

# IBM Tivoli Security Compliance Manager Version 5.1

---

## Deployment and Tuning Guide

Document Version Number: 1.2  
Document Creation Date: 4.30.04  
Document Update Date: 6.24.04

Copyright International Business Machines Corporation 2004. All rights reserved.  
U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA  
ADP Schedule Contract with IBM Corp.

## Table of Contents

|                                                     |      |
|-----------------------------------------------------|------|
| Table of Contents .....                             | iii  |
| Preface.....                                        | VI   |
| Preface.....                                        | VI   |
| Who should read this book .....                     | VI   |
| Publications .....                                  | VI   |
| IBM Tivoli Security Compliance Manager library..... | VI   |
| Related publications .....                          | VII  |
| IBM DB2 Universal Database™ .....                   | VII  |
| Accessing publications online.....                  | VII  |
| Accessibility .....                                 | VII  |
| Contacting software support.....                    | VIII |
| Conventions used in this book.....                  | VIII |
| Typeface conventions .....                          | VIII |
| Operating system differences.....                   | VIII |
| Introduction .....                                  | 1    |
| Tivoli Security Compliance Manager Components ..... | 2    |
| Server component.....                               | 2    |
| Client component .....                              | 2    |
| Collectors .....                                    | 3    |
| Administration utilities.....                       | 4    |
| Policies.....                                       | 4    |
| Deployment: Preparation.....                        | 5    |
| Network Topology .....                              | 5    |
| System Information .....                            | 5    |
| Message Flow .....                                  | 5    |

|                                                                   |    |
|-------------------------------------------------------------------|----|
| Deployment: IBM Tivoli Security Compliance Manager Topology ..... | 6  |
| The Tivoli Security Compliance Manager Server.....                | 6  |
| The Database Server .....                                         | 6  |
| Tivoli Security Compliance Manager Proxy Relays .....             | 6  |
| Tivoli Security Compliance Manager Clients .....                  | 6  |
| Deployment: Process .....                                         | 7  |
| Proof of Concept .....                                            | 7  |
| Deployment: Capacity Planning.....                                | 9  |
| Capacity planning process.....                                    | 9  |
| Determining physical network connections.....                     | 10 |
| Identifying server transactions .....                             | 10 |
| Estimating transactions .....                                     | 10 |
| Defining server transaction throughput requirements .....         | 10 |
| Basing estimates on empirical data.....                           | 11 |
| Basing estimates on registered clients.....                       | 11 |
| Choosing hardware .....                                           | 11 |
| Memory and disk storage planning .....                            | 11 |
| Obtaining measurements and throughput rates.....                  | 12 |
| Scaling for hardware differences.....                             | 12 |
| Operating system differences.....                                 | 13 |
| Scaling for CPU utilizations less than 100%.....                  | 13 |
| Scaling for physical network differences .....                    | 13 |
| Capacity planning example #1 .....                                | 14 |
| Identifying server transactions .....                             | 14 |
| Defining server transaction throughput requirements .....         | 14 |
| Choosing hardware .....                                           | 14 |
| Obtaining measurements and throughput rates.....                  | 14 |

|                                                           |    |
|-----------------------------------------------------------|----|
| Capacity planning example #2 .....                        | 15 |
| Identifying server transactions .....                     | 15 |
| Defining server transaction throughput requirements ..... | 15 |
| Choosing hardware .....                                   | 15 |
| Obtaining measurements/throughput rates .....             | 15 |
| Message Throughput Calculation .....                      | 16 |
| Actual Deployment .....                                   | 17 |
| Tuning .....                                              | 18 |
| Suggested Configuration .....                             | 18 |
| Hardware Specifications .....                             | 18 |
| Tuning Considerations .....                               | 19 |
| Notices .....                                             | 29 |

## Preface

### Preface

This document provides information on IBM Tivoli Security Compliance Manager 5.1 architecture, recommended deployment procedures, and recommended hardware and tuning specifications for Security Compliance Manager components to be used for deployment planning. These recommendations are based on performance benchmark testing and represent the best available information at this time.

Performance measurements are derived in a test lab, based on specific system configurations. All performance numbers are to be used as a guideline for planning purposes only. Actual production numbers will vary based on uniqueness of the deployed environment.

Performance testing and enhancements are ongoing and further information will be published as it becomes available.

### Who should read this book

The target audience for this deployment guide includes:

- System administrators
- Installation planners responsible for planning the deployment of IBM Tivoli Security Compliance Manager

### Publications

Read the descriptions of the IBM Tivoli Security Compliance Manager library, the prerequisite publications, and the related publications to determine which publications you might find helpful. After you determine the publications you need, refer to the instructions for accessing publications online.

### IBM Tivoli Security Compliance Manager library

The publications in the IBM Tivoli Security Compliance Manager library are:

*IBM Tivoli Security Compliance Manager Installation Guide: All Components* (GC32-1592-00) Explains how to install and configure Tivoli Security Compliance Manager software.

*IBM Tivoli Security Compliance Manager Installation Guide: Client Component* (GC32-1593-00) Explains how to install and configure the Tivoli Security Compliance Manager client component software.

*IBM Tivoli Security Compliance Manager Administration Guide* (SC32-1594-00)  
Explains how to manage and configure Tivoli Security Compliance Manager services using the administration console.

*IBM Tivoli Security Compliance Manager Collector Development Guide* (SC32-1595-00) Explains how to design and implement custom Tivoli Security Compliance Manager collectors.

*IBM Tivoli Security Compliance Manager Warehouse Enablement Pack, Version 1.1 Implementation Guide for Tivoli Data Warehouse, Version 1.2* (SC32-1596-00) Explains how to integrate Tivoli Security Compliance Manager with Tivoli Data Warehouse.

*IBM Tivoli Security Compliance Manager Release Notes* (GI11-4695-00)  
Provides late-breaking information, such as software limitations, workarounds, and documentation updates.

## Related publications

The Tivoli Software Library provides a variety of Tivoli publications such as white papers, datasheets, demonstrations, redbooks, and announcement letters. The Tivoli Software Library is available on the Web at: <http://www.ibm.com/software/tivoli/library/>

The *Tivoli Software Glossary* includes definitions for many of the technical terms related to Tivoli software. The *Tivoli Software Glossary* is available, in English only, from the **Glossary** link on the left side of the Tivoli Software Library Web page <http://www.ibm.com/software/tivoli/library/>

## IBM DB2 Universal Database™

IBM DB2® Universal Database is required when using Tivoli Security Compliance Manager. Additional information about DB2 can be found at: <http://www.ibm.com/software/data/db2/>

## Accessing publications online

The publications for this product are available online in Portable Document Format (PDF) or Hypertext Markup Language (HTML) format, or both in the Tivoli software library: <http://www.ibm.com/software/tivoli/library> To locate product publications in the library, click the **Product manuals** link on the left side of the library page. Then, locate and click the name of the product on the Tivoli software information center page. Product publications include release notes, installation guides, user's guides, administrator's guides, and developer's references. **Note:** To ensure proper printing of PDF publications, select the **Fit to page** check box in the Adobe Acrobat Print window (which is available when you click **File • Print**).

## Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. You can use assistive technologies to hear and navigate the product documentation. You also can use the keyboard instead of the mouse to operate some features of the graphical user interface.

## Contacting software support

Before contacting IBM Tivoli Software Support with a problem, refer to the IBM Tivoli Software Support site by clicking the **Tivoli support** link at the following Web site:  
<http://www.ibm.com/software/support/>

If you need additional help, contact software support by using the methods described in the *IBM Software Support Guide* at the following Web site:  
<http://techsupport.services.ibm.com/guides/handbook.html> The guide provides the following information:

- Registration and eligibility requirements for receiving support
- Telephone numbers, depending on the country in which you are located
- A list of information you should gather before contacting customer support

## Conventions used in this book

This reference uses several conventions for special terms and actions and for operating system-dependent commands and paths.

## Typeface conventions

The following typeface conventions are used in this reference:

- o **Bold** Lowercase commands or mixed case commands that are difficult to distinguish from surrounding text, keywords, parameters, options, names of Java classes, and objects are in **bold**.
- o *Italic* Variables, titles of publications, and special words or phrases that are emphasized are in *italic*.
- o Monospace Code examples, command lines, screen output, file and directory names that are difficult to distinguish from surrounding text, system messages, text that the user must type, and values for arguments or command options are in monospace.

## Operating system differences

This book uses the UNIX convention for specifying environment variables and for directory notation. When using the Windows command line, replace *\$variable* with *%variable%* for environment variables and replace each forward slash (/) with a backslash (\) in directory paths. If you are using the bash shell on a Windows system, you can use the UNIX conventions.

## Introduction

IBM Tivoli Security Compliance Manager is a policy-driven data collection and security compliance product. You can define a consistent security policy and then regularly monitor compliance with that policy to detect vulnerabilities. Security policies can be created that monitor compliance with both internal security requirements as well as industry-standard or governmental security and privacy requirements. Fundamentally, Tivoli Security Compliance Manager performs two primary functions:

- Collects data from multiple computer systems and stores that data in a central repository.

- Provides reporting capabilities to show adherence to, or violation of defined security policies.

As a data collection service, Tivoli Security Compliance Manager can gather and store a wide variety of information from multiple computer systems. This information can include any data located on the system, such as operating system version information, software patch levels, registry information, file and directory information, application information, and other security-related data. As a monitoring and reporting service, Tivoli Security Compliance Manager can extract the data collected, analyze the data for vulnerabilities, and produce reports that can reveal adherence to internal and industry-standard security policies.

## Tivoli Security Compliance Manager Components

Tivoli Security Compliance Manager consists of the following components:

### Server component

The server is Java™ language-based software that centrally manages all data associated with Tivoli Security Compliance Manager. By default, the server runs as a daemon with scmsrver authority on UNIX® systems and as a service running as the local system account on Microsoft Windows® systems. The server manages:

- The registration of clients
- The assignment of clients to one or more client groups
- The installation, authorization, registration, deployment, and scheduling of collectors
- The creation, modification, and deployment of policies
- The administrative users and their roles
- The assignment of administrative users to one or more user groups
- The keystores, authorization keys, and their associated passwords
- The creation and scheduling of reports and snapshots
- The server itself

The data associated with the objects being managed by the server is stored in a DB2 database. For example, data collected by the collectors is sent from the clients to the server and stored in the database. A report being run from an administration console to validate that the proper level of encryption is in use requests the data from the server, which extracts the relevant data from the database. The server is the only Tivoli Security Compliance Manager component that directly accesses the database.

### Client component

The client is Java language-based software that runs on systems to be monitored for security compliance. For all operating system platforms other than HP-UX, the client component includes its own Java Virtual Machine (JVM), making the client a self-contained component. The client can be installed using the InstallShield MultiPlatform (ISMP) installation program or distributed to systems using software distribution tools, such as IBM Tivoli Configuration Manager.

By default, the client runs as a daemon with root authority on UNIX systems, or as a service running under the local system account on Microsoft Windows systems. The client provides the runtime environment for collectors deployed to the system and handles communication with the server.

The type of client determines how communications are initiated between the server and the client. There are two types of clients: a push client and a pull client. A push client can establish a Secure Sockets Layer (SSL) connection to the server and send data. A pull client must wait until the server establishes a persistent SSL connection with the client before data can be sent.

Defining a client as a push client, which is the default, permits communication with the server to be established by either the client or the server. In some network environments, however, inbound connections to the server are not permitted. In these cases, defining the client as a pull client forces the server to initiate the communication with the client. Pull clients are generally needed when the server is located behind a firewall.

After a connection between the client and the server has been made, either can send data to the other. Clients contact the server at periodic intervals called a *heartbeat*, which is every 10 minutes by default, to check for updates.

During this heartbeat, the client receives any new or updated collectors from the server, along with any new or updated collector schedules and parameters. The client component software itself can be sent by the server and the client updates itself and restarts. This client/server heartbeat can be initiated from the administration console using the Soft reset request function.

Data gathered by the collectors that have run on the client is queued for delivery to the server on a more frequent basis, which is every minute, by default. Each client is uniquely identified to the server using a client identification (CLI\_ID) number.

## Collectors

A *collector* is a Java language-based software module, packaged as a Java Archive (JAR) file, which collects specific information from a client system. A collector is designed to be short-lived and non-invasive. The collector collects data by:

- reading the content of one or more files on the client system
- running an operating system command or utility and examining the output
- running an executable program packaged as part of the collector JAR file and examining the output
- reading information from the registry on Microsoft Windows systems

The first time a collector is deployed to a client, the JAR file for the collector is sent from the server to the client, along with the collector schedule and any associated parameters. Multiple instances of a collector can be deployed to a client. Subsequent instances of the collector share the same JAR file, but run on their own schedule and with their own parameters. Each instance of a collector is uniquely identified by a collector instance number (INSTANCE\_ID).

Data collected by each collector instance is queued by the client and delivered to the server on a periodic basis, usually every minute. The server stores the information received from the client into one or more tables in the database. The data in the database table is uniquely identified by the client identification number (CLI\_ID) and the collector instance number (INSTANCE\_ID).

When a collector instance is removed from a client, any data associated with that instance of the collector is removed from the database tables by the server.

## Administration utilities

The administration console is used to perform nearly all of the administration tasks in Tivoli Security Compliance Manager. The administration console runs only on Microsoft Windows systems. The administration commands, which run on both UNIX and Windows systems, are provided to permit a subset of the administration functions to be performed from a command shell, batch file, or script. The administration utilities communicate with the server through an SSL connection established using the Java Remote Method Invocation (RMI) technology.

