivoli service management products on SQL Server is available at the following link:

[http://www.ibm.com/support/docview.wss?rs=3214&context=SSLKT6&q1=%22row+level+locking%22&uid=swg21261979&loc=en\\_US&cs=utf-8&lang=en](http://www.ibm.com/support/docview.wss?rs=3214&context=SSLKT6&q1=%22row+level+locking%22&uid=swg21261979&loc=en_US&cs=utf-8&lang=en)

## User Queries

Most of the WHERE clauses in queries are generated by individual users on the List tabs of applications. This powerful feature of IBM Tivoli service management products can produce inefficient SQL. You can



improve the ease of use and convenience for users by setting the appropriate search types for database columns. Using appropriate search types also reduces the load on the database.

## Setting the Appropriate Search Type

You can configure search options. You can change the default search type of WILDCARD to TEXT or EXACT. TEXT and EXACT searches can use indexes. You specify the search type for a database column in the Database Configuration application. You also can change the search type for groups of columns. See <http://www.ibm.com/support/docview.wss?rs=3214&context=SSLKT6&uid=swg21261876>.

### *EXACT Search Type*

When an end user does a search, the default method of searching on many database fields is to use wildcards (SEARCHTYPE = WILDCARD). The WILDCARD search type causes IBM Tivoli service management products to construct a condition of the following form when the user enters "value" in a field on the List tab:

```
column like '%value%'
```

In wildcard searching, the database engine cannot use indexes. Searching without indexes can result in slower search times, especially on tables with many rows.

You can specify a search type of EXACT when word searching is not needed. For example, key fields (such as Work Order and Purchase Order) and value list fields (such as Work Order Status) can benefit from the indexing that is used in EXACT searches. EXACT searches use wildcarding only if a user explicitly enters wildcard characters on the List tab or in the WHERE clause.

### *TEXT Search Type*

Most tables have one or longer character columns for descriptions, memos, or remarks. You can provide a search type of TEXT, and a corresponding Oracle Text index or SQL Server full text catalog, for columns that have a lot of text. (Full text indexing is not available with IBM Tivoli service management products on IBM DB2®.) Text indexing puts some load on the database because of the constant need for background processing to keep the indexes synchronized. However, text indexing produces efficient word searching of description fields.

Specify a search type of TEXT on description fields that word searching is required. Use TEXT on description fields of tables with large numbers of rows (tens of thousands, for example).

The text search engine takes some time to refresh the indexes, so new records may not be found until the text search index refreshes itself.

On Oracle, you can modify the procedure `maximo_ts_job_call` to change the schedule of the synchronization process to any interval that you want.

On SQL Server, you can set and modify the population schedule for the Full Text Catalog. Use SQL Server 2000 Enterprise Manager or SQL Server 2005 Management Studio.

IBM DB2 does not provide text search, by default. However, you can install a text search tool and enable text searching by using `dbconfig` to change the search type to TEXT.

### *WILDCARD Search Type*

Tables with fewer than 2000 or 3000 records are often completely scanned regardless of indexes. The I/O cost to read the entire table is less than the average I/O cost of the index lookup plus the table lookup. The SEARCHTYPE value has no effect on database behavior when such scans are performed.

You can use the default search type of WILDCARD on description fields of tables that have a relatively small number of rows (2,000 or fewer, for example). Wildcard searching provides more flexibility for the end users. Tables with relatively few rows have no noticeable degradation in performance.

### *NONE Search Type*

The NONE search type prevents a column from being used in a WHERE clause, and still allows the display of the returned values on the List tab.

## Limit query times to improve performance

Database queries degrade system performance. Long running queries are especially disruptive. IBM Tivoli service management products provide two, configurable, system properties that determine how long queries can run. A control is provided in the user interface so users can cancel their own queries, before a configured timeout property cancels them. Both configurable attributes are set in the System Properties application.

### **mxedb.QueryTimeout**

This attribute determines how long an SQL query is allowed to run without producing a result set. The default value is 300 seconds. If an SQL query has not completed in the timeout interval, the SQL query is stopped.

### **webclient.ResultsetQueryTimeout**

This attribute determines how long Query By Example (QBE) queries are allowed to run without producing a result set. These queries are constructed from user input that is specified on forms and fields in the user interface. Users do not specify the SQL commands or verbs needed to run the query. Instead, QBE queries are translated into SQL queries by a QBE program that processes form input.

This attribute also controls how long determines how long filter queries, such as when you filter table rows, users, or when you ask to see all records for an object type (for example, all assets).

The default timeout for QBE and filter queries is 360 seconds; if users do not click the control to cancel a query, the query stops when this timer expires.

## Resetting the DATETIME Data Type of a Column

When you search on fields whose data type is DATETIME, the resulting database query uses TO\_TIMESTAMP in the WHERE clause. Queries that use TO\_TIMESTAMP are time-consuming.

You can edit the WHERE clause manually and replace TO\_TIMESTAMP with TO\_DATE. Queries that use TO\_DATE in the WHERE clause yield a faster search. However, you must edit the query each time you search on a DATETIME field.

In many cases, a better approach is to change the data type of routinely queried DATETIME fields to the data type DATE. Fields with a data type of DATE use TO\_DATE in the WHERE clause.

Use the Database Configuration application to change the data type of a column.

### **ALN type of fields**

When queried through normal filtering (QBE query), which is case insensitive, the ALN type fields having search type WILDCARD uses the UPPER function in the query; thus the index is not used. In customization, use UPPER case field if case does not really matter.

## Determining What Users Will Query

Users usually have a well-defined set of columns that they want to query in each application. You can identify these columns in user team interviews during implementation, or later, by examining reports of slow-running SELECT statements. You can then index these columns to improve system performance.

Users can create and save their own queries, and can share queries with other users. Saved queries are stored in a table named QUERY. You should periodically review these saved queries for inefficient conditions and use of unindexed columns, and to remove order by clauses.

Someone with SQL expertise can help create special-purpose queries (for example, return all PM work orders created since Monday of this week). Using these queries can save end users the effort of querying larger sets and sorting and scrolling through them.

You also can set up users with an efficient default query for their most-used applications so that they see their preferred record set when they enter the application. For example, in Work Order Tracking, you might specify a default query for a supervisor named "Smith" so that the query only shows work orders with "SMITH" in the Supervisor field.

