IBM i
7.3

Performance

Note

Before using this information and the product it supports, read the information in “Notices” on page 173.

This edition applies to IBM i 7.3 (product number 5770-SS1) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CISC models.

This document may contain references to Licensed Internal Code. Licensed Internal Code is Machine Code and is licensed to you under the terms of the IBM License Agreement for Machine Code.

© Copyright International Business Machines Corporation 2002, 2015.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Performance **1**
What's new in IBM i 7.3 1
PDF file for Performance 2
Managing system performance 2
Selecting a Performance Management Strategy 3
Determining when and how to expand your system 5
Comparing performance metrics before and after system changes 5
Tracking performance 6
Researching a performance problem 7
Identifying a performance problem 7
Identifying and resolving common performance problems 7
Collecting system performance data 9
Collecting information about system resource utilization 10
Collecting information about an application's performance 11
Dumping trace data 11
Dumping memory 12
The basics of IBM i Wait Accounting 12
Scenario: Improving system performance after an upgrade or migration 15
Displaying performance data 16
Tuning performance 17
Performing basic system tuning 17
Adjusting performance automatically 19
Determining when to use simultaneous multithreading 20
Electronic business performance 20
Client performance 21
Network performance 21
Java performance in IBM i 22
IBM HTTP Server performance 22
WebSphere performance 23
Applications for performance management 24
Performance data collectors 26
Collection Services 26
How Collection Services works 27
Creating database files from Collection Services data 28
Customizing data collections 29
Collection Services support for system monitoring 31
Collection Services support for historical data 33
Implementing user-defined categories in Collection Services 34
Managing collection objects 43
User-defined transactions 43
Finding wait statistics for a job, task, or thread 51
Understanding disk consumption by Collection Services 51
Collecting and displaying CPU utilization for all partitions 53
Collecting ARM performance data 54
Short lifespan threads and tasks 54
IBM i Job Watcher 55
IBM i Disk Watcher 55
Performance Explorer 56
Performance Explorer concepts 56
Configuring Performance Explorer 63
Viewing and analyzing data 65

IBM Navigator for i 65
IBM Navigator for i Performance interface 65
IBM Navigator for i Monitors 90
IBM Performance Management for Power Systems (PM for Power Systems) - support for IBM i 98
PM Agent concepts 98
Configuring PM Agent 99
Managing PM Agent 105
IBM Systems Workload Estimator 107
IBM Performance Tools for i 108
Performance Tools concepts 108
Installing and configuring Performance Tools 112
Performance Tools reports 113
Scenarios: Performance 166
Scenario: Improving system performance after an upgrade or migration 166
Scenario: System monitor 167
Scenario: Message monitor 168
Related information for Performance 169

Notices 173
Programming interface information 174
Trademarks 174
Terms and conditions 175

Performance

Monitoring and managing your system's performance is critical to ensure you are keeping pace with the changing demands of your business.

To respond to business changes effectively, your system must change too. Managing your system, at first glance, might seem like just another time-consuming job. But the investment pays off soon because the system runs more efficiently, and this is reflected in your business. It is efficient because changes are planned and managed.

Managing performance of any system can be a complex task that requires a thorough understanding of that system's hardware and software. IBM® i is an industry leader in the area of performance management and has many qualities that are not found in other systems, including unparalleled performance metrics, always-on collection services, and graphical viewing of performance data. While understanding all the different processes that affect system performance can be challenging and resolving performance problems requires the effective use of a large suite of tools, the functions offered by IBM i are intended to make this job easier for users.

This topic will guide you through the tasks and tools associated with performance management.

Note: By using the following code examples, you agree to the terms of the "Code license and disclaimer information" on page 171.

Related concepts

Work management

What's new in IBM i 7.3

Learn what is new or changed in the performance topic this release.

Collection Services

- Named memory pools: The main storage pool name is now stored in files QAPMPOOLB and QAPMSMPOL.
- Timestamp modernization: An SQL time stamp format of the sample interval date and time was added to the database files. An SQL time stamp format of the Coordinated Universal Time of the sample interval date and time was also added. The time zone name and time zone offset from Coordinated Universal Time is stored in file QAPMCONF.
- HTTP system monitor metrics: A new file QAPMSMHTP contains summarized metrics from IBM HTTP Server (powered by Apache) data that is used in support of system monitoring.
- Many other existing files have changes and new fields. For more information, see the Collection Services data files topic.

Collection Services support for historical data

Collection Services can be configured to create and maintain a historical performance data collection. The **Performance** > **Graph History** task in IBM Navigator for i can be used to analyze the historical performance data in chart or table form.

A historical performance data collection contains data that is collected by Collection Services. The data is reduced and summarized with the intent of providing data to be kept for a long time. The data is intended to be just a subset of all the original Collection Services collected data.

© Copyright IBM Corp. 2002, 2015

IBM Navigator for i Performance interface

The performance interface includes Performance Data Investigator (PDI), Collection Manager, Batch Model, Performance Reports, System Monitor, and web-based GUI interfaces for Collection Services, Job Watcher, and Disk Watcher. New functions for Graph History and System Monitor are described here.

- Graph History: The new **Performance** > **Graph History** task provides a graphical view of the historical performance data collection created by Collection Services. You can view historical performance data by using the **Graph History** task in IBM Navigator for i. To view the historical data, you must use Collection Services to collect data and enable historical data creation in the Collection Services configuration.
- System Monitor: To view all monitor metrics in one place, a new **Visualize Monitor Data** action from the System Monitor function of IBM Navigator for i was added.

IBM Navigator for i Monitors

New performance metrics can now be monitored with IBM Navigator for i system monitors. Disk read response time, disk write response time, and various new IBM HTTP Server (powered by Apache) server metrics can now be monitored.

New MOVPFRCOL command

The Move Performance Collection (MOVPFRCOL) command moves a performance collection from one library to another.

PDF file for Performance

You can view and print a PDF file of this information.

- To view or download the PDF version of the performance topic, select Performance. This PDF does not include the reference information for performance.
- To view or download the PDF version of the reference information for performance, select Reference information for Performance.

Saving PDF files

To save a PDF on your workstation for viewing or printing:

1. Right-click the PDF link in your browser.
2. Click the option that saves the PDF locally.
3. Navigate to the directory in which you would like to save the PDF.
4. Click **Save**.

Downloading Adobe Reader

You need Adobe Reader installed on your system to view or print these PDFs. You can download a free copy from the Adobe Web site (https://get.adobe.com/reader/).

Related reference

Related information for Performance
The following product manuals and IBM Redbooks (in PDF format), websites, and IBM Knowledge Center topics relate to the Performance topic. You can view or print any of the PDFs.

Managing system performance

Successfully managing performance ensures that your system is efficiently using resources and that your server provides the best possible services to your users and to your business needs. Moreover, effective

performance management can help you quickly respond to changes in your system and can save you money by postponing costly upgrades and service fees.

Performance management is necessary to optimize utilization of your computer system by measuring current capabilities, recognizing trends, and making appropriate adjustments to satisfy end user and management requirements such as response time or job throughput. It is needed to maintain business efficiency and avoid prolonged suspension of normal business activities. Therefore, managing performance is part of your daily operations.

Understanding the factors that affect system performance helps you respond to problems and make better long-term plans. Effective planning can prevent potential performance problems from developing and ensures that you have the system capacity to handle your current and growing workloads.

Related reference

Performance Management on IBM iSee the Performance Management website for a wealth of reference information for the IBM i operating system, including white papers, articles, the latest tool documentation, and more.

Selecting a Performance Management Strategy

Developing a good performance management strategy will help you manage your system's performance.

Your performance management strategy depends in a large part on the amount of time you can afford to spend managing performance. If you are working with a small company, you may be managing many different aspects of your business and cannot devote many hours to managing performance. Many large companies employ performance specialists to keep their systems tuned and running effectively.

Different business needs require different performance management strategies. For determining a basic performance management strategy and for identifying which performance applications to use, classify your company in one of three categories: small business, mid-sized business, and large business. The business resources vary for each size, and your management strategy will vary accordingly.

Small business

A small business most likely has fewer resources to devote to managing performance than a larger business. For that reason, use as much automation as possible. You can use IBM Performance Management for Power Systems (PM for Power Systems) to have your performance data sent directly to IBM where it will be compiled and generated into reports for you. This not only saves you time, but IBM also makes suggestions to you when your server needs an upgrade.

The following is a list of recommended performance applications for a small business:

- IBM Navigator for i Performance interface: Display and manage performance data.
- Collection Services: Collect sample data at user-defined intervals for later analysis.
- IBM Performance Management for Power Systems (PM for Power Systems): Automate the collection, archival, and analysis of system performance data.
- IBM Performance Tools for i: Gather, analyze, and maintain system performance information.
- IBM Navigator for i Monitors: Observe graphical representations of system performance, and automate responses to predefined events or conditions.

Mid-sized business

The mid-sized business probably has more resources devoted to managing performance than the small business. You may still want to automate as much as possible and can also benefit from using IBM Performance Management for Power Systems (PM for Power Systems).

The following is a list of recommended performance applications for a mid-sized business:

- IBM Navigator for i Performance interface: Display and manage performance data.
- Collection Services: Collect sample data at user-defined intervals for later analysis.

- IBM Performance Management for Power Systems (PM for Power Systems): Automate the collection, archival, and analysis of system performance data.
- IBM Performance Tools for i: Gather, analyze, and maintain system performance information.
- IBM Navigator for i Monitors: Observe graphical representations of system performance, and automate responses to predefined events or conditions.

Large business

The large business has resources devoted to managing performance.

The following is a list of recommended performance applications for a large business:

- IBM Navigator for i Performance interface: Display and manage performance data.
- Collection Services: Collect sample data at user-defined intervals for later analysis.
- IBM Performance Management for Power Systems (PM for Power Systems): Automate the collection, archival, and analysis of system performance data.
- IBM Performance Tools for i: Gather, analyze, and maintain system performance information.
- IBM Navigator for i Monitors: Observe graphical representations of system performance, and automate responses to predefined events or conditions.
- IBM i Job Watcher: Collect detailed information about a specific job or thread resource.
- IBM i Disk Watcher: Collect detailed information about disk performance data.
- Performance explorer: Collect detailed information about a specific application or system resource.

Related concepts

IBM Navigator for i Performance interface
Use the IBM Navigator for i Performance interface to view, collect, and manage performance data by bringing together various performance information and tools into one centralized place.

Collection Services
Collection Services provides for the collection of system and job level performance data. It is the primary collector of performance data.

IBM i Job Watcher
IBM i Job Watcher provides for the collection of job data for any or all jobs, threads, and tasks on the system. It provides call stacks, SQL statements, objects being waited on, Java JVM statistics, wait statistics and more which are used to diagnose job related performance problems.

IBM i Disk Watcher
IBM i Disk Watcher provides for the collection of disk performance data to diagnose disk related performance problems.

IBM Performance Management for Power Systems (PM for Power Systems) - support for IBM i
The IBM Performance Management for Power Systems (PM for Power Systems) in support of IBM i offering automates the collection, archival, and analysis of system performance data and returns reports to help you manage system resources and capacity.

Performance Explorer
Performance Explorer collects more detailed information about a specific application, program or system resource, and provides detailed insight into a specific performance problem. This includes the capability both to perform several types and levels of traces and to run detailed reports.

Related reference

IBM Performance Tools for i
The IBM Performance Tools for i licensed program product includes many supplemental features that supplement or extend the capabilities of the basic performance tools that are available in the operating system.

IBM Navigator for i Monitors
IBM Navigator for i monitors track current information about the performance of your system. Additionally, you can use them to carry out predefined actions when a specific event occurs. Monitors

continue to monitor and perform any threshold commands or actions you specified until you stop the monitor.

Determining when and how to expand your system

As your business needs change, your system must also change. To prepare for any changes, you will want to model the current system and then see what would happen if the system, the configuration, or the workload were changed.

As your business needs evolve, so do your system needs. To plan for future system needs and growth, you will need to determine what would happen if the system, the configuration, or the workload were changed. This process is called trend analysis and should be done monthly. As your system approaches resource capacity guidelines, you may want to gather this data more frequently.

Trend analysis should be done separately for interactive and batch environments. If your company uses a certain application extensively, you may want to perform a trend analysis for the application. Another environment that may be important to track would be the end-of-month processing. It is important that you collect trend analysis data consistently. If your system's peak workload hours are between 10:00 AM and 2:00 PM and you collect trend analysis data for this time period, do not compare this data to data collected from other time periods.

To do a proper job of capacity planning and performance analysis, you must collect, analyze, maintain, and archive performance data. IBM offers several tools that help you with your capacity planning, resource estimating, and sizing:

- IBM Performance Management for Power Systems (PM for Power Systems)
- IBM Systems Workload Estimator

Related concepts

IBM Performance Management for Power Systems (PM for Power Systems) - support for IBM i
The IBM Performance Management for Power Systems (PM for Power Systems) in support of IBM i offering automates the collection, archival, and analysis of system performance data and returns reports to help you manage system resources and capacity.