A default administration user called *admin* is created during the installation of the server. This default user cannot be removed and has the necessary permissions to perform any task using either the administration console or the administration commands.

By default, authentication of administrative users is handled by the Tivoli Security Compliance Manager server. User information is stored in the database with the password being stored in MD5 message-digest format. The server does not enforce any password rules or perform any password strength testing and no mechanism exists to recover a forgotten password. To provide additional security, or to integrate Tivoli Security Compliance Manager into your existing authentication mechanism, you can use the information in the *Release Notes* to develop and install a Java Authentication and Authorization Service (JAAS) plug-in that handles the authentication and authorization of users.

## Policies

A *policy* is a package consisting of one or more collectors and one or more definitions, or conditions, to be verified. These definitions or conditions are expressed as compliance queries. A compliance query is a Structured Query Language (SQL) statement that extracts data from one or more database tables, evaluates the data extracted, and then returns a list of the clients that are in violation of the condition being tested. A policy consists of a number of compliance queries that together show whether client systems are in compliance with a specific security directive, standard, or guideline. The collectors included in a policy, along with their schedules and parameters, are deployed to clients to ensure that the necessary data is collected and stored in the database so that a meaningful evaluation of the compliance state of the systems can be made. Existing policies can be modified and new policies can be created using the administration console.

## Deployment: Preparation

The first step in the deployment process is getting a thorough understanding of the overall network environment in which Tivoli Security Compliance Manager will be deployed.

Every network is unique. Some factors that make a network unique are the types of systems in the network, the overall structure of the network, and the types of security-related products (such as firewalls) deployed within the network.

Such properties must be taken into account when deploying Tivoli Security Compliance Manager, in order to achieve a robust, efficient, and effective setup. A further discussion of these properties follows.

### Network Topology

In regards to topology, here are some questions you might ask:

- How many geographic locations are systems deployed in?

- How are these locations connected? Is there a firewall between them and how restrictive is this firewall?

- Where are the high speed and low speed links?

This basic information will tell you some important information about how Tivoli Security Compliance Manager should be deployed. In places where there is a firewall, you will need to deploy Tivoli Security Compliance Manager proxy relays in order to send and receive data through an authorized port.

### System Information

After you've outlined your network's topology, the next step is to identify the systems that exist within that network. This scenario is primarily concerned with those systems that will fall within the scope of security monitoring. For proper analysis, the network systems should be broken up into logical groups that will map later to client groups within IBM Tivoli Security Compliance Manager: For instance, a company might have 200 file servers on Windows 2000 systems, 100 Web servers on Sun Solaris operating environment systems, and so on.

You should identify:

- Categories of systems in the network (Web servers, file servers, and so on)

- Number of systems running in each of these roles

- Hardware and software platforms for systems in each role

The more exact numbers you can get during this phase of investigation, the more accurate your planning will be. This information will also be useful when creating client groups within IBM Tivoli Security Compliance Manager.

### Message Flow

The next step is to figure out what happens when data starts to move. Where do the messages that you're concerned about come in to the network? How do they move around the network? How frequent are they?

## Deployment: IBM Tivoli Security Compliance Manager Topology

It is now time to map out what the actual deployment will look like.

The suggested topology has the following characteristics:

### The Tivoli Security Compliance Manager Server

The IBM Tivoli Security Compliance Manager server should be installed on a system with a high processor speed and ample disk space. Refer to the section entitled [Hardware Specifications and Recommendations](#) for more details.

The system that contains the server should be solely dedicated to that task. This configuration allows the system to be tuned and optimized for running the IBM Tivoli Security Compliance Manager. This configuration also keeps the server from having to compete with other applications for system resources.

### The Database Server

The database server serves as the repository for all IBM Tivoli Security Compliance Manager data. The database server can be the same system as the IBM Tivoli Security Compliance Manager server, however, for better performance; the database server can be installed on a separate system. For even greater performance the database server can be installed on a multi-processor machine.

### Tivoli Security Compliance Manager Proxy Relays

The Tivoli Security Compliance Manager proxy relay provides a solution for the scenario of a server separated from destination clients by one or more intermediary networks because of firewall policies or address space concerns. The goal of the Tivoli Security Compliance Manager proxy relay is to permit the server to successfully connect to and communicate with each destination client system.

The Tivoli Security Compliance Manager proxy relay is a small-scale solution that is not intended to replace the need for a full-scale proxy server. In general, the number of clients connected through a proxy relay should be kept to a minimum due to the resources used for each connection.

If a large number of systems are accessed through a proxy relay, the proxy relay system, depending on operating system, might need to be configured on the system level to allow for a large number of open sockets. In respect to hardware, the proxy relay system should fall to the upper end of the recommended hardware for clients.

### Tivoli Security Compliance Manager Clients

The IBM Tivoli Security Compliance Manager client is designed to be lightweight and thus can be placed on any system that meets the minimum hardware specifications.

## Deployment: Process

No matter how much planning is done, you won't have a crystal clear understanding of how IBM Tivoli Security Compliance Manager will operate in your environment until you install it and experience it first hand. It is recommended that you take a three-phase approach to deploying IBM Tivoli Security Compliance Manager: proof of concept, test environment, and production rollout.

### Proof of Concept

Despite the fact that a “real world” IBM Tivoli Security Compliance Manager deployment can span numerous systems in a wide variety of roles and responsibilities, it is actually possible to install an entire IBM Tivoli Security Compliance Manager demo on one physical system.

On this one system, install the following components in the following order:

**DB Server:** A database that is used to store the IBM Tivoli Security Compliance Manager data.

**SCM Server:** IBM Tivoli Security Compliance Manager server.

**SCM Client:** **Client software that** executes collectors to retrieve specific information from the client machine.

**SCM Proxy relay:** A special collector used to allow the server to communicate with multiple clients that are located behind a firewall.

**Admin Utilities:** A command line user interface to the IBM Tivoli Security Compliance Manager server. On Microsoft Windows systems, these utilities also provide a graphical user interface, called the administration console.

A deployment like the one described serves several purposes. First, it allows you to become familiar with the installation of the product. Second, it demonstrates how components interact with each other. Mainly, however, it gives you experience using the product.