You want to prevent users from performing queries that retrieve hundreds of thousands of records. These types of queries adversely affect database performance, causing slower user response times. To reduce

the likelihood of performance robbing queries, define default queries for your users, or educate them about how to use efficient queries.

## List Panel Order by Queries

Queries with order by clauses can be expensive, especially list panel queries since these are heavily used. It is recommended to edit the presentation XMLs and remove `order by` clauses on these queries.

## Restricting Querying in Applications

You can control or restrict user access to query features and user ability to query on specific columns by using a combination of methods:

- Application Designer
- Application cloning
- Security groups

### Application Designer

You can use the Application Designer to customize an application by adding or removing columns from the List tab. You can then ensure that the columns to be queried are all indexed.

### Application Cloning

You can clone an application and then use the Application Designer to create an alternate version of an application that has a restricted number of columns that can be queried.

### Security Groups

After you clone applications, you can use security groups to assign users to specific application clones.

You can also use security groups to prohibit access to the More Search Fields and Where Clause advanced query options. By prohibiting access to those options, you limit users to querying on the List tab of the application.

## Database Maintenance

### Key Performance Indicators

Key performance indicators (KPIs) display the state of systems and processes in IBM Tivoli service management products. Because KPIs can be user-defined, you should monitor them for efficiency.

#### *Live KPIs on Start Centers*

When a KPI is defined on a Start Center, you have a choice of how to retrieve information. You can run an immediate query to get the information, or you can retrieve the information from a table that is updated by a cron task.

The immediate query runs every time the Start Center opens. This approach can cause a long delay in opening the Start Center, and it puts a load on the database.

#### *KPI Queries*

You should periodically review queries that you write for KPIs. Check for SQL efficiency and index usage.

#### *KPI cron Task Frequency*

How up-to-date does your KPI information need to be? 5-minute intervals might sound like a good idea initially, but 60-minute or 120-minute intervals might be as useful to the people who want to see the information. Longer KPI cron task intervals reduce the load on the database from KPIs and can improve system performance.

In general, use longer rather than shorter KPI cron task frequency intervals when the value of the data to the end user is essentially the same at longer intervals.

### ***KPI Best Practices***

- Set all KPIs to retrieve their data from the KPI table.
- Use cron tasks to run KPI queries at reasonable intervals.

For information on moving cron tasks to a separate server, see Chapter 3: Scheduled Tasks (cron Tasks) on page 13.

## **Escalations**

Escalations are, in effect, batch users that do not have the overhead of a user interface. But escalations are very resource intensive whenever they do something.

### ***Efficiency and Frequency***

An escalation selects a set of records and performs a list of one or more actions on the result set. The columns in the WHERE clause of the selection should be efficiently indexed.

Set the frequency or schedule of an escalation according to its importance:

- An escalation that dispatches emergency work orders for critical safety issues might warrant a frequency of 5 minutes.
- An escalation that ensures that service requests from executives are dealt with promptly might need to run every 15 minutes.
- An escalation that closes work orders that were completed 90 or more days ago might need to run only once a week at an off-peak time.

## **Reports**

Review custom reports for efficient SQL and use of indexes. Most reports receive a WHERE clause from IBM Tivoli service management products. For all these cases, improving the efficiency of user queries also improves the efficiency of reports.

### **Properties file (7.1.1.6 and later)**

In the process automation engine version 7.1.1.6 and later versions, the following properties are available.

#### **Maximum lookup count**

The property `mxe.db.lookupmaxrow` controls how many records are returned by the lookup. Because unfiltered lookups can take a long time to perform, this property can be used to prevent database degradation by limiting the number of records being processed by a lookup.

#### **Fetch stop limits**

New properties are available to set a limit the number of objects that can be fetched from the database, and then constructed on the server into a single set. These properties are documented on the support Web site. See <http://www.ibm.com/support/docview.wss?uid=swg21412865>

## **Load Testing**

If possible, do load testing during the implementation phase to expose performance problems before you put IBM Tivoli service management products into production. If you have the equipment to perform load testing, you can use load testing after IBM Tivoli service management products are in production to determine if there is any performance impact from patches or from data growth over time.

## Chapter 7: Network and Bandwidth

This chapter discusses approaches to improving system performance by focusing on network issues and bandwidth.

### ***IBM Tivoli service management products on a Network***

Clients connect to the application over the network. The application also communicates with its various parts (application server, database, report server) over the network. If any segment of the network performs poorly, the end user experiences a system that is slow and hard to navigate.

IBM Tivoli service management products are Web-based products that operate on a request and response basis. If the requests and responses are delivered slowly, IBM Tivoli service management products have no control over response time.

Optimum network configurations for IBM Tivoli service management products should include the ability to produce 50 ms or faster round-trip packet response between the client and the server. The system needs enough bandwidth to support 6 kbps per user. Users may begin to experience performance degradation if the network does not operate within these parameters.

### ***Using Citrix or Windows Terminal Server***

You can use network caching, acceleration, and compression utilities to improve network performance.

Other options for resolving low bandwidth or high-latency network performance issues include services such as Windows<sup>®</sup> Terminal Server and Citrix. These services can help provide maximum performance between the Windows Terminal Server or Citrix client and the application server with a minimum of traffic between the Windows Terminal Server or Citrix server and the end user.

A benefit of running IBM Tivoli service management products through Citrix is that Citrix traffic is treated as “business traffic”. In some customer environments, business traffic can take priority over non-business traffic.

Some customers have found considerable improvements in network performance when they use Citrix or Windows Terminal Server. Note that IBM does not test or certify Citrix or other bandwidth tools.

### ***Using compression techniques to improve performance***

IBM Tivoli service management products provide performance improvement capabilities in a variety of areas. Sites that have bandwidth or latency issues can use one of the following techniques to improve performance:

- Configure hardware compression
- Configure HTTP compression (new in IBM Tivoli service management products version 7)
- Use gzip compression

Each of these compression options is mutually exclusive of the others. That is, you can employ hardware compression or software compression or use built-in gzip.

### **Hardware compression using network appliances**

Customers typically use commercial network appliances to optimize network performance. Network appliances, such as Juniper, Riverbed, and others, provide compression and caching features. Network appliances can help compress data and optimize bandwidth. Customers report that network appliances can prove to be beneficial to system performance, especially in a high-latency environment

Hardware compression affects all servers in the configuration.