IBM Systems Workload Estimator
The IBM Systems Workload Estimator is a Web-based sizing tool for System i, System p, and System x. You can use this tool to size a new system, to size an upgrade to an existing system, or to size a consolidation of several systems.

Related reference

Selecting a Performance Management Strategy
Developing a good performance management strategy will help you manage your system's performance.

Comparing performance metrics before and after system changes

Comparing performance metrics before and after system changes provide important information for both troubleshooting and planning.

You should establish a set of system performance metrics before any major change in the system configuration, for example adding a new application or performing a system upgrade. Maintaining accurate system performance metrics can provide essential troubleshooting information. At a minimum, system performance metrics should include current collection objects from Collection Services.

Related concepts

Collection Services

Collection Services provides for the collection of system and job level performance data. It is the primary collector of performance data.

Tracking performance

Tracking your system performance over time allows you to plan for your system's growth and ensures that you have data to help isolate and identify the cause of performance problems. Learn which applications to use and how to routinely collect performance data.

Tracking system performance helps you identify trends that can help you tune your system configuration and make the best choices about when and how to upgrade your system. Moreover, when problems occur, it is essential to have performance data from before and after the incident to narrow down the cause of the performance problem, and to find an appropriate resolution.

The system includes several applications for tracking performance trends and maintaining a historical record of performance data. Most of these applications use the data collected by Collection Services. You can use Collection Services to watch for trends in the following areas:

- Trends in system resource utilization. You can use this information to plan and specifically tailor system configuration changes and upgrades.
- Identification of stress on physical components of the configuration.
- Balance between the use of system resource by interactive jobs and batch jobs during peak and normal usage.
- Configuration changes. You can use Collection Services data to accurately predict the effect of changes like adding user groups, increased interactive jobs, and other changes.
- Identification of jobs that might be causing problems with other activity on the system
- Utilization level and trends for available communication lines.

The following tools will help you monitor your system performance over time:

- IBM Navigator for i Performance interface
- Collection Services
- IBM Performance Management for Power Systems (PM for Power Systems)

Related concepts

IBM Navigator for i Performance interface
Use the IBM Navigator for i Performance interface to view, collect, and manage performance data by bringing together various performance information and tools into one centralized place.

Collection Services
Collection Services provides for the collection of system and job level performance data. It is the primary collector of performance data.

IBM Performance Management for Power Systems (PM for Power Systems) - support for IBM i
The IBM Performance Management for Power Systems (PM for Power Systems) in support of IBM i offering automates the collection, archival, and analysis of system performance data and returns reports to help you manage system resources and capacity.

Related reference

IBM Navigator for i Monitors
IBM Navigator for i monitors track current information about the performance of your system. Additionally, you can use them to carry out predefined actions when a specific event occurs. Monitors

continue to monitor and perform any threshold commands or actions you specified until you stop the monitor.

Researching a performance problem

There are many options available to help you identify and resolve performance problems. Learn how to use the available tools and reports that can help you find the source of the performance problem.

Most of the tools that collect or analyze performance use either trace or sample data. Collection Services regularly collects sample data on a variety of system resources. Several tools analyze or report on this sample data, and you can use this to get a broader view of system resource utilization and to answer many common performance questions. IBM i Job Watcher and IBM i Disk Watcher also collect sample data. For more detailed performance information, several tools generate trace-level data. Often, trace-level data can provide detailed information about the behavior and resource consumption of jobs and applications on your system. Performance Explorer and the Start Performance Trace (STRPFRTRC) command are two common tools for generating trace data.

For example, if your system is running slowly, you might use the IBM Navigator for i monitors to look for problems. If you see that the CPU utilization is high, you could identify any jobs that seem to be using an unusually large amount of resources. Then, you may be able to correct the problem by making configuration changes. However, some problems will require additional information. To get detailed information about that job's performance you could start an IBM i Job Watcher collection for the desired job, gather detailed information about that job's behavior on the server, and potentially make changes to the originating program.

Identifying a performance problem

Learn the common steps involved with identifying a performance problem.

When you try to identify a performance problem, it is important to assess whether the hardware configuration is adequate to support the workload. Is there enough CPU capacity? Is the main storage sufficient for the different types of applications? Answering these questions first, perhaps through capacity modeling techniques, prevents needless effort later.

With an understanding of the symptoms of the problem and the objectives to be met, the analyst can formulate a hypothesis that may explain the cause of the problem. The analyst can use commands and tools available with IBM i and the IBM Performance Tools for i licensed program to collect and review data related to the system performance.

Reviewing the data helps you to further define the problem and helps you to validate or reject the hypothesis. Once the apparent cause or causes have been isolated, a solution can be proposed. When you handle one solution at a time, you can redesign and test programs. Again, the analyst's tools can, in many cases, measure the effectiveness of the solution and look for possible side effects.

To achieve optimum performance, you must recognize the interrelationship among the critical system resources and attempt to balance these resources, namely CPU, disk, main storage, and for communications, remote lines. Each of these resources can cause a performance degradation.

Improvements to system performance, whether to interactive throughput, interactive response time, batch throughput, or some combination of these, may take many forms, from simply adjusting activity level or pool size to changing the application code itself. In this instance, an activity level is a characteristic of a subsystem that specifies the maximum number of jobs that can compete at the same time for the processing unit.

Identifying and resolving common performance problems

Many different performance problems often affect common areas of the system. Learn how to research and resolve problems in common areas, for example, backup and recovery.

When performance problems occur on the system, they often affect certain areas of the system first. Refer to the following table for some methods available for researching performance on these system areas.

Area	Description	Available tools
Processor load	Determine if there are too many jobs on the system or if some jobs are using a large percentage of processor time.	• Performance Data Investigator found in IBM Navigator for i. • Work with Active Jobs (WRKACTJOB) command. • Work with System Activity (WRKSYSACT) command. • The work management function in IBM Navigator for i. • CPU utilization metrics within the IBM Navigator for i monitors.
Main storage	Investigate faulting and the wait-to-ineligible transitions.	• Performance Data Investigator found in IBM Navigator for i. • Disk storage metrics within the IBM Navigator for i monitors. • Work with System Status (WRKSYSSTS) command. • The Memory Pools function under Work Management in IBM Navigator for i.
Disk	Determine if there are too few arms or if the arms are too slow.	• Performance Data Investigator found in IBM Navigator for i. • Work with Disk Status (WRKDSKSTS) command. • Disk arm utilization metrics within the IBM Navigator for i monitors. • IBM Performance Tools for i System and Component report.
Communications	Find slow lines, errors on the line, or too many users for the line.	• Performance Data Investigator found in IBM Navigator for i. • IBM Performance Tools for i Component Report. • LAN utilization metrics within the IBM Navigator for i monitors.

Area	Description	Available tools
Software	Investigate locks and mutual exclusions (mutexes).	• Performance Data Investigator found in IBM Navigator for i. • IBM Performance Tools for i Locks report. • IBM Performance Tools for i Trace report. • Work with Object Locks (WRKOBJLCK) command. • View details of suspected jobs under Work Management in IBM Navigator for i. • Work with System Activity (WRKSYSACT) command. • Display Performance Data (DSPPFRDTA) command.
Backup and recovery	Investigate areas that affect backup and recovery and save and restore operations.	• IBM i Performance Capabilities Reference (Save/Restore Performance chapter).

Related concepts

IBM Navigator for i Performance interface
Use the IBM Navigator for i Performance interface to view, collect, and manage performance data by bringing together various performance information and tools into one centralized place.

Work management

Related reference

CL commands for performanceThe operating system includes several CL commands to help you manage and maintain system performance.

Monitor metrics
To effectively monitor system performance, you must decide which aspects of system performance you want to monitor. IBM Navigator for i offers various performance measurements, which are known as *metrics*, to help you pinpoint different aspects of system performance.

Performance Management on IBM i web site - Performance Capabilities Reference PDFFrom the Performance Management on IBM i website, select the appropriate Performance Capabilities Reference PDF. See the Save/Restore Performance chapter of the Performance Capabilities Reference for information about backup and recovery related performance.

Backup and recovery FAQThis topic contains questions and answers about backup and recovery procedures and concepts.

Collecting system performance data

Collecting data is an important step toward improving performance.

When you collect performance data, you gather information about your system that can be used to understand response times and throughput. It is a way to capture the performance status of the system, or set of systems, involved in getting your work done. The collection of data provides a context, or a starting point, for any comparisons and analysis that can be done later. When you use your first data collections, you have a benchmark for future improvements and a start on improving your performance today. You can use the performance data you collect to make adjustments, improve response times, and help your systems achieve peak performance. Performance problem analysis often begins with the simple question: "What changed?" Performance data helps you answer that question.

There are four collectors that have access to and can collect the data.

- Collection Services
- Job Watcher
- Disk Watcher
- Performance Explorer

Related concepts

Collection Services
Collection Services provides for the collection of system and job level performance data. It is the primary collector of performance data.

IBM i Job Watcher
IBM i Job Watcher provides for the collection of job data for any or all jobs, threads, and tasks on the system. It provides call stacks, SQL statements, objects being waited on, Java JVM statistics, wait statistics and more which are used to diagnose job related performance problems.

IBM i Disk Watcher
IBM i Disk Watcher provides for the collection of disk performance data to diagnose disk related performance problems.

Performance Explorer
Performance Explorer collects more detailed information about a specific application, program or system resource, and provides detailed insight into a specific performance problem. This includes the capability both to perform several types and levels of traces and to run detailed reports.

Collecting information about system resource utilization

Several tools monitor how resources like central processing unit (CPU), disk space, interactive capacity, and many other elements, are being used. You can use these tools to start identifying problem areas.

Many tools are available to help you monitor and track the way the system and your applications are using the available resources. You can use this information as a starting point for problem analysis, and to identify trends that will help you with capacity planning and managing the growth of your system.

See the following topics to learn how and when to use these tools:

- IBM Navigator for i Performance interface
- IBM Navigator for i monitors
- CL commands for performance
- IBM Performance Management for Power Systems (PM for Power Systems)

Related concepts

IBM Navigator for i Performance interface
Use the IBM Navigator for i Performance interface to view, collect, and manage performance data by bringing together various performance information and tools into one centralized place.

IBM Performance Management for Power Systems (PM for Power Systems) - support for IBM i
The IBM Performance Management for Power Systems (PM for Power Systems) in support of IBM i offering automates the collection, archival, and analysis of system performance data and returns reports to help you manage system resources and capacity.

Related reference

IBM Navigator for i Monitors
IBM Navigator for i monitors track current information about the performance of your system. Additionally, you can use them to carry out predefined actions when a specific event occurs. Monitors continue to monitor and perform any threshold commands or actions you specified until you stop the monitor.

CL commands for performanceThe operating system includes several CL commands to help you manage and maintain system performance.

Collecting information about an application's performance

An application might be performing slowly for various reasons. You can use several of the tools included in IBM i and other licensed programs to help you get more information.

Collecting information about an application's performance is quite different from collecting information about system performance. Collecting application information can be done only with certain performance applications such as Performance Explorer and Job Watcher. Alternately, you can get an overview of application performance by using IBM Performance Tools for i to track and analyze server jobs.

Note: Collecting an application's performance data can significantly affect the performance of your system. Before beginning the collection, make sure that you have tried all other collection options.

The Start Performance Trace (STRPFRTRC) command collects multiprogramming and transaction data. After running this command, you can export the data to a database file with the Dump Trace (DMPTRC) command.

Related concepts

IBM i Job Watcher
IBM i Job Watcher provides for the collection of job data for any or all jobs, threads, and tasks on the system. It provides call stacks, SQL statements, objects being waited on, Java JVM statistics, wait statistics and more which are used to diagnose job related performance problems.

Performance Explorer
Performance Explorer collects more detailed information about a specific application, program or system resource, and provides detailed insight into a specific performance problem. This includes the capability both to perform several types and levels of traces and to run detailed reports.

Related reference

IBM Navigator for i Monitors
IBM Navigator for i monitors track current information about the performance of your system. Additionally, you can use them to carry out predefined actions when a specific event occurs. Monitors continue to monitor and perform any threshold commands or actions you specified until you stop the monitor.

IBM Performance Tools for i
The IBM Performance Tools for i licensed program product includes many supplemental features that supplement or extend the capabilities of the basic performance tools that are available in the operating system.

Start Performance Trace (STRPFRTRC) commandSee the Start Performance Trace (STRPFRTRC) command to collect Multiprogramming level (MPL) and Transaction trace data.

Java performance in IBM i
IBM i provides several configuration options and resources for optimizing the performance of Java applications or services on the system. Use this topic to learn about the Java environment and how to get the best possible performance from Java-based applications.