## Deployment: Capacity Planning

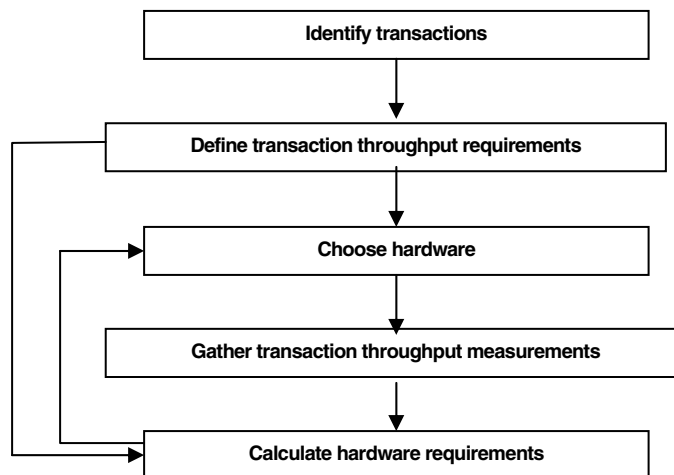
Capacity planning is the process of assessing system requirements and ensuring that your IBM Tivoli Security Compliance Manager environment has adequate resources to service requests at an acceptable rate. This section is intended to cover capacity planning of IBM Tivoli Security Compliance Manager. Note that this section does not contain specific measurements. For performance measurements, refer to IBM Tivoli Security Compliance Manager performance reports or supplement existing measurements with those taken in your own lab.

This section contains the following main sections:

- Capacity planning process
- Identifying server transactions
- Defining server transaction throughput requirements
- Choosing hardware
- Obtaining measurements and throughput rates
- Capacity planning examples

### Capacity planning process

The *capacity planning process* begins with an idea of the type of workload that the communication system is required to handle. The next step in the capacity planning process is to identify the types of transactions that the server is to handle and to map the workload-to-server throughput requirements. Then choose hardware and gather measurements. In turn, these requirements and measurements are input to a calculation that determines the portion of a machine needed for the server. The following flowchart illustrates the capacity planning process:



The accuracy of the planning process is directly related to the accuracy of the throughput requirements and estimates that are inputs to the process. And as with any planning exercise, the result is an estimate and a margin of error is assumed.

The margin of error varies depending on many factors, including experience level, confidence in measurements, and how closely the measurements match the desired transactions. Note that this section makes no claims as to the margin of error that can be assumed with the use of the described methods. It is up to you, as planner, to decide the margin of error.

### **Determining physical network connections**

An item to consider regarding network topology is the physical network connection choices. You must determine how many physical networks and line speeds are possible or desired.

Network bandwidth might be one reason to require multiple physical networks. If the network is the bottleneck, multiple physical networks can be one way to solve the problem. Higher line speeds are another option.

### **Identifying server transactions**

The next step is to determine the type of workload that the server is expected to handle. Express the workload as server level transactions. This step in the capacity planning process seeks to identify all the transactions that the server is expected to handle. In this step, it is not necessary to determine the frequency of each transaction. The frequency of each transaction is determined in the next step in the process—"Defining server transaction throughput requirements".

### **Estimating transactions**

The primary transaction in an IBM Tivoli Security Compliance Manager environment is a collector message from a client. Other transactions include the client update request and the user administrative commands, such as list clients.

However, the frequency of the collector messages generates the primary load on the IBM Tivoli Security Compliance Manager server. Over any appreciable time interval, collector messages will exceed the traffic of all other transactions by orders of magnitude and thus, other transactions can safely be ignored in our calculations of hardware requirements. Having said this, there are sure to be cases where snapshots and other administrative tasks are done with such frequency that they need to be factored into the workload equations.

### **Defining server transaction throughput requirements**

The next step in the capacity planning process is to define the server transaction throughput requirements for each transaction on the server. This is when the frequency of each server transaction is defined. To start, define the server level transaction throughputs.

Deciding upon the frequency of transactions, or *throughput*, in the communication system is perhaps the hardest part of the capacity planning process. It typically is just an estimate and, as such, can be the biggest source of error when calculating the hardware required. To reduce the margin of error, base the estimate on empirical data from

production environments. This is not always possible, so other methods, or a combination of methods, for estimating transaction throughputs are necessary.

### **Basing estimates on empirical data**

There are a number of ways to gather empirical data. One way is to take IP packet traces and extract the types of transactions and their frequencies over time. After ensuring that server logging is enabled, you can examine the types and frequencies of transactions. For example, you can determine the frequency client heartbeats and client update requests from such traces. Note that when using such measurements, you must take into account any differences between the test environment being measured and the planned production environment.

### **Basing estimates on registered clients**

Another method for estimating transaction throughputs, or requirements, is to base estimates on the number of clients registered. The idea is that some percentage of these clients will use the system in any given time period. Along with this, there is an idea of what an average client does in terms of putting load on the system. For example, the percentage of collectors executed in any given time period is the message rate. The message rate multiplied by the load applied by an average message gives the throughput for the average workload. Note that this method tends to lead to greater margins of error. Following is an example of this method.

Assume that there are 10000 registered clients with 20 collectors each and that approximately 50% of the collectors run each day. The resulting message rate is 1.16 messages per second as calculated in the following formula:

$$10,000 \text{ clients} * 20 \text{ collectors} * 0.50 \text{ percent} / 24 \text{ hours in a day} / 60 \text{ minutes in an hour} / 60 \text{ seconds in a minute} = 1.16 \text{ msgs/sec}$$

### **Choosing hardware**

The choice of hardware might or might not be flexible. You might have a requirement for a specific type of machine or network or you might need to make the most cost-effective choice from a range of hardware. In either case, you must choose hardware. To evaluate the cost of different hardware choices, the process can be done iteratively with a different hardware specification each time.

From a machine perspective, the choices are defined in terms of manufacturer, model number, CPU speed, and number of processors. Often, the choice for hardware forces a choice of operating system. For workload capacity planning purposes, the server is assumed to have enough RAM and disk space to run efficiently. From a physical network perspective, the choices are defined in terms of type of network, such as Ethernet or token ring, fiber optics, line speed, and network adapter card manufacturer.

The following sections describe capacity planning issues related to hardware.

### **Memory and disk storage planning**

Related to capacity planning is planning for memory and disk storage requirements. For the components in the system, consult the installation guide for information on the product's base memory and storage requirements.

## Obtaining measurements and throughput rates

In this step of the capacity planning process, you must match all identified server transactions to real measurements on hardware comparable to that chosen. For capacity planning purposes, you need to gather measurements of maximum throughput rate for each transaction type. Maximum throughput rates are typically gathered by ramping up the load on the server until throughput levels off and response times become too long. Server throughput measurements can come from several sources, including the provider of the server software, performance reports from third party performance testing companies, and testing in your own lab.

Careful reading of any measurement report is highly recommended to better understand what is being measured. Try to find measurements that most closely match the environment in which the communication system is to be deployed.

One item to observe is the hardware used for the measurement. Attempt to find measurements that most closely match the hardware that you plan to deploy. Then try locating reports with similar architectures, for example, view information about IBM RS/6000 systems if you plan to deploy RS/6000 systems. Also attempt to match similar CPU speeds, operating systems, and operating system levels.

If measurement reports are not available or not complete, you will need to estimate from existing measurements, or take new ones. The following sections explain some common techniques for estimating when transaction types and environments do not match available measurements.