### **HTTP compression**

HTTP compression is a capability that can be built into Web servers, such as IBM HTTP Server, and Web browsers to make better use of available bandwidth, and provide faster transmission speeds.

HTTP compression affects all servers in a cluster. HTTP data is compressed before it is sent from the server (Note that hardware load balancers that bypass the HTTP Server cannot employ this method):

- Compression-compliant browsers announce, to the server, what compression methods the browser supports, so the server can provide the compressed data in the correct format.
- Browsers that do not support compression simply download uncompressed data.

Data can be compressed by using a compression module (like Apache's mod\_deflate) or by using gzip modules. The compression scenario is dictated by what the server software supports.

The following compression scenarios have been tested and have been found to be good solutions for low bandwidth locations. Gzip compression is supported; however, the CPU overhead associated with it is costly, for most applications. It is a best practice to compress data with a compression module instead of using gzip to compress it.

- IBM HTTP Server  
Use Apache mod\_deflate and set DeflateCompressionLevel to 3 to improve response time in environments that have low bandwidth and high latency.
- Oracle WebLogic  
Use Apache as an external http server. Use mod\_deflate and set DeflateCompressionLevel to 3 to improve response time in environments that have low bandwidth and high latency.

See [http://httpd.apache.org/docs/2.2/mod/mod\\_deflate.html](http://httpd.apache.org/docs/2.2/mod/mod_deflate.html) for information about configuring mod\_deflate.

## Runtime gzip compression

The IBM Tivoli service management products distribution contains the gzip utility, which you can configure to compress files before sending them to browser users. This does reduce bandwidth usage, but it also increases CPU and memory consumption. HTTP server compression is more efficient.

For information about how to set up to use gzip, see the following support Web page:

[http://www.ibm.com/support/docview.wss?rs=3214&context=SSLKT6&q1=cache&uid=swg21262009&loc=en\\_US&cs=utf-8&lang=en](http://www.ibm.com/support/docview.wss?rs=3214&context=SSLKT6&q1=cache&uid=swg21262009&loc=en_US&cs=utf-8&lang=en)

## Image and JavaScript Browser Caching

You can enable a browser file caching filter on the application server. This lets the browser store images, CSS files, and JavaScript files locally. Storing images and files locally benefits performance in two ways:

- Images and files do not need to be constantly requested from the server.
- Because images are not downloaded as often, less bandwidth is required.

Browser file caching is not enabled by default. *We recommend that you enable browser file caching.* For information on how to enable this feature, see the following support Web page:

[http://www.ibm.com/support/docview.wss?rs=0&q1=swg21262009&uid=swg21262009&loc=en\\_US&cs=utf-8&cc=us&lang=en](http://www.ibm.com/support/docview.wss?rs=0&q1=swg21262009&uid=swg21262009&loc=en_US&cs=utf-8&cc=us&lang=en)

For Microsoft Internet Explorer 6, set the number of days until refresh is enabled to 32 days or longer. Modify the web.xml file. Set the max-age=<seconds> value to 2764800 or higher. 2764800 seconds is 32 days.

## Use Quality-of-Service Guarantees for Network Traffic

You can prioritize IBM Tivoli service management products traffic over standard Web traffic. Contact your network administrator for details about prioritizing IBM Tivoli service management products traffic.

# Chapter 8: Server OS Configuration

## AIX Configuration

The following outlines the recommended AIX operating system settings:

### Networking:

```
/usr/sbin/no -r -o sb_max=6192000
/usr/sbin/no -r -o tcp_sendspace=4096000
/usr/sbin/no -r -o tcp_recvspace=4096000
/usr/sbin/no -r -o udp_sendspace=65536
/usr/sbin/no -r -o udp_recvspace=655360
/usr/sbin/no -r -o rfc1323=1
/usr/sbin/no -r -o ipqmaxlen=150
/usr/sbin/no -r -o clean_partial_conns=1
/usr/sbin/no -r -o tcp_keepidle=600
/usr/sbin/no -r -o tcp_keepintvl=10
/usr/sbin/no -r -o tcp_keepinit=40
/usr/sbin/no -r -o tcp_timewait=1
/usr/sbin/no -r -o tcp_finwait2=60
/usr/sbin/no -r -o tcp_ephemeral_low=1024
```

### Resource (ulimit):

```
time(seconds) unlimited
file(blocks) unlimited
data(kbytes) unlimited
stack(kbytes) 4194304
memory(kbytes) unlimited
coredump(blocks) unlimited
nofiles(descriptors) unlimited
threads(per process) unlimited
processes(per user) unlimited
```

### Process:

```
chdev -l sys0 -a maxuproc='4096'
```

### Virtual Memory:

```
vmo -p -o lru_file_repage = 0
vmo -p -o maxclient% = 90
vmo -p -o maxperm%=90
vmo -p -o minperm%=5
```

## RedHat Linux Configuration

### Networking:

```
sysctl -w net.ipv4.ip_local_port_range="1024 65535"
```

## Windows Configuration

Set the following networking parameters located under the Windows registry key  
HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters:

TcpTimedWaitDelay	dword:0000001e (30)
StrictTimeWaitSeqCheck	dword:00000001 (1)
MaxFreeTcbs	dword:00011940 (72000)
MaxHashTableSize	dword:0000ffff (65535)
TcpWindowSize	dword:0000ffff (65535)
EnableDynamicBacklog	dword:00000001 (1)
MinimumDynamicBacklog	dword:00000032 (20)
MaximumDynamicBacklog	dword:000003eb (1000)

DynamicBacklogGrowthDelta    dword:0000000a (10)  
Interfaces\TcpAckFrequency    dword:00000001 (1)  
MaxUserPort                    dword:0000ffff (65535) <For Windows Server 2003. See below for  
Windows Server 2008>

Note that if some of the above parameters are not currently in the registry, they may be added by following the instructions at this link: <http://technet.microsoft.com/en-us/library/cc772023.aspx>.

For Windows Server 2008, the default dynamic port range has changed. The new default start port is 49152 and the default end port is 65535. Therefore, 16384 ports are available by default (not 5000). To view the dynamic port range, start the Command Prompt and use the netsh command:  
netsh int ipv4 show dynamicport tcp

To change the dynamic port range for the maximum number of ports allowed, issue the following command:

```
netsh int ipv4 set dynamicport tcp start=1025 num=64510
```

Note that the minimum start port is 1025 and the maximum end port cannot exceed 65535.