Dumping trace data

The Dump Trace (DMPTRC) command puts information from an internal trace table into a database file.

It is not a good practice to dump trace data during peak activity on a loaded system or within a high priority (interactive) job. You can delay a trace dump, but you want to dump the data before you forget that it exists. If the trace table becomes cleared for any reason, you lose the trace data. However, delaying the dump slightly and then using the DMPTRC command to dump the trace in a batch job can preserve performance for the users.

To dump trace data, issue the following command:

```
 DMPTRC MBR
(member-name) LIB
(library-name)
```

You must specify a member name and a library name in which to store the data. You can collect sample-based data with Collection Services at the same time that you collect trace information. When you collect sample data and trace data together like this, you should place their data into consistently named members. In other words, the names that you provide in the CRTPFRDTA TOMBR and TOLIB parameters should be the same as the names that you provide in the DMPTRC MBR and LIB parameters.

Related concepts

Collection Services
Collection Services provides for the collection of system and job level performance data. It is the primary collector of performance data.

Related reference

Dump Trace (DMPTRC) commandSee the Dump Trace (DMPTRC) command to put information from an internal trace table into a database file.

Dumping memory

The Dump Main Memory Information (DMPMEMINF) command dumps information about pages of main memory to a file.

To dump memory data, issue the following command:

```
DMPMEMINF OUTFILE(MYLIBRARY/DMPMEMFILE)
```

The command to view the dump could be something like the following SQL:

```
SELECT count(*),POOL, OBJNAME, LIBNAME FROM mylibrary/dmpmemfile
group BY POOL, OBJNAME, LIBNAME
order by 1 desc
```

Related reference

Dump Main Memory Information (DMPMEMINF) commandSee the Dump Main Memory Information (DMPMEMINF) command to dump information about pages of main memory to a file.

The basics of IBM i Wait Accounting

Wait Accounting is the patented technology built into the IBM i operating system that tells you what a thread or task is doing when it appears that it is not doing anything.

When a thread or task is not executing, it is waiting. Wait accounting, a concept exclusive to IBM i, is a very powerful capability for detailed performance analysis. The following information is going to focus on waiting, why threads wait, and how you can use wait accounting to troubleshoot performance problems or to simply improve the performance of your applications.

A job is the basic mechanism through which work is done. Every job has at least one thread and may have multiple threads. Every thread is represented by a licensed internal code (LIC) task, but tasks also exist without the IBM i thread-level structures. LIC tasks are generally not visible externally except through the IBM i Performance or Service Tools. Wait accounting concepts apply to both threads and tasks, thus, the terms thread and task are used when referring to an executable piece of work.

A thread or task has two basic states it can be in:

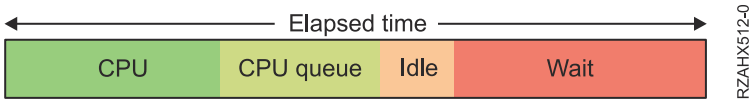
- Executing on the processor. This is the "running" state.
- Waiting to run on the processor.

  There are three key wait conditions:

  1. Ready to run, waiting for the processor. This is a special wait state and is generally referred to as "*CPU Queuing*. This means the thread or task is queued and is waiting to run on the CPU. There are a few different reasons that CPU queuing can occur. An example could be if the partition is overloaded and there is more work than the partition can accommodate, then work can be queued to wait for the CPU. This can be compared to a highway that has ramp meters; when the highway is congested, the ramp meters have a red signal so that the cars have to stop and wait before they can enter traffic. Logical partitioning and simultaneous multithreading can also result in CPU queuing.

2. Idle waits. Idle waits are a normal and expected wait condition. Idle waits occur when the thread is waiting for external input. This input may come from a user, the network, or another application. Until that input is received, there is no work to be done.
3. Blocked waits. Blocked waits are a result of serialization mechanisms to synchronize access to shared resources. Blocked waits may be normal and expected. Examples include serialized access to updating a row in a table, disk I/O operations, or communications I/O operations. However, blocked waits may not be normal and it is these unexpected block points that are situations where wait accounting can be used to analyze the wait conditions.

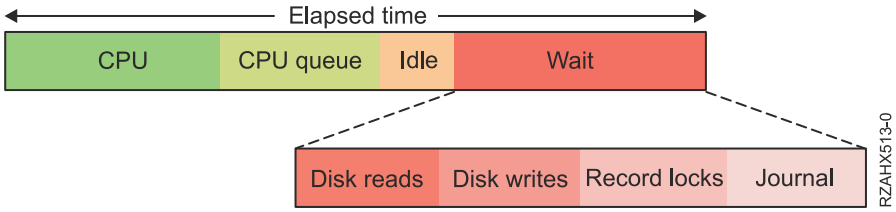
You can think of the life-time of a thread or a task in a graphical manner, breaking out the time spent running or waiting. This graphical description is called the "run-wait time signature". At a high level, this signature looks as follows:



Traditionally, the focus for improving the performance of an application was to have it use the CPU as efficiently as possible. On IBM i with wait accounting, we can examine the time spent waiting and understand what contributed to that wait time. If there are elements of waiting that can be reduced or eliminated, then the overall performance can also be improved.

Nearly all of the wait conditions in the IBM i operating system have been identified and enumerated - that is, each unique wait point is assigned a numerical value. This is possible because IBM has complete control over both the licensed internal code and the operating system. As of the IBM i 6.1 release, there are 268 unique wait conditions. Keeping track of over 250 unique wait conditions for every thread and task would consume too much storage, so a grouping approach has been used. Each unique wait condition is assigned to one of 32 groups, or "buckets". As threads or tasks go into and out of wait conditions, the task dispatcher maps the wait condition to the appropriate group.

If we take the run-wait time signature, using wait accounting, we can now identify the components that make up the time the thread or task was waiting. For example:



If the thread's wait time was due to reading and writing data to disk, locking records for serialized access, and journaling the data, we could see the waits broken out above. When you understand the types of waits that are involved, you can start to ask yourself some questions. For the example above, some of the questions that could be asked are:

- Are disk reads causing page faults? If so, are my pool sizes appropriate?
- What programs are causing the disk reads and writes? Is there unnecessary I/O that can be reduced or eliminated? Or can the I/O be done asynchronously?
- Is my record locking strategy optimal? Or am I locking records unnecessarily?
- What files are being journaled? Are all the journals required and optimally configured?

The following are the 32 wait groups or "buckets" that have been defined. The definition of the wait groups varies from release to release and may change in the future.

1. Time dispatched on a CPU
2. CPU queuing
3. Reserved
4. Other waits
5. Disk page faults

6. Disk non-fault reads
7. Disk space usage contention
8. Disk operation start contention
9. Disk writes
10. Disk other
11. Journaling
12. Semaphore contention
13. Mutex contention
14. Machine level gate serialization - call IBM support
15. Seize contention - call IBM support
16. Database record lock contention
17. Object lock contention
18. Ineligible waits
19. Main storage pool contention - call IBM support
20. Journal save while active
21. Reserved
22. Reserved
23. Reserved
24. Socket transmits
25. Socket receives
26. Socket other
27. IFS
28. PASE
29. Data queue receives
30. Idle/waiting for work
31. Synchronization Token contention
32. Abnormal contention - call IBM support

There are many of these wait groups that you may see surface if you do wait analysis on your application. Understanding what your application is doing and why it is waiting in those situations can possibly help you reduce or eliminate unnecessary waits.

If we take group 16 (Database record lock contention), there are actually several different enumerated waits within this group. They are:

- Read
- Update
- Weak
- Transfer
- Check
- Conflict exit

Holders and Waiters

Not only does IBM i keep track of what resource a thread or task is waiting on, it also keeps track of the thread or task that has the resource allocated to it. This is a very powerful feature. A "holder" is the thread or task that is using the serialized resource. A "waiter" is the thread or task that wants access to that serialized resource.

Call Stacks

IBM i also manages call stacks for every thread or task. This is independent of the wait accounting information. The call stack shows the programs that have been invoked and can be very useful in understanding the wait condition; knowing some of the logic that led up to either holding a resource or wanting to get access to it. The combination of holder, waiter, and call stacks provide a very powerful capability to analyze wait conditions.

Collecting and Analyzing the Data

Collection Services and Job Watcher are two performance data collection mechanisms on IBM i that collect the wait accounting information. Job Watcher also collects holder and waiter information, as well as call stacks. Once the performance data has been collected, you can graphically analyze the data. The iDoctor product has a Windows client for graphically viewing performance data. The IBM Navigator for i web console has the "Investigate Data" feature to graphically view performance data through a web browser interface.

Related concepts

IBM i Job Watcher
IBM i Job Watcher provides for the collection of job data for any or all jobs, threads, and tasks on the system. It provides call stacks, SQL statements, objects being waited on, Java JVM statistics, wait statistics and more which are used to diagnose job related performance problems.

Investigate Data
Selecting the Investigate Data task launches the powerful Performance Data Investigator tool. With this tool, you can view and analyze data that is stored in performance collections in chart or table form.

Scenario: Improving system performance after an upgrade or migration

In this scenario, you have just upgraded or migrated your system and it now appears to be running slower than before. This scenario will help you identify and fix your performance problem.

Situation

You recently upgraded your system to the newest release. After completing the upgrade and resuming normal operations, your system performance has decreased significantly. You would like to identify the cause of the performance problem and restore your system to normal performance levels.

Details

Several problems may result in decreased performance after upgrading the operating system. You can use the performance management tools included in IBM i and IBM Performance Tools for i licensed program (5770-PT1) to get more information about the performance problem and to narrow down suspected problems to a likely cause.

1. Check CPU utilization. Occasionally, a job will be unable to access some of its required resources after an upgrade. This may result in a single job consuming an unacceptable amount of the CPU resources.
   - Use WRKSYSACT, WRKSYSSTS, WRKACTJOB, or IBM Navigator for i monitors to find the total CPU utilization.
   - If CPU utilization is high, for example, greater than 90%, check the amount of CPU utilized by active jobs. If a single job is consuming more than 30% of the CPU resources, it may be missing file calls or objects. You can then refer to the vendor, for vendor-supplied programs, or the job's owner or programmer for additional support.
2. Start a performance trace with the STRPFRTRC command, and then use the system and component reports to identify and correct the following possible problems:
   - If the page fault rate for the machine pool is higher than 10 faults/second, give the machine pool more memory until the fault rate falls below this level.

- If the disk utilization is greater than 40%, look at the waiting and service time. If these values are acceptable, you may need to reduce workload to manage priorities.
- If the page faults in the user pool are unacceptably high, you might want to automatically tune performance.

3. Run the job summary report and refer to the Seize lock conflict report. If the number of seize or lock conflicts is high, ensure that the access path size is set to 1TB. If the seize or lock conflicts are on a user profile, and if the referenced user profile owns many objects, reduce the number of objects owned by that profile.

Related concepts

Performance Tools reportsPerformance Tools reports provide information on data that was collected over time. Use these reports to get more information about the performance and use of system resources.

Adjusting performance automatically
Most users should set up the system to make performance adjustment automatically. When new systems are shipped, they are configured to adjust automatically.

Related reference

STRPFRTRC commandSee the Start Performance Trace (STRPFRTRC) command to collect Multiprogramming level (MPL) and Transaction trace data.

Displaying performance data

After you have collected performance data, learn how to display the data using the most appropriate tool for your purposes.

Displaying performance data helps you analyze your system's performance more accurately. Performance data can be displayed in many different ways; however, you may find a certain performance application more appropriate in some situations. Most applications display data collected with either Collection Services or from a performance trace. The best way to access that data depends on whether you are attempting to resolve a performance problem, are monitoring your system performance to plan for future growth, or are identifying trends.

Displaying near real-time performance data

Use the following tools to display current or recent performance information:

- IBM Navigator for i Performance interface
- CL commands for performance
- IBM Performance Tools for i
- IBM Navigator for i monitors

Displaying historical performance data

Use the following tools to view data that is stored on your system:

- IBM Navigator for i Performance interface
- IBM Performance Management for Power Systems (PM for Power Systems)
- IBM Performance Tools for i

Related concepts

IBM Navigator for i Performance interface
Use the IBM Navigator for i Performance interface to view, collect, and manage performance data by bringing together various performance information and tools into one centralized place.

Collection Services
Collection Services provides for the collection of system and job level performance data. It is the primary collector of performance data.

IBM Performance Management for Power Systems (PM for Power Systems) - support for IBM i

The IBM Performance Management for Power Systems (PM for Power Systems) in support of IBM i offering automates the collection, archival, and analysis of system performance data and returns reports to help you manage system resources and capacity.

Related reference

CL commands for performanceThe operating system includes several CL commands to help you manage and maintain system performance.

IBM Navigator for i Monitors
IBM Navigator for i monitors track current information about the performance of your system. Additionally, you can use them to carry out predefined actions when a specific event occurs. Monitors continue to monitor and perform any threshold commands or actions you specified until you stop the monitor.