## Scaling for hardware differences

One method for bridging hardware differences is to use published benchmarks, available from the following Web site:

<http://www.spec.org>

The Standard Performance Evaluation Corporation (SPEC) provides hardware manufacturers a way to publish the results of certain performance benchmarks. Because communication programs tend to act like integer arithmetic programs, the benchmark of interest is SPECint. To find SPECint results, select **SPEC CPU2000** from the Web site. Then select **Submitted Results** and **SPECint2000**. Look for machines of interest in any of the submitted results. If a benchmark result lists both machines, use the results as a ratio to scale between the two machines.

The formula is as follows:

$$\text{<performance of machine one>} = \text{<performance of machine two>} \\ * \text{< SPECint result for machine one>} / \text{< SPECint result for machine two>}$$

If there are no benchmark results listing both machines, it might be possible to use an intermediate machine as a bridge. The intermediate machine must be listed in the benchmark results for both of the machines of interest. In this case, you can use ratios to scale from one machine to the intermediate machine and then from the intermediate machine to the second machine.

If a machine does not appear in any benchmark result, it might be similar in architecture to a machine that is listed in the benchmark results. For similar architectures, CPU speed or MHz ratings are good factors to use for scaling. In this case, use the ratio of the CPU speeds to scale from one machine to another.

## **Operating system differences**

Although the SPECint benchmark results display the relative difference in the performance of two machines, the benchmark results tend to be less accurate at predicting communication software performance when operating systems differ. This decrease in accuracy is because the integer arithmetic benchmarks do not consider certain operating system-specific differences, such as SMP scaling, I/O processing, and resource usage. You should use a larger margin of error when the operating systems differ between machines being scaled, especially if the transaction of interest includes processing that might be operating-system specific.

## **Scaling for CPU utilizations less than 100%**

When CPU utilizations are provided, less than maximum throughput might be indicated by less than 100% CPU utilization on the system being measured. It is typical and good (what does good mean here??) when the server runs at less than 100% CPU utilization. It is possible to estimate maximum throughput from measurements where less than maximum throughput has been achieved, but it requires knowledge of the CPU utilization. It also results in a larger margin of error, because software systems do not always behave well when they reach or go beyond maximum throughput. Following is the formula for estimating maximum throughput from measured throughput at less than maximum:

Maximum throughput = measured throughput / CPU utilization

For example, if throughput is measured at 50% CPU utilization, the estimated maximum throughput is twice the measured rate, because only half (50%) the machine is utilized.

## **Scaling for physical network differences**

When the physical network used in a measurement report differs from the physical network being deployed, the difference is usually not a concern in terms of the throughputs that the servers are capable of achieving. Line speed differences do not affect the server's ability to do work. It can be a concern if the network adapter card makes heavy use of the CPU. However, this is typically not the case. To guard against this, consider using high-quality and highlyrated network adapter cards.

Physical network differences do affect the network's capacity or throughput. For capacity planning of the physical network, you should obtain measurement reports of the physical network throughput. Alternatively, the physical network's throughput can be estimated as some percentage of its theoretical rate.

For example, a 100 Mbps Ethernet network can easily achieve 7 MB per second rates with file transfer protocol (ftp), which is 70% of the theoretical rate. The physical network capacity tends to be less of a concern because most physical networks achieve high rates of transfer. Even so, be sure to consider the physical network capacity.

## Capacity planning example #1

The following example steps you through the capacity planning process for a fictitious company.

### Identifying server transactions

Assume the following observations:

- IBM Tivoli Security Compliance Manager server and DB server are on the same machine
- There are 10000 registered and active clients
- Each client contains 20 scheduled collectors
- The average collector message size is 10 KB in size.

### Defining server transaction throughput requirements

Assume that policies have been pushed to the clients and the schedules have been set so that the collectors run at a frequency of once every 12 hours.

The IBM Tivoli Security Compliance Manager server throughput requirement is calculated as follows:

$$(10000 \text{ clients} * 20 \text{ collectors} / 12 \text{ hours} / 60 \text{ minutes} / 60 \text{ sec}) = 4.6 \text{ messages/sec}$$

The Intranet network throughput requirement is calculated as follows:

$$4.6 \text{ messages/sec} * 10\text{KB/msg} = 40.6 \text{ KB/sec}$$

### Choosing hardware

Assume that the choice for all servers is the Sun Enterprise V120. The V120 is a uniprocessor 650 MHz UltraSPARC-II processor machine. Also assume that the choice for all networks is 100 Mbps Ethernet LANs.

### Obtaining measurements and throughput rates

Assume that you obtained the following fictitious measurements for the Sun V120:

IBM Tivoli Security Compliance Manager 10KB collector message throughput  
16 messages/sec

Intranet throughput  
7+ MB/sec

If you compare the requirements and measurements you can see that matching measurements exist for both the requirements.

## Capacity planning example #2

While the case of needing to update the clients via the client update feature will be infrequent, it is wise to verify that the existing or predicted hardware is capable of handling such a case.

The following example steps you through the capacity planning process for a fictitious company applying a client update.

### Identifying server transactions

Assume the following observations:

- IBM Tivoli Security Compliance Manager server and DB server are on the same machine

- There are 10000 registered and active clients

- Each client contains 20 scheduled collectors

- The average collector message size is 10 KB in size.

- The client update JAR file is 7MB in size

### Defining server transaction throughput requirements

Assume that policies have been pushed to the clients and the schedules have been set such that the collectors run at a frequency of once every 12 hours.

The IBM Tivoli Security Compliance Manager server throughput requirement is calculated as follows:

$(10000 \text{ clients} * 20 \text{ collectors} / 12 \text{ hours} / 60 \text{ minutes} / 60 \text{ sec}) = 4.6 \text{ messages/sec}$

$(10000 \text{ clients} / 4 \text{ hours} / 60 \text{ minutes} / 60 \text{ sec}) = 0.7 \text{ updates/sec}$

The Intranet network throughput requirement is calculated as follows:

$4.6 \text{ messages/sec} * 10\text{KB/msg} = 40.6\text{KB/sec}$

$(10000 \text{ clients} * 7,000,000 \text{ bytes for client JAR} / 4 \text{ hours} / 60 \text{ minutes} / 60 \text{ sec}) = 4.86 \text{ MB/sec}$

### Choosing hardware

Assume that the choice for all servers is the Sun Enterprise V120. The V120 is a uniprocessor 650 MHz UltraSPARC-II processor machine. Also assume that the choice for all networks is 100 Mbps Ethernet LANs.

### Obtaining measurements/throughput rates

Assume that you obtained the following fictitious measurements for the Sun V120

IBM Tivoli Security Compliance Manager 7MB client update throughput (concurrent with recommended maximum message throughput)  
0.5 updates/sec

IBM Tivoli Security Compliance Manager 10KB collector message throughput  
16 messages/sec

Intranet throughput  
7+ MB/sec

If you compare the requirements and measurements you can see that matching measurements exist for all the requirements. While the network can support the 4.86MB/sec requirement, the load on the IBM Tivoli Security Compliance Manager server will exceed the measured capacity. Therefore a faster machine might be needed. An alternative would be to extend the client update interval from the default of 4 hours to 8 hours, halving the server load and bandwidth required.