## Chapter 8: Client Workstation Configuration

Customers report that client workstation configuration is, initially, the most important area to focus on when users experience performance issues.

Hardware and software products constantly evolve. IBM Tivoli service management products can run on some older hardware and software platforms, but the best performance is achieved by using the newest supported operating systems and the latest hardware. Robust workstations provide better performance. For example, client workstations must have, at least, the minimum RAM required by the operating system and all other applications that it runs, but adding additional RAM can boost performance. Similarly, using workstations with multiple CPUs and higher clock speeds also boosts performance.

A support matrix is available at <http://www.ibm.com/support/docview.wss?uid=swg21067036>. The matrix provides a list of supported operating systems, databases, applications, and some infrastructure requirements. In addition to the requirements shown in the matrix, the following lists additional requirements:

### Monitor Resolution

Notebooks and desktop systems must have a monitor capable of supporting a resolution of 1280 x 1024 to fit the entire user interface on the screen.

### Browser

Microsoft Internet Explorer 7 and Mozilla Firefox version 3 are supported browsers.

### File Reader

Install Adobe® Reader 6, or newer versions, on the client workstation to view PDF files.

### Antivirus Software

Install antivirus software on all client workstations, but scanning for viruses can affect system performance. If possible, schedule virus scans during low usage periods.



## Chapter 9: Performance Improvement Tips and Customer Suggestions

This chapter contains various tips that have been derived both from experience with IBM Tivoli service management products and from customers. You can use these tips to improve system performance.

### ***Apply the Latest Patch***

Apply the latest available patch, fix pack, or hot fix for your release of IBM Tivoli service management products. Patches and hot fixes contain fixes for application issues and often contain features that can improve system performance.

Patches can contain fixes specifically designed to improve performance. Patches also contain new features and parameters that help you to better monitor and debug system performance issues.

### ***Staged Roll-out for New Users***

The first time a user logs in to IBM Tivoli service management products, there is an initial setup of the user's start center. This setup results in many database queries and inserts. If too many users perform this initial login simultaneously, all available database resources could be consumed by the setup operations. Therefore, IBM Tivoli service management products recommend that in large deployments, new users are brought on-board in a staged manner to avoid this bottleneck.

### ***Start Center Portlets***

A large number of portlets on the start center can contribute to slow performance. IBM Tivoli service management products recommend that start center tabs be used so that a limited number of portlets are displayed on any given tab. Some customers have suggested using default start centers without any portlets to help alleviate this issue.

### ***Start Center Result Set and Pre-defined Queries***

The start center is a heavily used component of IBM Tivoli service management products. As such, it is important to ensure that any start center result set queries are well tuned to avoid performance issues. In general, all pre-defined queries should be well tuned for optimal performance across the system.

### ***Client Workstation Suggestions***

This section contains suggestions for the client workstation that can improve system performance.

#### **Limit or Prevent Some Workstation Activities and Processes**

Customers report that some user activities and workstation processes can degrade performance. You should check for and monitor these activities and processes, and respond as necessary.

##### ***Monitor the Network for Streaming Audio and Video***

Customers report that monitoring the network in order to prevent users from using streaming audio and video can noticeably increase the bandwidth available to the system.

##### ***Have Only One Network Link Active***

If a user has both a wireless network link and a LAN link active, it can cause system performance issues. Limit users to one active network link.

##### ***Monitor for Hung Processes and Applications***

Processes and applications that are not responding (are hanging) use memory and can affect the performance of the client workstation. A workstation can have hung processes or applications that the user might be unaware of.

Use system tools, such as the Windows Task Manager, to check for and end hung processes and applications.

## ***Garbage Collection Parameters for WebSphere Application Server***

On WebSphere Application Servers, garbage collection for objects with remote references is not done in an optimal manner. To correct this, set the following JVM parameters using the administrative console. In the console, go to **Servers ->Application servers->servername-> Java and Process Management->Process Definition->Java Virtual Machine**.

- -Dsun.rmi.dgc.ackTimeout=10000
- -Djava.net.preferIPv4Stack=true

For additional information on garbage-collection parameters for WebSphere Application Server, follow this link:

[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/urun\\_rconfproc\\_jvm.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/urun_rconfproc_jvm.html)

## ***Limiting Use of E-Audit***

E-audit is a feature that is designed to support industries, such as life sciences, that are required to track changes by user, date, and time. Because e-audit creates a transaction record for every change in the system, it has an effect on system performance. You should weigh the benefits of enabling e-audit against the impact on performance.

By default, when you enable e-audit for an object, such as the Workorder object, all fields are e-audited.

You should limit the number of objects that you e-audit. For each object that is e-audited, limit the number of fields that you e-audit.

Limiting the use of e-audit to only those objects and fields that require an audit trail limits the impact on system performance.

## ***Archive and delete historical data***

Deleting unnecessary data from the database helps to optimize its performance. IBM Maximo Archiving with Optim Data Growth Solution 7.1 provides a way to archive and delete historical data from your database. Once archived, you can browse the data in IBM Optim™, or you can restore the data to a reporting database where you can view it in applications and reports.

Archiving historical data can also facilitate upgrading to a new version of IBM Tivoli service management products because the historical data is not critical to day-to-day operations and might not need to be included in the new system.

## Chapter 10: Troubleshooting

This chapter provides information on troubleshooting system performance issues.

IBM Tivoli service management products have features that log health statistics. In the most recent versions of IBM Tivoli service management products, these features are turned on by default. These features and their associated data help you troubleshoot performance problems.

### ***Documents or Files to Collect for Troubleshooting***

Before you troubleshoot for system performance issues, collect the following information and documents or files.