IBM Performance Tools for i
The IBM Performance Tools for i licensed program product includes many supplemental features that supplement or extend the capabilities of the basic performance tools that are available in the operating system.

Tuning performance

When you have identified a performance problem, you will want to tune the system to fix it.

The primary aim of performance tuning is to make the most efficient use of the system resources. Performance tuning is a way to adjust the performance of the system either manually or automatically. Many options exist for tuning your system. Each system environment is unique in that it requires you to observe performance and make adjustments that are best for your environment; in other words, you are required to do routine performance monitoring.

In addition, you may also want to consider some tuning options that allow processes and threads to achieve improved affinity for memory and processor resources.

Related reference

Performance system values: Thread affinitySee the thread affinity system value to specify whether secondary threads have affinity to the same group of processors and memory as the initial thread.

System and user defaults system values: Processor multitaskingSee the processor multitasking system value to specify whether processor multitasking is on, off, or determined by the system.

Performing basic system tuning

To tune your system's performance, you need to set up your initial tuning values, observe the system performance, review the values, and determine what to tune.

To begin tuning performance, you must first set initial tuning values by determining your initial machine and user pool sizes. Then, you can begin to observe the system performance.

Set initial tuning values

Setting initial tuning values includes the steps you take to initially configure the system pool sizes and activity levels to tune your system efficiently. The initial values are based on estimates; therefore, the estimates may require further tuning while the system is active. The following steps set the initial tuning values:

- Determine initial machine pool size

  Tune the machine pool to under 10 faults/second.
- Determine initial user pool sizes

  Tune user pools so that the sum of faults for all user pools is less than the number of processors times the processors percent busy. For example, in a system with four processors running at 50 percent busy (4 * 50 = 200), you would set the faults to less than 200 faults/seconds.

Observe system performance

To observe the system performance, you can use the Work with System Status (WRKSYSSTS), Work with Disk Status (WRKDSKSTS), and Work with Active Jobs (WRKACTJOB) commands. With each observation period, you should examine and evaluate the measurements of system performance against your performance goals.

1. Remove any irregular system activity. Irregular activities that may cause severe performance degradation are, for example, interactive program compilations, communications error recovery procedures (ERP), open query file (OPNQRYF), application errors, and signoff activity.
2. Use the WRKSYSSTS, WRKDSKSTS, WRKACTJOB and WRKSYSACT CL commands to display performance data.
3. Allow the system to collect data for a minimum of 5 minutes.
4. Evaluate the measures of performance against your performance goals. Typical measurements include:
   - Interactive throughput and response time, available from the WRKACTJOB display.
   - Batch throughput. Observe the auxiliary input/output (AuxIO) and CPU percentage (CPU%) values for active batch jobs.
   - Spooled throughput. Observe the auxiliary input/output (AuxIO) and CPU percentage (CPU%) values for active writers.
5. If you observe performance data that does not meet your expectations, tune your system based on the new data. Be sure to:
   - Measure and compare all key performance measurements.
   - Make and evaluate adjustments one at a time.

Review performance

Once you have set good tuning values, you should periodically review them to ensure your system continues to do well. Ongoing tuning consists of observing aspects of system performance and adjusting to recommended guidelines.

To gather meaningful statistics, you should observe system performance during typical levels of activity. For example, statistics gathered while no jobs are running on the system are of little value in assessing system performance. If performance is not satisfactory in spite of your best efforts, you should evaluate the capabilities of your configuration. To meet your objectives, consider the following:

- Processor upgrades
- Additional storage devices and controllers
- Additional main storage
- Application modification

By applying one or more of these approaches, you should achieve your objectives. If, after a reasonable effort, you are still unable to meet your objectives, you should determine whether your objectives are realistic for the type of work you are doing.

Determine what to tune

If your system performance has degraded and needs tuning, you need to identify the source of the performance problem and make specific corrections.

Related reference

Researching a performance problem

There are many options available to help you identify and resolve performance problems. Learn how to use the available tools and reports that can help you find the source of the performance problem.

Adjusting performance automatically

Most users should set up the system to make performance adjustment automatically. When new systems are shipped, they are configured to adjust automatically.

The system can set performance values automatically to provide efficient use of system resources. You can set up the system to tune system performance automatically by:

- Adjusting storage pool sizes and activity levels
- Adjusting storage pool paging

Adjusting storage pool sizes and activity levels

Use the QPFRADJ system value to control automatic tuning of storage pools and activity levels. This value indicates whether the system should adjust values at system restart (IPL) or periodically after restart.

You can set up the system to adjust performance at IPL, dynamically, or both.

- To set up the system to tune only at system restart (IPL):
  1. Select **Configuration and Service** > **System Values** from your IBM Navigator for i window.
  2. Select the **Performance** category.
  3. From the **Actions** menu, select **Properties**.
  4. Click the **Memory Pools** tab and select **At system restart** under **Automatically adjust memory pools and activity levels**. This is equivalent to setting the QPFRADJ system value to 1.
- To set up the system to make storage pool adjustments at system restart (IPL) and to make storage pool adjustments periodically after restart:
  1. Select **Configuration and Service** > **System Values** from your IBM Navigator for i window.
  2. Select the **Performance** category.
  3. From the **Actions** menu, select **Properties**.
  4. Click the **Memory Pools** tab and select both **At system restart** and **Periodically after restart** under **Automatically adjust memory pools and activity levels**. This is equivalent to setting the QPFRADJ system value to 2.
- To set up the system to make storage pool adjustments periodically after restart and not at system restart (IPL):
  1. Select **Configuration and Service** > **System Values** from your IBM Navigator for i window.
  2. Select the **Performance** category.
  3. From the **Actions** menu, select **Properties**.
  4. Click the **Memory Pools** tab and select **Periodically after restart** under **Automatically adjust memory pools and activity levels**. This is equivalent to setting the QPFRADJ system value to 3.

The storage pool values are not reset at system restart (IPL) to the initial values.

Adjusting storage pool paging

The dynamic tuning support provided by the system automatically adjusts pool sizes and activity levels for shared pools to improve the performance of the system. This tuning works by moving storage from storage pools that have minimal use to pools that would benefit from more storage. This tuning also sets activity levels to balance the number of threads in the pool with the storage allocated for the pool. To adjust the system, the tuner uses a guideline that is calculated based on the number of threads.

When dynamic adjustment is in effect, the following performance values are changed automatically to the appropriate settings:

- Machine (*MACHINE) memory pool size (QMCHPOOL system value)
- Base (*BASE) memory pool activity level (QBASACTLVL system value)
- Pool size and activity level for the shared pool *INTERACT
- Pool size and activity level for the shared pool *SPOOL
- Pool sizes and activity levels for the shared pools *SHRPOOL1-*SHRPOOL60

When dynamic adjustment is in effect (the QPFRADJ system value is set to 2 or 3), the job QPFRADJ that runs under profile QSYS is seen as active on the system.

Related concepts

Memory pools

Determining when to use simultaneous multithreading

Simultaneous multithreading allows sharing of process facilities to run two applications or two threads of the same application at the same time.

Although an operating system gives the impression that it is concurrently executing a very large number of tasks, each processor in a symmetric multiprocessor (SMP) traditionally executes a single task's instruction stream at any moment in time. The QPRCMLTTSK system value controls whether to enable the individual SMP processors to concurrently execute multiple instruction streams. Each instruction stream belongs to separate tasks or threads. When enabled, each individual processor is concurrently executing multiple tasks at the same time. The effect of its use will likely increase the performance capacity of a system or improve the responsiveness of a multithreaded application. Running multiple instruction streams at the same time does not improve the performance of any given task. As is the case with any performance recommendations, results vary in different environments.

The way that multithreading is done depends on the hardware model, and therefore, the performance capacity gains vary. Some models support this approach through a concept called simultaneous multithreading (SMT). This approach, called hyperthreading on some Intel processors, shares processor facilities to execute each task's instructions at the same time. Older processors use an approach called hardware multithreading (HMT). In the hardware multithreading approach, the hardware switches between the tasks on any long processing delay event, for example, a cache miss. Some models do not support any form of multithreading, which means the QPRCMLTTSK system value has no performance effect.

Because the QPRCMLTTSK system value enables the parallel use of shared processor resources, the performance gains depend highly on the application and the model. Refer to the *IBM i Performance Capabilities Reference* for guidelines about what performance gains might be expected through its use. In some cases, some applications are better served by disabling this system value.

Related reference

System and user defaults system values: Processor multitaskingSpecifies whether processor multitasking is on, off, or determined by the system.

Performance Management on IBM i web site - Performance Capabilities Reference PDFFrom the Performance Management on IBM i website, select the appropriate Performance Capabilities Reference PDF for guidelines about performance gains that might be expected by using the QPRCMLTTSK system value.

Electronic business performance

Managing performance in an electronic business environment introduces several new problems for the system administrator.

In addition to routine tuning on the server, administrators must also monitor and optimize the hardware and services supporting their electronic business transactions.

Related reference

Domino for iSeries sizing and performance tuningSee the IBM Redbooks publication Domino for iSeries Sizing and Performance Tuning for Domino for iSeries performance information.

Client performance

While the system administrator often has little control of the client-side of the electronic business network, you can use these recommendations to ensure that client devices are optimized for an electronic business environment.

Clients consisting of a PC with a Web browser often represent the electronic business component that administrators have the least direct control over. However, these components still have a significant effect on the end-to-end response time for web applications.

To help ensure high-end performance, client PCs should:

- Have adequate memory. Interfaces that use complex forms and graphics and resource intensive applets may also place demands on the client's processor.
- Use a high-speed and optimized network connection. Many communication adapters on a client PC may function while they are not optimized for their network environment. For more information, refer to the documentation for your communication hardware.
- Use browsers that fully support the required technologies. Moreover, browser support and performance should be a major concern when designing the Web interface.

Network performance

The network design, hardware resources, and traffic pressure often have a significant effect on the performance of electronic business applications. You can use this topic for information on how to optimize network performance, and tune server communication resources.

The network often plays a major role in the response time for web applications. Moreover, the performance impact for network components is often complex and difficult to measure because network traffic and the available bandwidth may change frequently and are affected by influences the system administrator may not have direct control over. However, there are several resources available to help you monitor and tune the communication resources on your server.

Refer to the following topics for more information:

Related concepts

IBM Navigator for i Performance interface
Use the IBM Navigator for i Performance interface to view, collect, and manage performance data by bringing together various performance information and tools into one centralized place.

IBM Navigator for i Monitors
IBM Navigator for i monitors track current information about the performance of your system. Additionally, you can use them to carry out predefined actions when a specific event occurs. Monitors continue to monitor and perform any threshold commands or actions you specified until you stop the monitor.

Tracking performance
Tracking your system performance over time allows you to plan for your system's growth and ensures that you have data to help isolate and identify the cause of performance problems. Learn which applications to use and how to routinely collect performance data.

Related reference

Performance data files: QAPMTCPThis Collection Services database file contains system-wide TCP/IP data.

Performance data files: QAPMTCPIFCThis Collection Services database file contains TCP/IP data that is related to individual TCP/IP interfaces.

Performance Management on IBM i web site - Performance Capabilities Reference PDFFrom the Performance Management on IBM i website, select the appropriate Performance Capabilities Reference PDF. The Performance Capabilities Reference provides detailed information, reports, and examples that can help you configure or tune your server for optimal performance. In particular, see Chapter 5, Communications Performance, to help you plan for and manage communication resources.

Java performance in IBM i

IBM i provides several configuration options and resources for optimizing the performance of Java applications or services on the system. Use this topic to learn about the Java environment and how to get the best possible performance from Java-based applications.

Java is often the language of choice for web-based applications. However, Java applications may require some optimization, both of the IBM i environment and of the Java application, to get optimal performance.

Use the following resources to learn about the Java environment in IBM i and the available tips and tools for analyzing and improving Java performance.

Related concepts

IBM Navigator for i Performance interface
Use the IBM Navigator for i Performance interface to view, collect, and manage performance data by bringing together various performance information and tools into one centralized place.

JavaThere are several important configuration choices and tools to help you get the best performance from Java applications.

Related reference

Collecting information about an application's performance
An application might be performing slowly for various reasons. You can use several of the tools included in IBM i and other licensed programs to help you get more information.

Performance Management on IBM i web site - Performance Capabilities Reference PDFFrom the Performance Management on IBM website, select the appropriate Performance Capabilities Reference PDF. The Performance Capabilities Reference provides detailed information, reports, and examples that can help you configure or tune your server for optimal performance. In particular, see Chapter 7, Java Performance, to help you optimize the performance of Java applications, and learn performance tips for programming in Java.

Java and WebSphere Performance on IBM eServer iSeries ServersUse this IBM Redbooks publication to learn how to plan for and configure your operating environment to maximize Java and WebSphere performance, and to help you collect and analyze performance data.