### Message Throughput Calculation

The message throughput of a deployment can be calculated using the following formula:

$$N * ( X/(604800 \text{ seconds/week}) + Y/(86400 \text{ seconds/day}) + Z/(3600 \text{ seconds/hour}) ) = M$$

Where:

N – the number of registered clients

X – the number of collectors on one client that are scheduled to run on a weekly basis

Y – the number of collectors on one client that are scheduled to run on a daily basis

Z – the number of collectors on one client that are scheduled to run on an hourly basis

| Number of Clients (N) | Weekly Collectors/Client (X) | Daily Collectors/Client (Y) | Hourly Collectors/Client (Z) | Messages/Sec Throughput (M) |
|-----------------------|------------------------------|-----------------------------|------------------------------|-----------------------------|
| 1000                  | 0                            | 24/86400                    | 0                            | 0.28                        |
| 10000                 | 24/604800                    | 0                           | 0                            | 0.40                        |
| 10000                 | 0                            | 24/86400                    | 0                            | 2.78                        |
| 10000                 | 0                            | 20/86400                    | 4/3600                       | 13.43                       |
| 10000                 | 0                            | 12/86400                    | 12/3600                      | 34.72                       |
| 10000                 | 10/604800                    | 10/86400                    | 4/3600                       | 12.43                       |

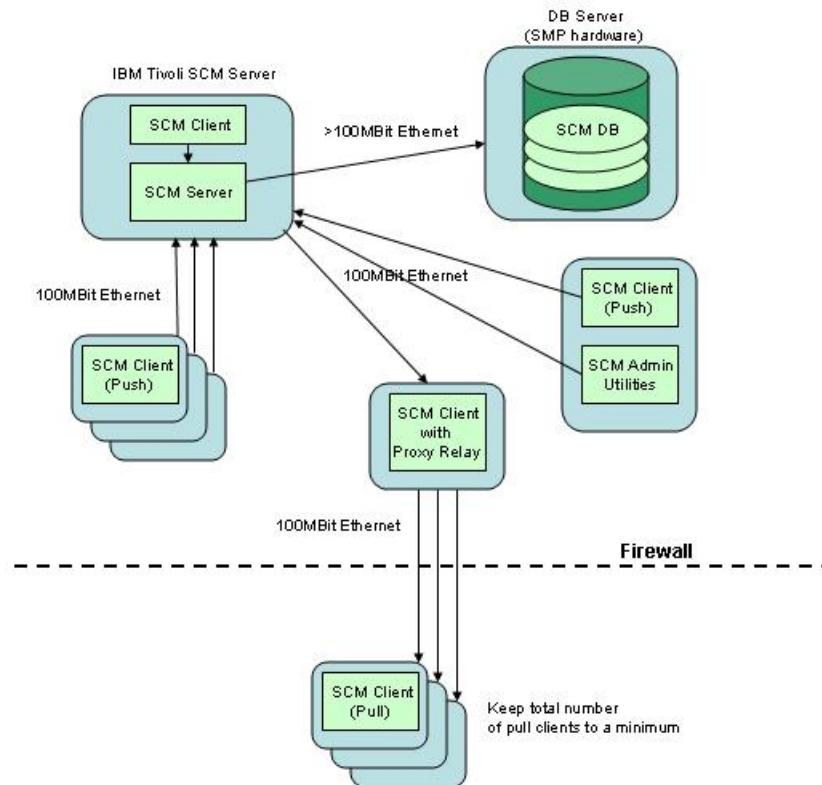
## Actual Deployment

After you've gathered all the necessary information about your network as a whole, and you've gone through test installations of IBM Tivoli Security Compliance Manager, you're ready to do the initial rollout.

IBM Tivoli Security Compliance Manager components, regardless of platform, are installed using the same GUI. This consistency allows for a fairly standard installation process.

## Tuning

### Suggested Configuration



### Hardware Specifications

| IBM Tivoli Security Compliance Manager Server |                                                |                                             |                                                             |
|-----------------------------------------------|------------------------------------------------|---------------------------------------------|-------------------------------------------------------------|
| Operating System                              | Minimum Specifications                         | Suggested Specifications                    | Maximum Message Throughput (12KB messages @ 75%server load) |
| AIX                                           | pSeries 615<br>1.20Ghz 1GB<br>RAM 36GB<br>Disk | pSeries 615<br>1.45Ghz 2GB<br>RAM 36GB Disk | 8 msg/sec                                                   |

|         |                                          |                                       |            |
|---------|------------------------------------------|---------------------------------------|------------|
| Solaris | SUN V210<br>1Ghz 1GB<br>RAM 36GB<br>Disk | SUN V210<br>1Ghz 2GB RAM<br>36GB Disk | 7 msg/sec  |
| Windows | Intel 2.2Ghz<br>1GB RAM<br>36GB Disk     | Intel 2.8Ghz<br>2GB Ram<br>36GB Disk  | 24 msg/sec |
| Linux   | Intel 2.2Ghz<br>1GB RAM<br>36GB Disk     | Intel 2.8Ghz<br>2GB Ram<br>36GB Disk  | 24 msg/sec |

| <b>IBM Tivoli Security Compliance Manager Proxy Relay</b> |                                    |                                  |                           |
|-----------------------------------------------------------|------------------------------------|----------------------------------|---------------------------|
| Operating System                                          | Minimum Specifications             | Suggested Specifications         | Maximum Number of clients |
| AIX                                                       | pSeries 610<br>450MHz<br>512MB RAM | pSeries 610<br>450MHz 1GB<br>RAM | 1000                      |
| Solaris                                                   | SUN V120<br>650MHz<br>512MB RAM    | SUN V120<br>650MHz 1GB<br>RAM    | 1000                      |
| Windows                                                   | Intel 2.2Ghz<br>512MB RAM          | Intel 2.2Ghz<br>1GB Ram          | 1000                      |
| Linux                                                     | Intel 2.2Ghz<br>512MB RAM          | Intel 2.2Ghz<br>1GB Ram          | 1000                      |

## Tuning Considerations

This section addresses some configuration issues and features that are essential to increasing the performance of the IBM Tivoli Security Compliance Manager deployment.

### Randomization Usage

The IBM Tivoli Security Compliance Manager contains a facility that allows collectors to be scheduled to run at random times within specified intervals, such as one day a week. This feature is immensely important in controlling server load. This randomization (what does this mean in this context??) allows collector messages to be spread out over an interval rather than flooding the server at one specified time. Therefore, consider using this feature extensively when designing policies or scheduling collectors.