- Version and release information for each of the IBM Tivoli service management products used.
- Product, version, and release information for your database and application-server software.
- All application server log files. If you have multiple servers, collect logs for each server.
- Log files. If you have multiple servers, collect logs for each server.
- Garbage collection information from the application servers. The application servers must be enabled for verbose garbage collection. For more information, see *Displaying Garbage Collection Statistics on the Server* on page 37.
- Configuration settings, which are defined in the properties file, under applications/properties.
- If the problem is related to deadlock or hanging, generate 5 to 10 thread dumps on the server instances at two-minute intervals.
  - For information on generating thread dumps for WebSphere Application Server, follow this link:  
<http://www.ibm.com/support/docview.wss?rs=0&uid=swg21138203>
  - For information on generating thread dumps for WebLogic, consult the WebLogic documentation.
- If the problem is an out-of-memory issue, collect the heap dump files associated with the server instance.
  - For information on collecting heap dump files for WebSphere Application Server, follow this link:  
[http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/tprf\\_generatingheapdumps.html](http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/tprf_generatingheapdumps.html)
  - For information on collecting heap dump files for WebLogic, consult the WebLogic documentation.

### ***Troubleshooting Performance Problems***

You use the troubleshooting procedures in a development or test environment for performance analysis and debugging. The procedures generally should not be used in a production environment. Use these procedures in a production environment only if you cannot isolate the problem in a test environment.

### **Setting Up a Stand-alone Application Server for Debugging**

Debugging SQL or business object problems and using detailed logging when you reproduce other problems can generate large logs and slow the system.

You can set up a stand-alone application server for debugging. However, if issues that you are investigating cannot be replicated in a test environment, setting up stand-alone loggers can provide limited benefit.

Set the parameters so that the server does not execute cron tasks. The stand-alone application server can be on a separate computer. Using a separate computer lets you easily stop and start the server to change logging parameters and reproduce problems with a single user.

## Debugging Parameters

You can use debugging parameters with IBM Tivoli service management products. You can configure the debugging parameters in the file `maximo.properties`.

- `mx.e.db.fetchResultLogLimit` (5.2 P02A and later)
- `mx.e.db.logSQLTimeLimit` (5.2 P02A and later)
- `mx.e.mbocount` (5.2 P05 and later - different implementations)
- `mx.e.db.logSQLPlan` (6.0 P01 and later; Oracle only)
- `mx.e.db.sqlTableScanExclude` (6.0 P01 and later; Oracle only)

For information about implementing the debugging parameters, see the following IBM Support Web page: [http://www.ibm.com/support/docview.wss?rs=3214&context=SSLKT6&uid=swg21291250&loc=en\\_US&cs=utf-8&lang=en](http://www.ibm.com/support/docview.wss?rs=3214&context=SSLKT6&uid=swg21291250&loc=en_US&cs=utf-8&lang=en)

## Loggers:

Logging is performed in IBM Tivoli service management products using a logger object. Loggers define a hierarchy and provide the run-time control to print statements or not. You make changes in the Logging application, located at System Configuration → Platform Configuration. You must be logged in as a user with administrative privileges to modify the logging, apply, and save changes.

The root logger, `log4j.rootLogger`, should be kept no higher than the INFO level to avoid performance impacts to the system.

The SQL logger, `log4j.logger.maximo.sql`, should be kept at the ERROR level for optimal performance.

For more information on the logging functionality in IBM Tivoli service management products, refer to the information at the following link:

<https://www.ibm.com/support/docview.wss?uid=swg21264064>.

## Displaying Garbage Collection Statistics on the Server

You can turn on garbage collection (GC) verbosity to find out how the garbage collection routine is functioning on the application server. in WebSphere Application Server or WebLogic.

In WebSphere Application Server, specify this parameter in the administration console by selecting Servers → Application servers → {ServerName} → Process Definition → Java Virtual Machine and placing a check mark by verbose garbage collection.

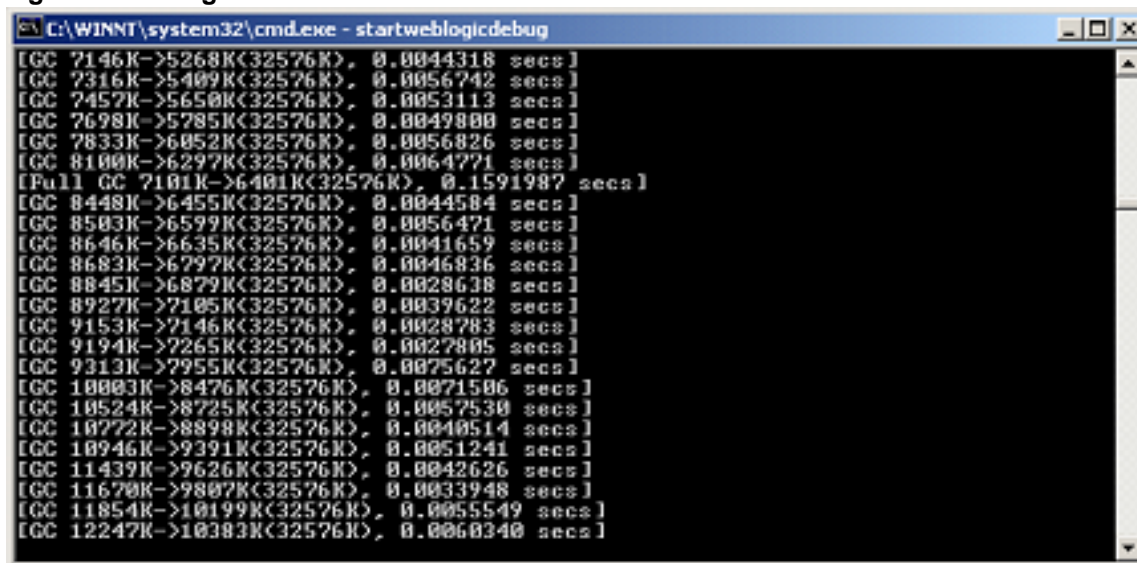
In WebLogic, set the following parameter in the start command file: `JAVA_OPTIONS=-verbose:gc`.

If you do not find the `JAVA_OPTIONS` parameter, you can set the `-verbose:gc` parameter before the `java` command. For example, `java -verbose:gc`.

After you set garbage collection to verbose, the console or the log file displays how much time is spent for each garbage collection cycle.