WebSphere J2EE Application Development for the IBM eServer iSeries ServerThis IBM Redbooks publication provides an introduction to J2EE, and offers suggestions and examples to help you successfully implement J2EE applications on the server.

IBM HTTP Server performance

The IBM HTTP Server is often an important part of electronic business performance. IBM provides several options and configuration choices that allow you to get the most out of this server.

IBM HTTP Server for IBM i can play an important role in the end-to-end performance of your Web-based applications, and several functions allow you to effectively monitor and improve Web server performance. In particular the Fast Response Caching Accelerator (FRCA) may allow you to significantly improve HTTP Server performance, particularly in predominantly static environments. The IBM HTTP Server for IBM i also provides a Web Performance Monitor and Web Performance Advisor.

Refer to the following resources for information on how to maximize HTTP Server performance.

Related concepts

IBM Navigator for i Performance interface
Use the IBM Navigator for i Performance interface to view, collect, and manage performance data by bringing together various performance information and tools into one centralized place.

HTTP ServerThe IBM HTTP Server for i documentation contains getting started, task oriented, and scenario-based information, supporting reference material, and conceptual information.

Related reference

Collection Services data files: QAPMHTTPBThis Collection Services database file contains basic data that is collected by the IBM HTTP Server (powered by Apache) category.

Collection Services data files: QAPMHTTPDThis Collection Services database file contains detail data that is collected by the IBM HTTP Server (powered by Apache) category.

Performance Management on IBM i web site - Performance Capabilities Reference PDFFrom the Performance Management on IBM i website, select the appropriate Performance Capabilities Reference PDF. The Performance Capabilities Reference provides detailed information, reports, and examples that can help you configure or tune your system for optimal performance. In particular, see Chapter 6, Web Server and Web Commerce, for HTTP server performance specifications, planning information, and performance tips.

IBM HTTP Server (powered by Apache): An Integrated Solution for IBM eServer iSeries serversUse this IBM Redbooks publication to get an in-depth description of HTTP Server (Powered by Apache) for i5/OS, including examples for configuring HTTP Server in common usage scenarios.

AS/400 HTTP Server Performance and Capacity PlanningUse this IBM Redbooks publication to learn about HTTP server impacts on performance tuning and planning. This publication also includes suggestions for using Performance Management tools to collect, interpret, and respond to web server performance data.

WebSphere performance

WebSphere Application Server is the electronic business application deployment environment of choice. Use this topic to learn how to plan for and optimize performance in a WebSphere environment.

Managing system performance in a WebSphere environment presents several challenges to the administrator. Web-based transactions may consume more resources, and consume them differently than traditional communication workloads.

Refer to the following topics and resources to learn how to plan for optimal performance, and to adjust system resources in a WebSphere environment.

Related reference

Collection Services data files: QAPMWASAPPThis Collection Services database file contains information about applications that run on the IBM WebSphere Application Server.

Collection Services data files: QAPMWASCFGThis Collection Services database file contains configuration information about the different server jobs that run on the IBM WebSphere Application Server.

Collection Services data files: QAPMWASEJBThis Collection Services database file contains information about applications with enterprise JavaBeans (EJBs) running on the IBM WebSphere Application Server.

Collection Services data files: QAPMWASRSCThis Collection Services database file contains information about pooled resources that are associated with an IBM WebSphere Application Server.

Collection Services data files: QAPMWASSVRThis Collection Services database file contains information about the server jobs that run on the IBM WebSphere Application Server.

WebSphere Application Server Performance web siteThis website provides resources for each version of WebSphere Application Server, including many useful performance tips and recommendations. This resource is valuable for environments using servlets, Java Server Pages (JSPs) and Enterprise JavaBeans (EJBs).

DB2 UDB/WebSphere Performance Tuning GuideThis IBM Redbooks publication provides an introduction to both the WebSphere and DB2 environments, and offers suggestions, examples, and solutions to common performance problems that can help you optimize WebSphere and DB2 performance.

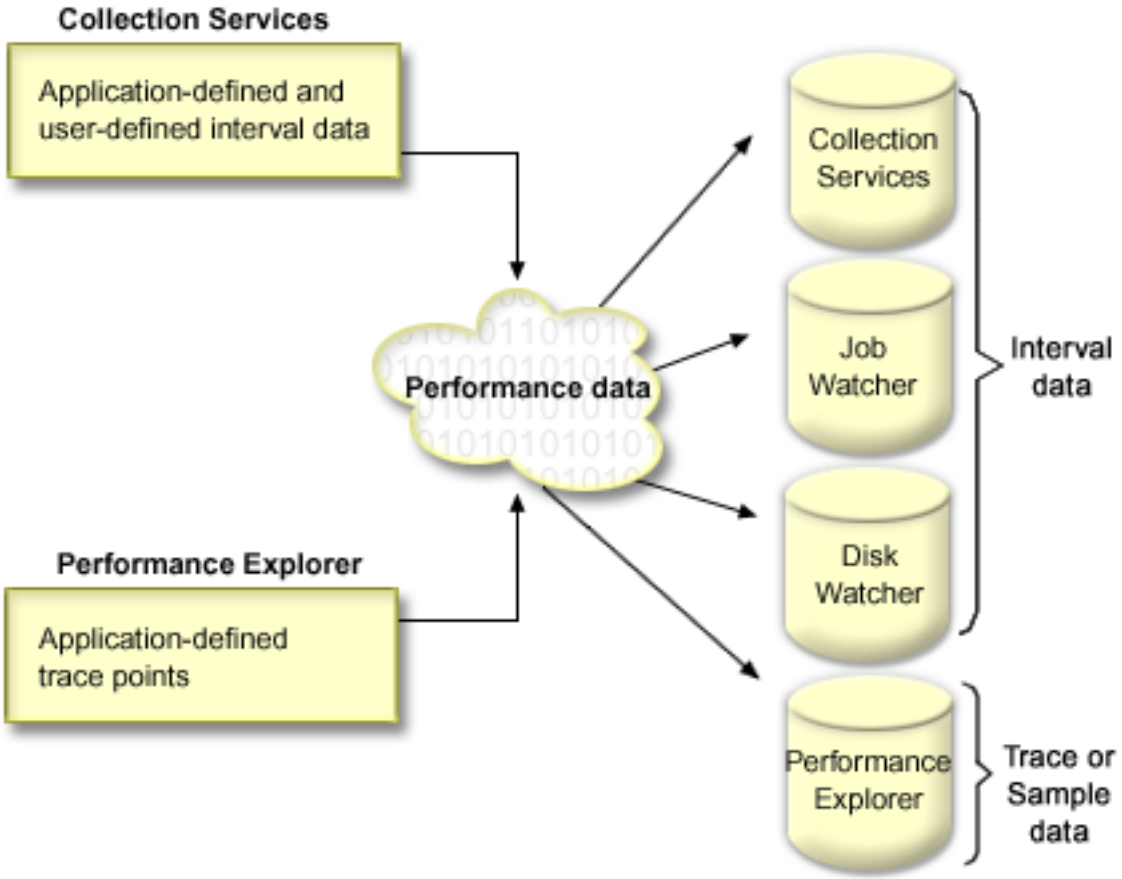
Java and WebSphere Performance on IBM eServer iSeries ServersUse this IBM Redbooks publication to learn how to plan for and configure you operating environment to maximize Java and WebSphere performance, and to help you collect and analyze performance data.

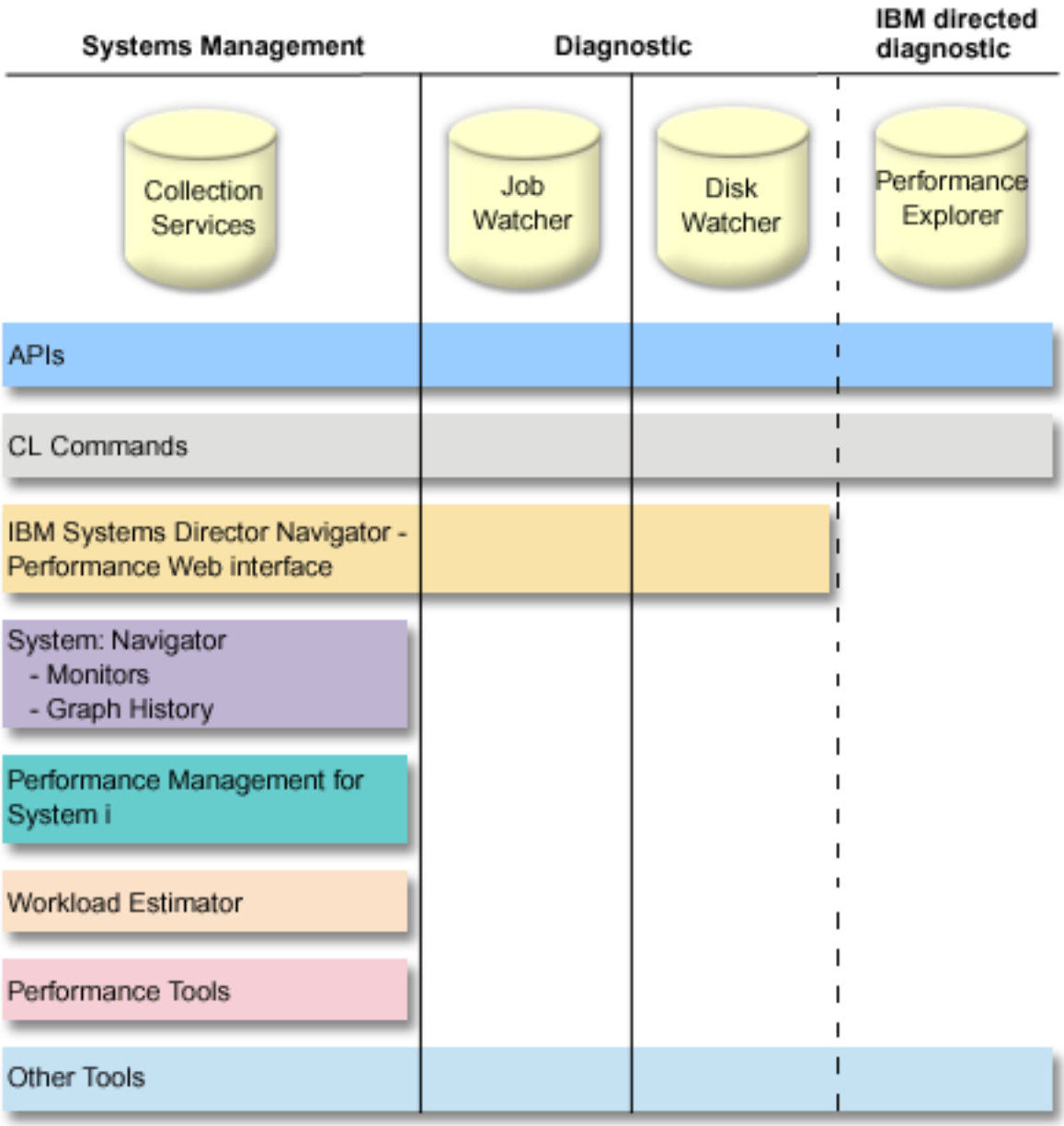
Performance Management on IBM i web site - Performance Capabilities Reference PDFFrom the Performance Management on IBM i website, select the appropriate Performance Capabilities Reference PDF. The Performance Capabilities Reference provides detailed information, reports, and examples that can help you configure or tune your server for optimal performance. In particular, see Chapter 6, "Web Server and Web Commerce", for performance tips specific to WebSphere Application Server.

Applications for performance management

Managing performance requires the use of a variety of specialized applications. Each of these applications offers a specific insight into system performance. Some applications collect the data, while others are used to display, analyze, monitor or manage the data collected.

The following figures represent the main performance applications. The cloud shape represents all of the data that exists in the system that can be collected. There are four collectors that have access to and can collect the data. Ultimately the data that is collected by a collector is deposited into a set of database files.





Each of the collectors has unique characteristics.

Collection Services
Collection Services provides for the collection of system and job level performance data. It is the primary collector of performance data. You can run this continuously to know what is happening with your system. Collection Services data is deposited into a management collection object and then converted and put into database files. The interval data that is collected is specified by either application defined or user defined interval data.

IBM i Job Watcher
IBM i Job Watcher provides for the collection of job data for any or all jobs, threads, tasks on the system. It provides call stacks, SQL statements, objects being waited on, Java JVM statistics, wait statistics and more which are used to diagnose job related performance problems.

IBM i Disk Watcher
IBM i Disk Watcher provides for the collection of disk performance data to diagnose disk related performance problems.

Performance Explorer

Performance explorer provides for the collection of detailed data at a program and application level to diagnose problems. It also traces the flow of work in an application and can be used to diagnose difficult performance problems. Application-defined performance explorer trace points, such as with Domino, NetServer, or WebSphere servers specify the data that is collected. It is intended to be used as directed by IBM. Performance Explorer data is deposited into a management collection object and then converted and put into database files.