## Client Tuning

An update feature permits the client to request from the server any new client program files that might exist. The default interval is 4 hours. This means that all clients will request an update within 4 hours after a new client update JAR becomes available. If there are several thousand clients in the SCM environment, the resulting traffic can be significant. Therefore, you should consider setting this value to the maximum acceptable interval.

```
[client updates]
update.enabled=true
check.interval=14400
```

By default, the client “pings” the server every 10 minutes to inform the server of the client’s activity. This request itself represents a very small network footprint. However, this interval also indirectly controls the flow of policies and collectors to the clients. When a policy is distributed to a client group, the client is not informed of the change until the server is contacted. The server informs the client of the new policies and collectors within its response to these “pings”. Therefore, by default, no matter how many clients exist in the system, they will all attempt to synchronize themselves by downloading the required policies and collectors within a 10-minute period. As the number of clients increase, this traffic will increase. As mentioned earlier, one method of throttling this traffic is to deploy policies in stages; another is to increase the “ping” interval from the default of 10 minutes to 30 minutes (1800 seconds). Note that if the ping interval on the client is modified, the corresponding inactivity timeout parameter on the server (*jac.server.no\_activity\_timeout*) must also be modified to reflect the change.

For example:

```
[scheduler]
ping.interval=1800
```

## Proxy Relay Tuning

The SCM proxy relay consists of an extensively threaded architecture. Each Java thread is allocated a specific amount of memory for local variables. By default, this amount is 256K in Java 1.3.1. Consider using a minimum value in order to maximize the potential number of threads. A value of 32K is suggested. Also consider increasing the maximum and initial Java heap settings on the client be increased from the default 64m to 256m. These values can be set in the *client.pref* file on the proxy relay system:

```
[jvm]
jvm.heap.max=256m
jvm.thread.stack.size=32k
jvm.heap.init=256m
```

## Server JVM Tuning

Both the server and client components allow specific Java Virtual Machine (JVM) parameters to be specified in the configuration files. These parameters give you control over the amount of memory that the JVM makes available to the components. Of particular importance is the maximum heap size parameter, *jvm.heap.max*. The default setting for this parameter on the server is 256m. The default server setting will be adequate for most installations, however, if memory allows, this value can be doubled for better performance. This performance boost occurs due to the reduced amount of garbage collection within the JVM. Also the *jvm.heap.init* parameter should be set to the same value as the *jvm.heap.max* parameter to short-circuit the heap expansion cycles. This setting sacrifices initial startup time in favor of runtime performance. If the deployment of a large number of pull clients (more than 1000) is planned, it is required that the thread stack size parameter be set to a lower value. By default this value is 256k, however, a suggested value is 128k.

For the server these settings are placed in the server.ini file:

```
# JVM initial heap size
jvm.heap.init=512m
# JVM maximum heap size
jvm.heap.max=512m
#JVM thread stack size
jvm.thread.stack.size=128k
```

For servers with more than 2GB of memory consider the following settings are recommended:

```
# JVM initial heap size
jvm.heap.init=768m
# JVM maximum heap size
jvm.heap.max=768m
#JVM thread stack size
jvm.thread.stack.size=128k
```

## Server DB Connection Tuning

By default the server is set to insert messages into the database on a row-by-row basis. Optionally the server can instead be configured to have messages inserted into the database in whole. This configuration is called batching and can increase the efficiency of the server-to-database connection. This mode is particularly useful when large messages are being processed as the message processor does not have to wait until each row is inserted. This parameter is located in the server.ini:

```
db.enable.batch=true
```

## AIX Platform Specific Server Tuning

For servers with a large number of pull clients (more than 1000) consider increasing the default maximum number of handles (this includes network connections) from the default 2000 to a higher number such as 20000. This parameter is found in the */etc/security/limits* file and is titled *nofile*. This value can be changed globally by setting it in the *default* stanza or a separate stanza can be created for the *scmsrver* user.

```
default:
    fsize = 2097151
    core = 2097151
    cpu = -1
    data = 262144
    rss = 65536
    stack = 65536
    nofiles = 20000
```

```
scmsrver:
    nofiles = 20000
```

## Database Server Tuning

On some operating systems there is a provision for limiting the maximum size of files. By default the IBM Security Compliance Manager database is configured with System Managed Storage (SMS) table spaces. These table spaces are subject to the limitations of the file system. Collectors that return large amounts of data (such as directory listings) might cause the SMS containers to approach the default file size limit of the operating system. On AIX the parameter that controls the maximum file size is the *fsize* parameter located in the */etc/security/limits* file. This parameter is set to 1GB by default (2097151 \* 512 byte blocks = 1GB), however, you should consider setting this value to -1, which is interpreted as unlimited.

```
default:
    fsize = 2097151
    core = 2097151
    cpu = -1
    data = 262144
    rss = 65536
    stack = 65536
```

```
nofiles = 20000
```

```
db2inst1:
```

```
fsiz = -1
```

## Database Manager Tuning

The following are some suggested tuning actions that can be applied to DB2 in order to improve the overall performance of the database.

Increasing the size of the buffer pools – A buffer pool is the area of memory where database pages are loaded from disk. The buffer pool is the key influencer of overall database performance because data can be accessed faster from memory than from disk. The buffer pool used by the IBM Security Compliance Manager can be enlarged by increasing the number of buffer pool pages. By default this buffer pool consist of 1000 pages of 32768 bytes each. Depending on memory available, the number of pages can be increased substantially. Warning: Do not increase the number of pages to the point where there is not enough memory to contain them all. (Example for a system with at least 2Gb memory: alter bufferpool JAC\_BP size 16384).

Separation of database log devices and data devices - Consider using the fastest disks in the system for the log devices. (Example: update db cfg for JAC using NEWLOGPATH=/logdev) (Is “update db cfg” a command?)

Increase the maximum size of the log files - Increasing the log file size reduces some of the administration overhead associated with maintaining log files. (Example: update db cfg for JAC using LOGFILSZ=4096) (Is “update db cfg” a command?)

Clear snapshots regularly.

## Database SQL Tuning

The IBM Tivoli Security Compliance Manager is foundationally based on the database and therefore efficiency of SQL statements is key not only to the operation of the IBM Tivoli Security Compliance Manager components but also to user-defined queries such as those used in reports and policies and accessing tables for in-house collectors. The following method can be used to pinpoint suboptimal SQL execution and also to optimize the efficiency of the execution. The following section focuses on the IBM DB2 database software. A database administrator (DBA) should perform these procedures.