The following figure shows the messages that the verbose garbage collection setting returns:

Figure 6: Garbage Collection Statistics



```
C:\WINNT\system32\cmd.exe - startweblogicdebug
[GC 7146K->5268K(32576K), 0.0044318 secs]
[GC 7316K->5409K(32576K), 0.0056742 secs]
[GC 7457K->5650K(32576K), 0.0053113 secs]
[GC 7698K->5785K(32576K), 0.0049800 secs]
[GC 7833K->6052K(32576K), 0.0056826 secs]
[GC 8100K->6297K(32576K), 0.0064771 secs]
[Full GC 7101K->6401K(32576K), 0.1591987 secs]
[GC 8448K->6455K(32576K), 0.0044584 secs]
[GC 8503K->6599K(32576K), 0.0056471 secs]
[GC 8646K->6635K(32576K), 0.0041659 secs]
[GC 8683K->6797K(32576K), 0.0046836 secs]
[GC 8845K->6879K(32576K), 0.0028638 secs]
[GC 8927K->7105K(32576K), 0.0039622 secs]
[GC 9153K->7146K(32576K), 0.0028783 secs]
[GC 9194K->7265K(32576K), 0.0027805 secs]
[GC 9313K->7955K(32576K), 0.0075627 secs]
[GC 10003K->8476K(32576K), 0.0071506 secs]
[GC 10524K->8725K(32576K), 0.0057530 secs]
[GC 10772K->8898K(32576K), 0.0040514 secs]
[GC 10946K->9391K(32576K), 0.0051241 secs]
[GC 11439K->9626K(32576K), 0.0042626 secs]
[GC 11670K->9807K(32576K), 0.0033948 secs]
[GC 11854K->10199K(32576K), 0.0055549 secs]
[GC 12247K->10303K(32576K), 0.0060340 secs]
```

- A line that starts with “Full GC” indicates that a complete garbage collection cycle was run.
- A line that starts with “GC” indicates that a minor garbage collection cycle was run.
- The value to the left of the arrow (->) indicates the memory (or number of live objects) before garbage collection.
- The value to the right of the arrow indicates the memory (or number of live objects) after garbage collection.
- The value in angle brackets (< >) indicates the total memory allocated for the heap at the time.

Several tools are available to assist with garbage collection analysis, such as the [IBM Pattern Modeling and Analysis Tool for Java Garbage Collector](#) and the [IBM Support Assistant Tool](#).

## Troubleshooting Performance Issues in Application Server Configuration

This section describes issues that can result in poor performance in the application server. It also outlines recommendations that can help you to resolve those problems.

### Monitoring the Issues

The following table outlines the most common issues and their causes:

**Table 5: Performance Problems**

Problem	Cause
Out of memory	<ul style="list-style-type: none"><li>• Many users are directed toward a single instance of the application server.</li><li>• A single instance of the application server is performing an excessive amount of work.</li><li>• Heap memory is set too small.</li><li>• Operations are fetching too much data at once from the database.</li><li>• Bugs in the code hold onto objects in memory.</li><li>• Dsun.rmi.dgc.ackTimeout is set too high.</li></ul>
Deadlocks	The Java code in the application or the Java Virtual Machine causes active threads to wait for resources that other threads are holding. This behavior eventually can cause all threads to wait.

Study the log files of your application server to determine the garbage collection output, the heap dumps, and thread dumps.

### ***IBM Tivoli Monitoring Agent***

The IBM Tivoli<sup>®</sup> Monitoring Agent is a product that is used to monitor and manage system and network applications on a variety of operating systems, track the availability and performance of your enterprise system, and to provide reports to track trends and troubleshoot problems.

For the application server, the Tivoli Monitoring Agent provides the following key monitoring capabilities:

- Memory utilization of each JVM: Ensure that each JVM has available memory.
- JVM statistics: Ensure that the JVM is healthy and running within normal specifications.
- Names and number of MBO Business Objects.
- cron tasks: Ensure that all cron jobs are scheduled and running properly.
- Product licenses: Determine which licensed products are installed.
- Connected Users: Use this metric to ensure that no more than 50 concurrent users are connected to a JVM.

For additional information, see

[http://www.ibm.com/support/docview.wss?rs=3214&context=SSLKT6&q1=itm&uid=swg24025477&loc=en\\_US&cs=utf-8&lang=en](http://www.ibm.com/support/docview.wss?rs=3214&context=SSLKT6&q1=itm&uid=swg24025477&loc=en_US&cs=utf-8&lang=en)

### **Addressing the Issues**

- Monitor the user load and server load. Use clustering and load balancing techniques to increase the number of server instances as necessary.
- On 32-bit systems, use 1 GB to 1.5 GB of heap memory. On 64-bit systems, use 2.5 GB of heap memory.
- Enable fetch stop limits.
- Use separate clusters for the user load system and background systems, such as cron tasks and integration module data loading.

# Appendix A: Setting up Two Clusters

This appendix provides information about how to set up two clusters in an environment in which the integration framework is deployed.

## Overview

Every external system that IBM Tivoli service management products communicate with should be set up with its own sequential inbound and sequential outbound queues. If you also set up each sequential queue with its own cron task, administering the queues is easier.

For example, you might set up a system with three clusters, and call them UI, Q, and Cron.

- UI cluster: This cluster provides connection for application end users. It hosts the Web application and sequential outbound queues.
- Q cluster: This cluster is principally for hosting continuous and sequential inbound queues.
- Cron cluster: This cluster is where the other cron tasks are configured. The integration framework cron tasks [File, Interface table, JMS sequential inbound queues] should be run on the Q cluster. Make sure that these cron tasks do not run on any other cluster other than the Q cluster. The MEA cron tasks for outbound sequential queues need to run on all the clusters that host the outbound sequential queues. So if the outbound queues are created on the Q and the cron clusters the corresponding outbound JMS cron tasks need to be run on those clusters too [along with the UI cluster].

The UI cluster is where users generate transactions when they save work in IBM Tivoli service management products. The UI cluster should therefore host sequential outbound queues to send out user-generated transactions to external systems.

The purpose of having a Q cluster is to separate the inbound queue processing from the interactive users. Separating the inbound queue processing this way prevents inbound traffic from affecting the user interface performance. Some setups may have been configured to allow inbound MEA transactions to generate outbound MEA transactions. In such cases the Q cluster needs to have sequential outbound queues setup too.

The optional Cron cluster is set up for other cron tasks. Setting up a third cluster for cron tasks is helpful and should bring the system configuration close to its ideally segregated potential.

Cron cluster setup details are not described here or in the accompanying figures. If cron tasks generate outbound transactions, then the Cron cluster needs its own set of sequential outbound queues. Outbound queue setup on the Cron cluster would be similar to outbound queue setup on the UI cluster and the Q cluster.