The performance data contained in any of the database files can be accessed though APIs or CL commands. The performance data contained in some of the database files can be investigated and analyzed using one or more of a variety of tools that are further described in Viewing and analyzing data.

Performance data collectors

There are four collectors on IBM i that have access to and can collect the data. Each of the collectors has unique characteristics.

Collection Services

Collection Services provides for the collection of system and job level performance data. It is the primary collector of performance data.

Collection Services samples system and job level performance data continuously and automatically with minimal system overhead. It can collect data at regular time intervals of 15 seconds up to 1 hour (default is 15 minutes).

Data is collected from many system resources, such as:

- CPU
- Memory pools
- Disk (internal and external)
- Communications

Use Collection Services performance data to:

- Monitor how your system is running
- Investigate a reported performance problem
- Understand resource usage and how it changed over time

Collection Services can be configured and managed through the IBM Navigator for i Performance interface or CL commands.

Related tasks

Activating PM Agent
PM Agent is a part of the operating system and you must activate it to use its collecting capabilities.

Related reference

Start Performance Collection (STRPFRCOL) commandSee the Start Performance Collection (STRPFRCOL) command for information on how to start data collection.

CL commands for performanceThe operating system includes several CL commands to help you manage and maintain system performance.

Performance Management APIsSee the Performance Management APIs for information about how to use Performance Management APIs to collect and manage performance data.

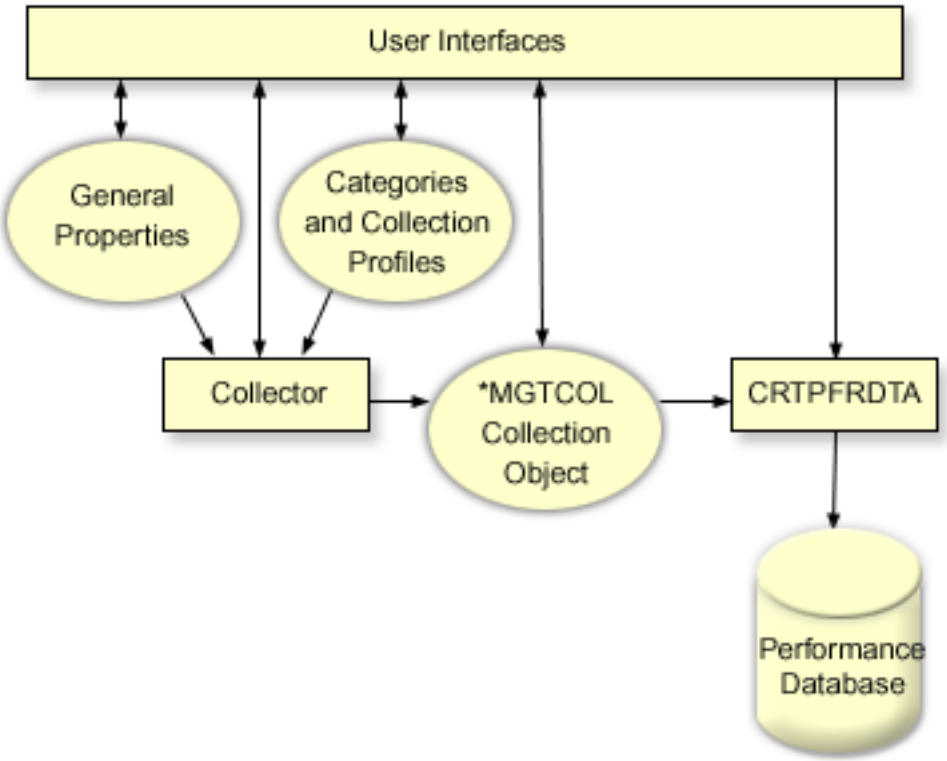
Collection Services data filesSee the Collection Services data files topic for information about the database files that are created by Collection Services that contain performance data.

How Collection Services works

Collection Services stores data for each collection in a single collection object from which you can create as many different sets of database files as you need.

Storing the data in a single collection object results in lower system overhead when collecting performance data. If you elect to create the database files during collection, Collection Services uses a lower priority (50) batch job to update these files. This low collection overhead makes it practical to collect detailed performance data at short intervals on a continuous basis. Collection Services enables you to establish a network-wide system policy for collecting and retaining performance data and to implement that policy automatically. For as long as you retain the management collection objects, if the need arises, you have the capability to look back and analyze performance-related events down to the level of detail that you collected.

The following figure provides an overview of the Collection Services elements:



User interfaces
Several methods exist that allow you to access the different features of Collection Services. For example, you can use CL commands, APIs, and the IBM Navigator for i Performance interface.

General properties
General properties define how a collection should be accomplished, and they control automatic collection attributes.

Data categories
Data categories identify the types of data to collect. You can configure categories independently to control what data is collected and how often the data is collected.

Collection profiles
Collection profiles provide a means to save and activate a particular category configuration.

Performance collector
The performance collector uses the general properties and category information to control the collection of performance data. You can start and stop the performance collector, or configure it to run automatically.

Collection Object
The collection object, a system object with a type of *MGTCOL, serves as an efficient storage medium for large quantities of performance data.

Create Performance Data (CRTPRFDTA) command
The CRTPFRDTA command processes data that is stored in the management collection object and generates the performance database files.

Performance database
The database files store the data that is processed by the CRTPFRDTA command. The files can be divided into these categories: Performance data files that contain time interval data, configuration data files, and trace data files.

Creating database files from Collection Services data

Use this information to manually or automatically create database files from Collection Services data.

Collection Services places the data you collected into management collection objects. To use this data, you must first place the data in a special set of database files. To create database files automatically as data is collected, select **Create database files** on the **Start Collection Services** dialog. You can also create the database files later when you want to export data to them from an existing management collection object.

You have many options that allow you to create database files.

- When you use Collection Services to collect performance data, you can create database files automatically as data is collected.
- You can create database files from the management collection object, where the data is stored after it has been collected. You can use the Create Performance Data (CRTPFRDTA) command to create a set of performance database files from performance information stored in a management collection (*MGTCOL) object. You can use either the IBM Navigator for i interface or the CRTPFRDTA command.

You can use the database files that you have created with the IBM Navigator for i Performance interface or other applications to produce performance reports. You can collect the performance data on one system and then move the management collection object (*MGTCOL) to another system to generate the performance data files and produce performance reports. This action allows you to analyze the performance data on another system without affecting the performance of the source system.

Using data stored in management collection objects instead of in database files

Why should you use the data stored in management collection objects instead of in the database files that you need to run your reports? Because you can manage the management collection objects separately from the database files, you can collect your performance data in small collection intervals (such as 5-minute intervals) and then create your database files with a longer sampling interval (such as 15-minute intervals).

From a single management collection object, you can create many different sets of database files for different purposes by specifying different data categories, different ranges of time, and different sampling intervals.

For example, you might collect performance data on the entire set of categories (all data, or the **Standard plus protocol** profile) in 5-minute collection intervals for 24 hours. From that one management collection object, you can create different sets of database files for different purposes. You could create one set of database files to run your normal daily performance reports. These files might contain data from all categories with a sampling interval of 15 minutes. Then, to analyze a particular performance problem, you could create another set of database files. These files might contain only data for a single category that you need to analyze, a specific time period within the 24 hours, and a more granular sampling interval of 5 minutes.

In addition, the single management collection object allows you to manage the data as a single object rather than as many files. The single collection object allows you to move the performance data between releases without converting the data. As long as you retain the collection objects, you can look back and analyze the performance-related events down to the level of detail that you collected.

Related tasks

Creating database files
To create database files, follow these steps.

Related reference

Create Performance Data (CRTPFRDTA) commandSee the Create Performance Data (CRTPFRDTA) command for information about creating performance database files.

Customizing data collections

When you use Collection Services to collect performance data, you control what data is collected and how often it is collected.

You can select from the collection profiles that are provided. The **Standard** profile collects the data categories that are typically needed by IBM Performance Tools for i licensed program. The **Standard plus protocol** profile collects the communications data categories and the data categories that are typically needed by IBM Performance Tools for i licensed program. Or you can select **Custom** to create your own customized profile. Several other profiles are also available.

When you create a customized profile, you can select from a list of available data categories, such as system CPU, local response time, or disk storage. For each category of data that you collect in your customized profile, you can specify how often the data is collected. For many categories, you want to select the default collection interval, which you can set from predefined settings between 15 seconds and 60 minutes. (The recommended setting is 5 minutes.)

Note: All categories use the default interval time except those categories with:

- Explicit time intervals defined in the **Custom** profile.
- Categories with interval restrictions such as disk storage and communications-related categories, which must collect at least every 5 minutes.

The collected data is stored in a management collection object (system object type *MGTCOL). To prevent these management collection objects from becoming too large, the collection must be cycled at regular intervals. *Cycling* a collection means to create a new collection object and begin storing data in it at the same time data collection stops in the original collection object. You can specify any cycle interval from 1 hour to 24 hours, depending on how you plan to use the data.

Related tasks

Configuring Collection Services
Configure Collection Services by doing the following.

Creating a Collection Services custom profile
Create a custom Collection Services profile by doing the following.

Collection Services collection profiles
Descriptions of the Collection Services collection profiles. The collection profiles define what is collected.

- **Minimum** - The Minimum collection profile includes the most essential categories. Note: This profile does not contain all the categories that are required by the tools in IBM Performance Tools for i licensed program. The Minimum collection profile includes the following categories:
  - System bus: This category contains data on the operation of each system bus.
  - Memory pool: This category contains memory pool configuration data and memory pool operation data.
  - Hardware configuration: This category contains hardware resource information for the system. This category contains the same data that the Display Hardware Resources (DSPHDWRSC) command acquires. Only the first instance of this data is reported in the database if more than one instance is encountered.
  - System CPU: This category contains data on system CPU usage for each processor.
  - System-level data: This category contains general system data that is used on a system-wide basis.

- Job (MI tasks and threads): This category contains information on every active task, job, and thread in the system. The data that is collected is provided by the machine interface (MI).
- Job (operating system): This category contains information on every active job in the system. The data that is collected is provided by the operating system.
- Disk storage: This category contains system storage unit data. It includes base storage unit information and operational data for disk drives.

- **Standard** - The Standard collection profile includes the data categories that are typically needed by the tools in IBM Performance Tools for i licensed program. The data categories in the Standard profile include all the categories in the Minimum profile plus the following categories:
  - Memory pool tuning: This category contains pool tuning configuration data for each system memory pool.
  - Subsystem: This category contains data on active subsystems and subsystem pools. Only the first instance of this data is reported in the database if more than one instance is encountered.
  - SNADS: This category contains transaction boundary information specific to active SNADS jobs in the system.
  - Local response time: This category contains response time information for workstations connected to 5254 controllers. Response time data is reported for each workstation and is saved in a set of response time buckets.
  - APPN: This category contains data on the system's APPN support. The data contains both general information and data classified according to transaction type and work activity.
  - SNA: This category contains data on the system's SNA support. Controller, task, and session data is reported for each active T2 task.
  - TCP/IP (base): This category contains system-wide performance information for TCP/IP.
  - User defined transaction data: This category contains data for application-defined transactions rather than IBM-defined transactions. You can create your own user-defined transactions.
  - IBM Domino for i: This category is included in this profile when the IBM Domino for i licensed program is installed on the system.
  - IBM HTTP Server for i (powered by Apache): This category is included in this profile when the IBM HTTP Server for i licensed program is installed on the system.
  - External storage: This category contains non-standardized data for disk units that are externally attached to an IBM i partition.
  - System internal data: This category contains internal data for the system.
  - Removable storage: This category contains data about removable storage devices connected to the system, more specifically tape device data.
  - SQL: This category contains performance information for SQL, more specifically SQL Plan Cache data.
- **Standard plus protocol** - The Standard plus protocol collection profile includes communications protocol data categories and the data categories that are typically needed by the tools in IBM Performance Tools for i licensed program. The data categories in the Standard plus protocol profile include all the categories in the Standard profile with the addition of the following categories:
  - Network server: This category contains information about network servers. For Integrated xSeries Servers, data is reported for CPU utilization. For virtual I/O adapters on hosting partitions (partitions that provide the physical resources), data is provided about the I/O activity that occurs within this partition due to the virtual device support that it is providing on behalf of guest partitions.
  - Communications (base): This category contains base protocol information for each communication line that is available for use (varied on).
  - Communications (station): This category contains station information for certain communication lines. Data is reported for each station that is available for use (varied on). Protocols supporting this data are Token Ring, Ethernet, DDI, Frame Relay, and X.25.