The IBM DB2 program includes provisions that permit the database administrator to enable monitors within the database program that keep track of the amount of time and work that is required for SQL statements to execute. You can view the current settings with the *get monitor switches* command.

```
#db2 get monitor switches
```

#### Monitor Recording Switches

```
Switch list for db partition number 0
Buffer Pool Activity Information (BUFFERPOOL) = OFF
Lock Information (LOCK) = OFF
Sorting Information (SORT) = OFF
SQL Statement Information (STATEMENT) = OFF
Table Activity Information (TABLE) = OFF
Take Timestamp Information (TIMESTAMP) = ON
Unit of Work Information (UOW) = OFF
```

By default, all of the monitor switches, except for the timestamp monitor, are disabled. To enable the monitor switches, use the *update monitor switches* command. Normally, it is safe to enable all the switches as shown in this example. However, it is mainly the *statements* monitor that you are interested in.

```
#db2 update monitor switches using bufferpool on
#db2 update monitor switches using lock on
#db2 update monitor switches using sort on
#db2 update monitor switches using statement on
#db2 update monitor switches using table on
```

```
#db2 get monitor switches
```

#### Monitor Recording Switches

```
Switch list for db partition number 0
Buffer Pool Activity Information (BUFFERPOOL) = ON
Lock Information (LOCK) = ON
Sorting Information (SORT) = ON
SQL Statement Information (STATEMENT) = ON
Table Activity Information (TABLE) = ON
Take Timestamp Information (TIMESTAMP) = ON
Unit of Work Information (UOW) = OFF
```

After you have enabled the monitor switches, you can perform operations through the IBM Tivoli Security Compliance Manager that cause database activity. After you have completed these operations, take a snapshot of the monitors using the *get snapshot* command.

```
#db2 get snapshot for all on JAC > snapshot.txt
#cat snapshot.txt
```

```
...
Number of executions = 1
Number of compilations = 1
Worst preparation time (ms) = 4
```

```

Best preparation time (ms)           = 4
Internal rows deleted                = 0
Internal rows inserted               = 0
Rows read                           = 2095777
Internal rows updated                = 0
Rows written                         = 0
Statement sorts                     = 0
Total execution time (sec.ms)        = 64.718209
Total user cpu time (sec.ms)         = 7.520000
Total system cpu time (sec.ms)       = 0.350000
Statement text                       = delete from
JAC_IDATA.CUSTOM_COLLECTOR_V5 where CLI_ID=1636 and
INSTANCE_ID=37
...

```

The snapshot will display information about each query executed such as *rows read* and *total execution time*. You can see in this example that this particular delete query takes over a minute to execute. Using the IBM DB2 *explain* tool you can discover how DB2 is executing the query.

```
#dynexpln -g
```

Optimizer Plan:

```

                                DELETE
                                ( 2 )
                                /-----\
                                |         |
TBSCAN                         Table:
( 3 )                         JAC_IDATA
|                             CUSTOM_COLLECTOR_V5
Table:
JAC_IDATA
CUSTOM_COLLECTOR_V5

```

The explain tool output tells you that the database is using a *table scan* to locate prospective entries. Table scans are notoriously slow on larger tables ( such as those over a million rows) and are usually replaced automatically with indexes. In some instances the IBM DB2 optimizer will automatically generate indexes for large tables, in some instances it will not, and therefore you might want to manually create the indexes if the query is to be executed frequently.

In this particular instance you can create the index on the *cli\_id* and *instance\_id* columns for the JAC\_IDATA.CUSTOM\_COLLECTOR\_V5 table using the *create index* command:

```
#db2 create index CUSTOM_COLLECTOR_INDEX on
JAC_IDATA.CUSTOM_COLLECTOR_V5 (cli_id,instance_id)
```

To reset the database monitors, use the *reset monitor* command. Next you should repeat the operations that you performed previously and then take another snapshot and use the *explain* tool to verify the predicted optimization.

```

Number of executions                = 1
Number of compilations              = 1
Worst preparation time (ms)        = 4
Best preparation time (ms)         = 4
Internal rows deleted              = 0
Internal rows inserted             = 0
Rows read                          = 0
Internal rows updated              = 0
Rows written                       = 0
Statement sorts                    = 0
Total execution time (sec.ms)      = 0.005257
Total user cpu time (sec.ms)       = 0.010000
Total system cpu time (sec.ms)     = 0.000000
Statement text                     = delete from
JAC_IDATA.CUSTOM_COLLECTOR_V5 where CLI_ID=1636 and
INSTANCE_ID=37
...

```

You can see in this example that the rows read decreased from millions to zero because of the use of the index. The corresponding decrease in time of execution was from 64 seconds to a few milliseconds. You can view the SQL execution path with the *explain* tool.

```
#dynexpln -g
```

Optimizer Plan:

```

                                DELETE
                                ( 2)
                                /-----\
                                \
                                IXSCAN
                                ( 3)
                                /      \
                                \      /
                                Table:
                                JAC_IDATA
                                CUSTOM_COLLECTOR_V5

Index:      Table:
DB2INST2   JAC_IDATA
CUSTOM_COLLECTOR_INDEX  CUSTOM_COLLECTOR_V5

```

The explain tool output shows that no table scans were used, instead only indexes were referred to when executing the query. Again this method can be used to optimize SQL statement execution for tables created for custom collectors and for SQL statements created for reports and policies.

## Backing up DB2 Database

You should back up the IBM Security Compliance Manager database be backed up at regular intervals. A backup lets you restore the database in the case of a catastrophic hardware or software failure.

**To backup the DB2 database:**

1. Stop IBM Security Compliance Manager server: `./scmserver stop`
2. Log in to the database server as the DB2 instance owner (for example db2inst1).
3. Type `db2` at the DB2 command line.
4. Type `connect to jac`
5. Type `force application all`
6. Type `db2stop`
7. Type `db2start`
8. Type `backup database jac to /backupdir`
9. Type `force application all`
10. Type `db2stop`
11. Type `db2start`
12. Type `quit` to exit from the DB2 command line.
13. Restart IBM Security Compliance Manager server: `./scmserver start`

**To restore the DB2 database from a backup**

1. Stop IBM Security Compliance Manager server: `./scmserver stop`
2. Log in to the database server as the instance owner (for example db2inst1).
3. Type `db2` at the DB2 command line.
4. Type `connect to jac`
5. Type `force application all`
6. Type `db2stop`
7. Type `db2start`
8. Type `restore database jac from /backupdir`
9. Type `force application all`
10. Type `db2stop`
11. Type `db2start`

12. Type `quit` to exit from the DB2 command line.
13. Restart IBM Security Compliance Manager server: `./scmserver start`

## 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 give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
500 Columbus Avenue  
Thornwood, NY 10594  
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, 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  
224A/101  
11400 Burnet Road  
Austin, TX 78758  
USA

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

The licensed program describe 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.

Customers are responsible for ensuring their own compliance with various laws such as the Graham-Leach-Bliley Act, the Sarbanes-Oxley Act, and the Health Insurance Portability and Accountability Act. It is the customer's sole responsibility to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal, accounting or auditing advice, or represent or warrant that its products or services will ensure that customer is in compliance with any law.

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 measurement 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.

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.

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

## **Trademarks**

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

AIX  
DB2  
DB2 Universal Database  
IBM  
IBM logo  
pSeries  
RS/6000  
Tivoli  
Tivoli logo

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

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

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

Intel, Intel Inside (logos), MMX and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

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

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