## Example: Setting Up the UI Cluster and the Q Cluster

Following is a summary list of steps that you can use to set up the UI cluster and the Q cluster discussed in the previous section. The steps assume that there are two clusters. The UI cluster consists of eight JVMs. The Q cluster consists of two JVMs.

The naming convention uses the queue name appended with `_UI` or `_Q`, for example, `sqoutbd_ui` and `sqinbd_q`. Because each external system should have its own outbound queue and inbound queue, there are multiple instances of OUT and IN queues, for example, `SAP_SQOUT_UI`, `ORA_SQIN_Q`, and so on. (The example assumes you are using WebSphere Application Server.)

## Steps to Set Up the UI Cluster and the Q Cluster

1. Create two EAR files to deploy the applications on the two clusters.
  - For the UI cluster, edit the `deployment-application.xml` file to comment out `MEAWEB`. From the modules `mboejb` `ejb-jar.xml` make sure that the MEA message driven bean (MDB) and the MEA MDB for the error queue are commented out. Make sure that the file `ibm-ejb-jar-bnd.xmi` also has the MEA MDB and the MEA MDB for the error queue commented out. [`maximo_ui.ear`]

- For the Q cluster, edit the files `ejb-jar.xml` and the `ibm-ejb-jar-bnd.xmi` in the module `mboejb` to include the MEA MDB and the MEA MDB for the error queue sections. Make sure that those sections are uncommented in the `ejb-jar.xml` as well as the `ibm-ejb-jar-bnd.xmi` files.[`maximo_q.ear`]
- 2. Create eight JVMs for the UI cluster.
- 3. Create two JVMs for the Q cluster.
- 4. Create two clusters, UI and Q.
- 5. Add the eight JVMs that you created in step 3 to the UI cluster.
- 6. Add the two JVMs that you created in step 4 to the Q cluster.
- 7. Deploy two applications, `maximo_ui.ear` one to the UI cluster, `maximo_q.ear` one to the Q cluster.
- 8. Create two JMS buses for the UI and Q clusters, for example, `meajmsbus_ui`, and `meajmsbus_q`.
- 9. Create two database schemas, which will be used by the data sources that we create in step 11. Make sure the schemas created have table creation permission.
- 10. Create two JDBC providers.
- 11. Create two data sources – associate the two schemas created in step 9 to these data sources. These data sources are used to create and store queue messages.
- 12. Create two J2C authentication aliases that point to the two schemas.
- 13. Add each cluster as a member to the respective bus. This creates an ME [messaging engine] for each cluster. At this time choose a data store and select an existing data source created in the previous steps. Make sure you choose the data source meant for the respective cluster. Set the corresponding schema name and ensure that the create tables check box are checked.
- 14. Set the message store authentication alias for each of the MEs in their corresponding bus.
- 15. Add four destinations to the Q bus, for example:
  - `sqoutbd_q`
  - `sqinbd_q`
  - `cqinbd_q`
  - `cqinerrbd_q` – set this as the Exception destination for the `cqinbd_q` as well as for itself [loop-back pattern]
- 15. Add a single destination to the UI bus, for example:
  - `sqoutbd_ui`
- 16. Add these JMS objects to the UI cluster:
  - One queue, and associate it with its respective bus and destination. Define the queue at the cluster scope [scope of the UI cluster]. Set the JNDI name as `jms/maximo/int/queues/sqout`.  
  
For example, associate `sqout_ui` with `meajmsbus_ui` and `sqoutbd_ui`.
  - One connection factory [scope of the UI cluster] and associate it with its respective bus. Set the JNDI name as `jms/maximo/int/cf/intcf`.  
  
For example, associate `conn_fact_ui` with `meajmsbus_ui`.
- 17. Add these JMS objects to the Q cluster:
  - Four queues, [all defined in the scope of the Q cluster] and associate them with their respective buses and destinations. For example, associate `sqout_q` with `meajmsbus_q` and `sqoutbd_q`. Set the JNDI name as `jms/maximo/int/queues/sqout`. Note that the JNDI name is the same with the UI cluster but the scope is different. Associate `sqin_q` with `meajmsbus_q` and `sqinbd_q`. Set the JNDI name as `jms/maximo/int/queues/sqin`.



Associate `cqin_q` with `meajmsbus_q` and `cqinbd_q`. Set the JNDI name as `jms/maximo/int/queues/cqin`.

- Associate `cqinerr_q` with `meajmsbus_q` and `cqinerrbd_q`. Set the JNDI name as `jms/maximo/int/queues/cqinerr`.
- One connection factory [scope of the Q cluster] and associate it with its respective bus. Set the JNDI name as `jms/maximo/int/cf/intcf`. For example, associate `conn_fact_q` with `meajmsbus_q`.
- Two activation spec, [scope of the Q cluster] and associate it with its respective bus and continuous queue and error queue JNDI name. Set the JNDI name as `intjmsact` and `intjmsacterr`. For example, associate `intjmsact` with `meajmsbus_q` and `jms/maximo/int/queues/cqin`.

For queues and connection factories, use JNDI names that are based on the naming convention shown in the steps (for example, `xxx_ui` and `yyy_q`). Also, be sure that these names match the names set in the external system setup in the integration framework.

If you model your clustering architecture on the example setup, be sure that JNDI names for connection factories and outbound queues are the same in the UI cluster and the Q cluster.

## Result of Setup

The result of the previous procedure is the creation of two messaging engines, one for each bus-cluster combination. Each engine has its own schema (pointed to by the data source of its respective cluster). One schema is for outbound sequential queue traffic of the UI cluster. The other schema is for inbound (sequential and continuous) and outbound (sequential) traffic of the Q cluster.

## Accessing the Sequential Queue

A potential issue in the example setup can occur if the UI cluster needs to access the sequential queue. This issue may happen if a user tries to import a file for data loading.

In such a situation, because there is no sequential queue visible at the UI cluster level, an error results. You can resolve the error by allowing the user to log in to the Q cluster application to perform the import operation. To allow the user to log in, open the respective port number on the virtual host setup.

## Example Deployment and Configuration

The following figures illustrate the example deployment and configuration.