- Communications (SAP): This category contains Service Access Point (SAP) information for certain communication lines. Data is reported for each configured SAP within lines available for use (varied on). Protocols supporting this data are Token Ring, Ethernet, DDI, and Frame Relay.
- Data port services: This category contains performance data obtained from data port services. Data port services is Licensed Internal Code that supports the transfer of large volumes of data between a source system and one of any number of specified target systems in an IBM i cluster environment. Data port services is used by Licensed Internal Code clients, such as, remote independent auxiliary storage pool (ASP) mirroring.
- Logical partition: This category contains performance data that is collected from eligible partitions if the IBM Director Server (5761-DR1) licensed program is installed on the partition that is running Collection Services. To collect data from other partitions, the IBM Director Agent (5761-DA1) licensed program must be installed on the other partitions and the server must be authorized to the other partitions.
- TCP/IP interface: This category contains information for each active TCP/IP interface.

- **Enhanced capacity planning** - The data categories in the Enhanced capacity planning profile include all the categories in the Standard plus protocol profile with the addition of the PEX Data - Processor Efficiency data category. The PEX Data - Processor Efficiency data category contains the cycles per instruction for Performance Explorer (PEX) data. Data is collected to enhance capacity planning capabilities or for other purposes. Special considerations apply when you use this category:
  - A Performance Explorer definition, QPMIPEXPEI, is created. If the Performance Explorer definition exists, it is deleted and re-created.
  - This category requires Collection Services to start a Performance Explorer (PEX) collection (session-ID QPMINTPEXD). This collection can conflict with other Performance Explorer collections.
  - Do not end or start the QPMINTPEXD session manually because it affects the validity of the data collected.
  - When collection of this category stops, it also ends the Performance Explorer collection for session QPMINTPEXD.

  IBM no longer recommends using this profile.
- **Custom** - This option allows for customization of not only the data categories that are collected, but also the specification of the interval. You can have different categories of data collected at different intervals.

Related tasks

Creating a Collection Services custom profile
Create a custom Collection Services profile by doing the following.

Collection Services support for system monitoring

Collection Services now fully supports the concept of system monitoring. Collection Services can be configured to collect system monitor performance metrics, even without configuring and running a IBM Navigator for i System Monitor.

Configuring Collection Services system monitoring

System monitoring can be enabled in Collection Services several different ways:

- Creating and starting a IBM Navigator for i System Monitor
- Using the Configure Performance Collection (CFGPFRCOL) command
- Using the Change Collection Services Attributes (QypsChgColSrvAttributes) API

When system monitoring is enabled, the Collection Services collector collects configured categories at the specified system monitoring collection interval. All other categories are collected at their normal collection interval.

Note: A IBM Navigator for i System Monitor might override other system monitoring configuration settings that were made by using the CFGPFRCOL command or QypsChgColSrvAttributes API if the monitor requires data to be collected more frequently.

System monitor database files and collections

The set of QAPMSMxxx database files provide system monitor metrics. Monitored metrics are most often calculated as percentages (ex: CPU utilization) or rates (ex: bytes per second). Monitored metrics are also system level metrics, which means data for multiple entities (jobs, disk, ...) is grouped, summarized, or averaged.

System monitor database files can be created for any Collection Services collection, even when system monitoring is not enabled.

- Use the “Create Standard Summary Data (CRTPFRSUM)” parameter on the Configure Performance Collection (CFGPFRCOL) command. This parameter enables the creation of the system monitor files for the system created standard database file collection.
- Use the “Create Standard Summary Data (CRTPFRSUM)” parameter on the Create Performance Data (CRTPFRDTA) command. This parameter enables the creation of system monitor files for the user created standard database file collection.

Note: When the system monitor database files for a standard database file collection are created, the interval in the QAPMSMxxx files is the same as specified for CRTPFRDTA to be consistent with the rest of the collection.

When Collection Services system monitoring is enabled, a second create performance data job (CRTPFRDTA2) is submitted to create a second Collection Services file based collection that consists of the system monitor database files. This job (CRTPFRDTA2) operates similar to the existing CRTPFRDTA job and uses the configured collection library. The following are the differences between the system monitor CRTPFRDTA2 job and the CRTPFRDTA job:

- The interval for the CRTPFRDTA2 job is fixed at 15 seconds. The interval for the CRTPFRDTA job is the configured default collection interval (usually 15 minutes).
- The CRTPFRDTA2 job only creates database files for the categories that are identified as a system monitor category in the configuration. The CRTPFRDTA job generates database files for all categories that are defined in the configured collection profile.
- The collection name for CRTPFRDTA2 is based on the *MGTCOL name except that the name begins with “R” rather than “Q”.

In the collection that is created by the CRTPFRDTA2 job, both the system monitoring database files and the normal collection database files are created for the collection categories that are selected for system monitoring for the following reasons:

- Metrics that are defined in the system monitor files require that CRTPFRDTA process the categories that contain the data. If the source data for a metric category is not processed, nothing can be output for the metric.
- To support drill down capabilities within Performance Data Investigator.

Data retention for system monitor collections

The data retention and expiration for system monitor database file collections works similar to the data retention and expiration for standard Collection Services file based collections. The only differences are that a separate retention period is configured and used for system monitor collections and the expiration of system monitor collections is not affected by the PM Agent data reduction status.

Related tasks

Creating a system monitor
Use this information to learn how to set up a system monitor in IBM Navigator for i and how to configure it to take the best advantage of the available options.

Related reference

Monitor metrics

To effectively monitor system performance, you must decide which aspects of system performance you want to monitor. IBM Navigator for i offers various performance measurements, which are known as *metrics*, to help you pinpoint different aspects of system performance.

Configure Performance Collection (CFGPFRCOL) commandSee the Configure Performance Collection (CFGPFRCOL) command for information about configuring Collection Services.

QypsChgColSrvAttributes APIThe Change Collection Services Attributes (QypsChgColSrvAttributes) API changes system or global collection attributes of Collection Services.

Collection Services data files: QAPMSMCMNThis Collection Services database file contains summarized metrics from communication protocol data (*CMNBASE collection category) that may be used in support of system monitoring.

Collection Services data files: QAPMSMDSKThis Collection Services database file contains summarized metrics from disk data (*DISK collection category) that may be used in support of system monitoring.

Collection Services data files: QAPMSMJMIThis Collection Services database file contains summarized metrics from job data (*JOBMI collection category) that may be used in support of system monitoring.

Collection Services data files: QAPMSMJOSThis Collection Services database file contains summarized metrics from job data (*JOBOS collection category) that may be used in support of system monitoring.

Collection Services data files: QAPMSMPOLThis Collection Services database file contains summarized metrics from pool data (*POOL collection category) that may be used in support of system monitoring.

Collection Services data files: QAPMSMSYSThis Collection Services database file contains summarized metrics from system data (*SYSLVL collection category) that may be used in support of system monitoring.

Collection Services support for historical data

Collection Services can be configured to create and maintain a historical performance data collection. The **Performance** > **Graph History** task in IBM Navigator for i can be used to analyze the historical performance data in chart or table form.

A historical performance data collection contains data that is collected by Collection Services. The data is reduced and summarized with the intent of providing data to be kept for a long time. The data is intended to be just a subset of all the original Collection Services collected data.

Configuring Collection Services historical data

Configure Collection Services to create historical data by using one of the following options:

- Configuring Collection Services in IBM Navigator for i
- Using the Configure Performance Collection (CFGPFRCOL) command
- Using the Change Collection Services Attributes (QypsChgColSrvAttributes) API

When historical data creation is enabled, the Collection Services collector submits a batch job that is named QYPSPFRHST at cycle time that creates historical data. This job processes the management collection object that just ended and populates the historical collection with the data from the management collection object.

Historical performance database files and collections

The historical performance data collection is the set of QAPMHxxxxx database files in library QPFRHIST.

Summary historical data: System level or summarized metrics that are useful in identifying trends or detecting changes in a system over a long time. This type of data is stored in database files that begin with QAPMHMxxxx.

Detailed historical data: The data from which metrics are derived and other relevant supplementary data. This data is used to look deeper into a problem that was identified by looking at summary historical data. Only the top number of contributors for each metric is stored as historical detailed data, as defined by the value of the historical detailed data filter. This type of data is stored in database files that begin with QAPMHDxxxx.

Data retention for historical data collections

The amount of historical data that you are allowed to keep depends on whether PM for Power Systems (PM Agent) is active.

- If PM for Power Systems is not active, you can keep up to 7 days of detailed data and 1 month of summary data.
- If PM for Power Systems is active, then you can keep up to 60 days of detailed data and 50 years of summary data.

Expired historical data is deleted by the expiration thread that runs in the Collection Services collector job (QYPSPFRCOL). This thread is responsible for handling the deletion of all expired Collection Services collections. The collector thread deletes any records in the historical data files that are older than the summary and detailed data retention periods. The summary and detailed data retention periods are defined in the Collection Services historical configuration settings.

Related concepts

Graph History concepts
Configure and generate historical performance data for use by the **Graph History** task in IBM Navigator for i.

Related tasks

Configuring historical data for Graph History
Configure the historical performance data settings in Collection Services to generate historical performance data for use by the **Graph History** task in IBM Navigator for i.

Creating historical data for Graph History
To create historical performance data immediately for use in **Graph History**, follow these steps.

Viewing historical data with Graph History
Use the **Graph History** task in IBM Navigator for i to view historical performance data.

Activating PM Agent
PM Agent is a part of the operating system and you must activate it to use its collecting capabilities.

Related reference

Configure Performance Collection (CFGPFRCOL) commandSee the Configure Performance Collection (CFGPFRCOL) command for information about configuring Collection Services.

QypsChgColSrvAttributes APIThe Change Collection Services Attributes (QypsChgColSrvAttributes) API changes system or global collection attributes of Collection Services.

Historical data files

Implementing user-defined categories in Collection Services

The user-defined categories function in Collection Services enables applications to integrate performance data collection into Collection Services.

This allows you to gather data from an application by writing a data collection program, registering it, and integrating it with Collection Services. Collection Services will then call the data collection program at every collection interval, and will store the data in the collection object. You should use the Collection Object APIs listed below to access the data stored in the collection object. You may access the data in real-time, as it is being collected, or for as long as the collection object is retained.

To implement this function, you need to:

1. Develop a program to collect performance data for a new category in Collection Services.
2. Create a job description for your collection program. The job description QPMUSRCAT in QGPL provides an example, but does not represent default values or recommendations.
3. Register the new category and specify the data collection program.
   - Register: QypsRegCollectorDataCategory
   - De-register: QypsDeregCollectorDataCategory

After you register the category, Collection Services includes it in the list of available collection categories.

4. Add the category to your Collection Services profile, and then cycle Collection Services
5. Develop a program to query the collection object.
   - Retrieve active management collection object name: QpmRtvActiveMgtcolName (Used only for querying the collection object in real-time)
   - Retrieve management collection object attributes: QpmRtvMgtcolAttrs
   - Open management collection object: QpmOpenMgtcol
   - Close management collection object: QpmCloseMgtcol
   - Open management collection object repository: QpmOpenMgtcolRepo
   - Close management collection object repository: QpmCloseMgtcolRepo
   - Read management collection object data: QpmReadMgtcolData

Your customized collection program now runs at each collection interval, and the collected data is archived in the collection objects.

You can also implement the Java versions of these APIs. The required Java classes are included in ColSrv.jar, in the integrated file system (IFS) directory QIBM/ProdData/OS400/CollectionServices/lib. Java applications should include this file in their classpath. For more information about the Java implementation, download the javadocs in a .zip file.

Query the collection object in real-time

If your application needs to query the collection object in real-time, it will need to synchronize the queries with Collection Services. To do this, the application should create a data queue and register it with Collection Services. Once registered, the collector sends a notification for each collection interval and for the end of the collection cycle. The application should maintain the data queue, including removing the data queue when finished, and handling abnormal termination. To register and deregister the data queue, refer to the following APIs:

- Add collector notification: QypsAddCollectorNotification
- Remove collector notification: QypsRmvCollectorNotification

Related reference

QpmCloseMgtcol APIThe Close Management Collection Object (QpmCloseMgtcol) API closes a management collection object that was previously opened by the Open Management Collection Object (QpmOpenMgtcol) API.

QpmCloseMgtcolRepo APIThe Close Management Collection Object Repository (QpmCloseMgtcolRepo) API closes a repository of a management collection object that was previously opened by the Open Management Collection Object Repository (QpmOpenMgtcolRepo) API.

QpmOpenMgtcol APIThe Open Management Collection Object (QpmOpenMgtcol) API opens a specified management collection object for processing and returns a handle to the open management collection object.

QpmOpenMgtcolRepo APIThe Open Management Collection Object Repository (QpmOpenMgtcolRepo) API opens a specified repository of a management collection object for processing.

QpmReadMgtcolData APIThe Read Management Collection Object Data (QpmReadMgtcolData) API returns information about a specific record in a repository of a management collection object.

QpmRtvActiveMgtcolName APIThe Retrieve Active Management Collection Object Name (QpmRtvActiveMgtcolName) API returns the object name and library name of an active management collection object.