**Figure 7: System Layout**

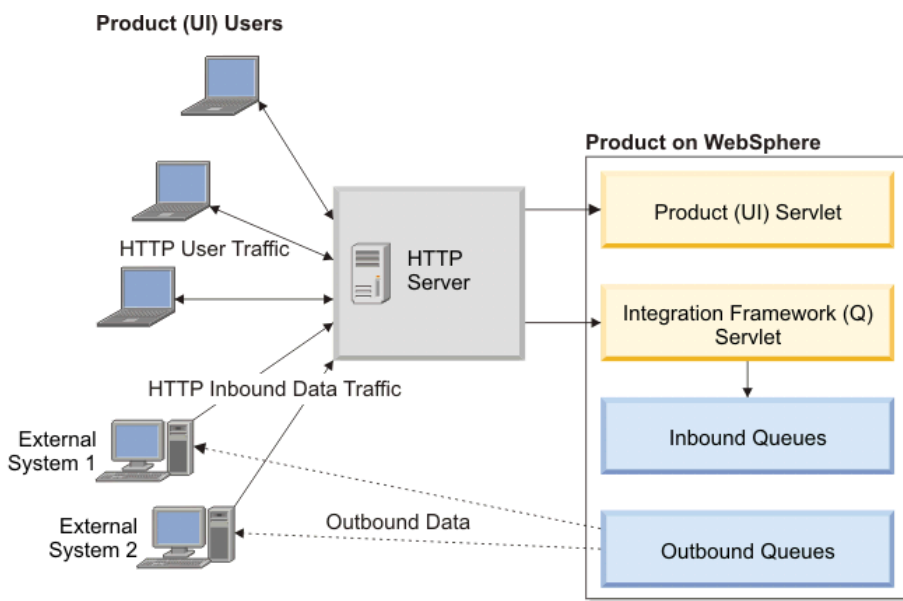
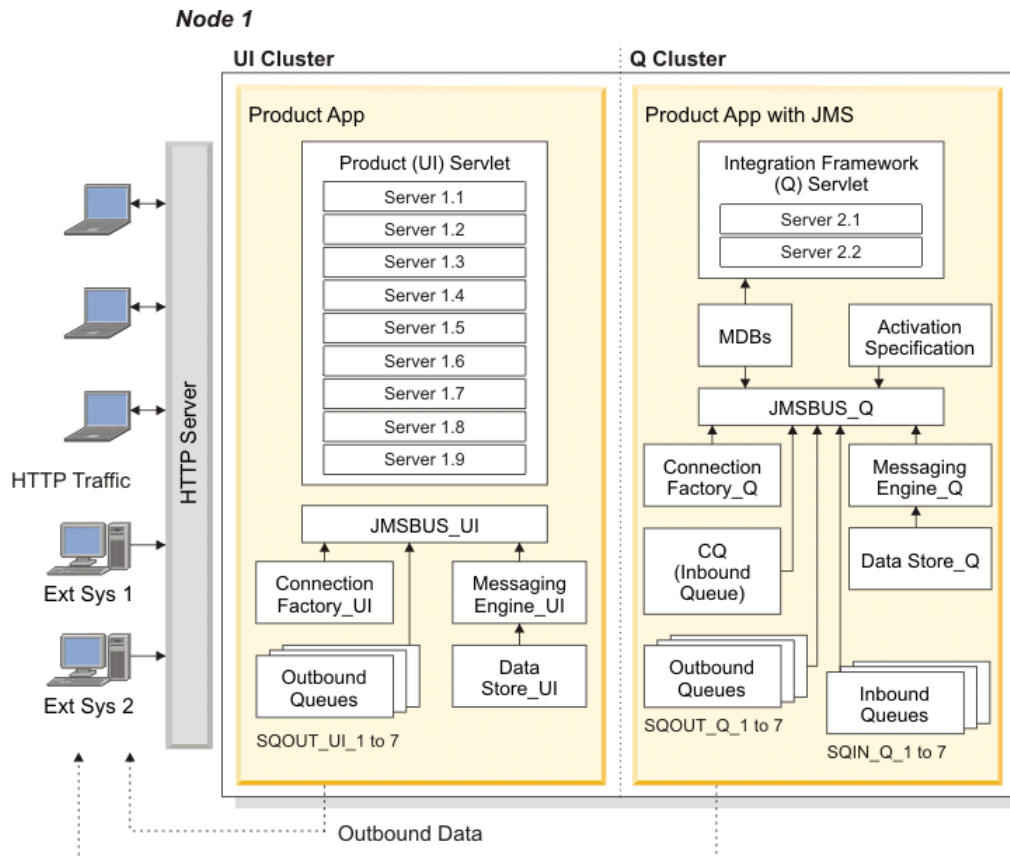


Figure 8. Multiple Cluster Configuration



When configuring multiple clusters, you may need to tune several settings. The numbers that are listed here are estimates. You may need to revise the values to optimize performance.

- Number of message-driven beans: 5
- Batch size: 5 connections
- JMS connection factory pool size: 100
- Default thread pool size: 50
- Memory allocation: 1.792 GB
- cron task frequency for JMS queues: 30 seconds

## References

[IBM Tivoli Provisioning Manager & Tivoli Service Automation Manager Version 7: Database Configuration and Hygiene Recommendations](#) (Leitch), IBM Integrated Service Management (ISM) Library white paper

[IBM Tivoli Provisioning Manager Version 7: A Deployment Engine Cluster Solution](#) (Leitch, Zhao, Postea), IBM Open Process Automation Library (OPAL) white paper

[IBM Tivoli Provisioning Manager 7.1.1: A DBMS Movement Solution](#) (Leitch, Zhao, Kaye-Cheveldayoff), IBM Integrated Service Management (ISM) Library white paper

# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation  
Licensing 2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106-0032, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:**

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

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.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation  
Software Interoperability Coordinator, Department 49XA  
3605 Highway 52 N  
Rochester, MN 55901  
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp.  
Sample Programs. © Copyright IBM Corp. \_enter the year or years\_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

## ***Trademarks***

For trademark attribution, visit the IBM Terms of Use Web site (<http://www.ibm.com/legal/us/>)

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

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

Intel, the Intel logo, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

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

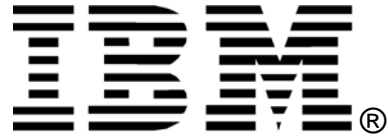
Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

VMware, the VMware "boxes" logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the "Marks") of VMware, Inc. in the United States and/or other jurisdictions.

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



Printed in USA