QpmRtvMgtcolAttrs APIThe Retrieve Management Collection Object Attributes (QpmRtvMgtcolAttrs) API returns information about attributes of a management collection object and repositories of a management collection object.

QypsAddCollectorNotification APIThe Add Collector Notification (QypsAddCollectorNotification) API registers with a collector to provide notifications to a specified data queue for a collection event.

QypsDeregCollectorDataCategory APIThe Deregister Collector Data Category (QypsDeregCollectorDataCategory) API removes a user-defined data category from Collection Services.

QypsRmvCollectorNotification APIThe Remove Collector Notification (QypsRmvCollectorNotification) API removes a notification registration from a collector for a specified data queue and collection event.

QypsRegCollectorDataCategory APIThe Register Collector Data Category (QypsRegCollectorDataCategory) API adds a user-defined data category to one or more collector definitions of Collection Services.

Collection program recommendations and requirements

Collection Services calls the data collection program once during the start of a collection cycle, once for each collection interval, and again at the end of the collection cycle.

The data collection program must perform any data collection and return that data to a data buffer provided by Collection Services. In addition to providing a data buffer, Collection Services also provides a work area, which allows the data collection program to maintain some state information between collection intervals.

The data collection program should collect data as quickly as possible and should perform minimal formatting. The program should not perform any data processing or sorting. Although data from the user-defined category is not converted into database files, Collection Services may run the Create Performance Data (CRTPFRDTA) command automatically and add the data from the collection object to database files at the end of each collection interval. If the data collection program cannot complete its tasks within the collection interval, the CRTPFRDTA command does not run properly.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 171.

You may create the data collection program in several environments:

- *PGM for OPM languages. This environment may not be used to query the collection object and may result in poor performance. However, it is supported for older programming languages.
- *SRVPGM, an entry point in a service program. This is for ILE languages.
- *JVAPGM, the required Java classes are included in ColSrv.jar. This file is included in the IFS at QIBM/ProdData/OS400/CollectionServices/lib. Download the javadocs.zip file and open index.html for a description of the Java implementations of the APIs.

Collection Services sends the following requests to the data collection program:

Request	Description
Start collection	The data collection program should initialize any interfaces or resources used during data collection. Optionally, it may also initialize a work area, provided by Collection Services, that preserves state information between collection intervals. If you want to include a control record prior to the collected data, the data collection program may also write a small amount of data to the data buffer. Typically, this control record would be used during data processing to help interpret the data.
Collection interval	Collection Services sends an interval request for each collection interval. The data collection program should collect data and return it in the data buffer. Collection Services then writes that data to the interval record in the collection object. If the amount of data is too large for the data buffer, the data collection program should set a "More data" flag. This action causes Collection Services to send another interval request with a modifier indicating that it is a continuation. Collection Services resets the more data flag before each call. This process is repeated until all the data is moved into the collection object.

Request	Description
End of collection	When the collection for the category containing the data collection program ends, Collection Services sends this request. The data collection program should perform any cleanup and can optionally return a collection control record. The data collection program should also send a return code that indicates the result of the collection.
Clean up and terminate (Shutdown)	Collection Services sends this request if an abnormal termination is necessary. Operating system resources are freed automatically when the data collection program job ends, but any other shutdown operations should be performed by the data collection program. The data collection program can receive this request at any time.

For a detailed description of these parameters, the work area, data buffer, and return codes, refer to the header file QPMDCPRM, which is located in QSYSINC.

Data storage in collection objects

Collection objects have a repository for each data collection category. This repository gets created by Collection Services when collections for that category are started. Each repository consists of the following records:

Record	Description
Control	This optional record can be the first or last record that results from the data collection program, and may occur in both positions. Typically, it should contain any information needed to interpret the record data.
Interval	Each collection interval creates an interval record, even if it is empty. The interval record contains the data written to the data buffer during the collection interval. It must not exceed 4 GB in size.
Stop	Collection Services automatically creates this record to indicate the end of a data collection session. If the collections for the user-defined category were restarted without ending or cycling Collection Services, you can optionally include a control record followed by additional interval records after the stop record.

Example: Implementing user-defined categories
Look here for sample programs that illustrate how you can use the provided APIs to integrate customized data collections into Collection Services.

Example: Data collection program
This program example collects some test data and stores it in a data buffer, which Collection Services copies to the collection object.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 171.

C++ sample code

```
#include "string.h"                     // memcpy(), memset(), strlen()
#include "stdio.h"                     // printf()
#include "qpmdcprm.h"                   // data collection program interface
#include "time.h"

extern "C"
void DCPentry( Qpm_DC_Parm_t *request, char *dataBuffer,
                                                        char *workArea, int *returnCode )
{
  static  char  testData[21] = "Just some test stuff";
```

```
  int           i;


/* Print contents of request structure */

  printf( "DCP called with parameters:\n" );
  printf( "  format name: \"%8.8s\";  category name: \"%10.10s\";\n",
          request->formatName, request->categoryName );
  printf( "  rsvd1: %4.4X; req type: %d; req mod: %d; buffer len: %d;\n",
          *(short *)(request->rsvd1), request->requestType,
          request->requestModifier, request->dataBufferLength);
  printf( "  prm offset: %d; prm len: %d; work len: %d; rsvd2: %8.8X;\n",
          request->parmOffset, request->parmLength, request->workAreaLength,
          *(int *)(request->rsvd2) );
  printf( "  rec key: \"%8.8s\"; timestamp: %8.8X %8.8X;\n",
          request->intervalKey,
          *(int *)(request->intervalTimestamp),
          *(int *)(request->intervalTimestamp + 4) );
  printf( "  return len: %d; more data: %d; rsvd3: %8.8X %8.8X;\n",
          request->bytesProvided, request->moreData,
          *(int *)(request->rsvd3),
          *(int *)(request->rsvd3 + 4) );

  switch ( request->requestType )
  {
  /* Write control record in the beginning of collection */
    case PM_DOBEGIN:
      printf( "doBegin(%d)\n", request->requestModifier );
      switch ( request->requestModifier)
      {
        case PM_CALL_NORMAL:
            memcpy( dataBuffer, testData, 20 );
            *(int *)workArea = 20;
            request->moreData = PM_MORE_DATA;
            request->bytesProvided = 20;
          break;

        case PM_CALL_CONTINUE:
          if ( *(int *)workArea < 200 )
          {
            memcpy( dataBuffer, testData, 20 );
            *(int *)workArea += 20;
            request->moreData = PM_MORE_DATA;
            request->bytesProvided = 20;
          }
          else
          {
            *(int *)workArea = 0;
            request->moreData = PM_NO_MORE_DATA;
            request->bytesProvided = 0;
          }
          break;

        default:
          *returnCode = -1;
          return;
      }
      break;
 /* Write control record in the end of collection */
    case PM_DOEND:
      printf( "doEnd(%d)\n", request->requestModifier );
      switch ( request->requestModifier)
      {
        case PM_CALL_NORMAL:
            memcpy( dataBuffer, testData, 20 );
            *(int *)workArea = 20;
            request->moreData = PM_MORE_DATA;
            request->bytesProvided = 20;
          break;

        case PM_CALL_CONTINUE:
          if ( *(int *)workArea < 200 )
          {
            memcpy( dataBuffer, testData, 20 );
            *(int *)workArea += 20;
            request->moreData = PM_MORE_DATA;
            request->bytesProvided = 20;
          }
          else
          {
            *(int *)workArea = 0;
```

```
            request->moreData = PM_NO_MORE_DATA;
            request->bytesProvided = 0;
          }
          break;

        default:
          *returnCode = -1;
          return;
      }
      break;

  /*Write interval record */
     case PM_DOCOLLECT:
      printf( "doCollect(%d)\n", request->requestModifier );
      for ( i = 0; i < 10000; i++ )
        dataBuffer[i] = i % 256;
      request->bytesProvided = 10000;

      switch ( request->requestModifier)
      {
        case PM_CALL_NORMAL:
            *(time_t *)(workArea + 4) = time( NULL );
            *(int *)workArea = 1;
            request->moreData = PM_MORE_DATA;
          break;

        case PM_CALL_CONTINUE:
          *(int *)workArea += 1;
          if ( *(int *)workArea < 20 )
            request->moreData = PM_MORE_DATA;
          else
          {
            *(time_t *)(workArea + 8) = time( NULL );
            printf( "doCollect() complete in %d secs (%d bytes transferred)\n",
                    *(time_t *)(workArea + 8) - *(time_t *)(workArea + 4), 10000 * 20 );
            request->moreData = PM_NO_MORE_DATA;
          }
          break;

        default:
          *returnCode = -1;
          return;
      }
      break;
  /* Clean-up and terminate */
     case PM_DOSHUTDOWN:
      printf( "doShutdown\n" );
      *returnCode = 0;
      return;
      break;

     default:
      *returnCode = -1;
      return;
      break;
   }

 }/* DCPentry() */
```

Related concepts

Collection program recommendations and requirements

Collection Services calls the data collection program once during the start of a collection cycle, once for each collection interval, and again at the end of the collection cycle.

Example: Program to register the data collection program

This sample program registers the data collection program from the previous example with Collection Services. After running, Collection Services displays the data collection program in the list of data collection categories.

Note: By using the code examples, you agree to the terms of the “Code license and disclaimer information” on page 171.

C++ sample code

```
#include "stdlib.h"
#include "stdio.h"
#include "string.h"
#include "qypscoll.h"


int main( int argc, char *argv[] )
{
    int    CCSID = 0;
    int    RC  = 0;
    Qyps_USER_CAT_PROGRAM_ATTR    *pgmAttr;
    Qyps_USER_CAT_ATTR             catAttr;
    char   collectorName[11] = "*PFR      ";
    char   categoryName[11]  = "TESTCAT   ";
    char   collectorDefn[11] = "*CUSTOM   ";  /* Register to *CUSTOM profile only */

      if ( argc > 2 )
      {
        int len = strlen( argv[2] );

        if ( len > 10 ) len = 10;
        memset( categoryName, ' ', 10 );
        memcpy( categoryName, argv[2], len );
      }

      if ( argc < 2 || *argv[1] == 'R' )
      {
        pgmAttr = (Qyps_USER_CAT_PROGRAM_ATTR *)malloc( 4096 );
        memset( pgmAttr, 0x00, sizeof(pgmAttr) );
        pgmAttr->fixedPortionSize = sizeof( Qyps_USER_CAT_PROGRAM_ATTR );
        memcpy( pgmAttr->programType,     "*SRVPGM   ", 10 );
        memcpy( pgmAttr->parameterFormat, "PMDC0100", 8 );
        memcpy( pgmAttr->ownerUserId,     "USERID    ", 10 );
        memcpy( pgmAttr->jobDescription,  "QPMUSRCAT QGPL      ", 20 );
        memcpy( pgmAttr->qualPgmSrvpgmName, "DCPTEST   LIBRARY    ", 20 );
        pgmAttr->workAreaSize = 123;
        pgmAttr->srvpgmEntrypointOffset = pgmAttr->fixedPortionSize;
        pgmAttr->srvpgmEntrypointLength = 8;
        pgmAttr->categoryParameterOffset = pgmAttr->srvpgmEntrypointOffset +
                                           pgmAttr->srvpgmEntrypointLength;
        pgmAttr->categoryParameterLength = 10;
   /* Set entry point name */
        memcpy( (char *)(pgmAttr) + pgmAttr->srvpgmEntrypointOffset,
                "DCPentry", pgmAttr->srvpgmEntrypointLength );  /* Set parameter string */
        memcpy( (char *)(pgmAttr) + pgmAttr->categoryParameterOffset,
                "1234567890", pgmAttr->categoryParameterLength );

        memset( &catAttr, 0x00, sizeof(catAttr) );
        catAttr.structureSize = sizeof( Qyps_USER_CAT_ATTR );
        catAttr.minCollectionInterval = 0;
        catAttr.maxCollectionInterval = 0;
        catAttr.defaultCollectionInterval = 30;   /* Collect at 30 second interval */
        memset( catAttr.qualifiedMsgId, ' ', sizeof(catAttr.qualifiedMsgId) );
        memcpy( catAttr.categoryDesc,
                "1234567890123456789012345678901234567890123456789012345678901234567890",
sizeof(catAttr.categoryDesc) );

        QypsRegCollectorDataCategory( collectorName,
                                      categoryName,
                                      collectorDefn,
                                      &CCSID,
                                      (char*)pgmAttr,
                                      (char*)&catAttr,
                                      &RC
                                      );
      }
      else
      if( argc >= 2 && *argv[1] == 'D' )
        QypsDeregCollectorDataCategory( collectorName, categoryName, &RC );
      else
        printf("Unrecognized option\n");

}/* main() */
```

Example: Program to query the collection object
This sample program illustrates how to query the data stored in the collection object using the Java classes shipped in the ColSrv.jar file in the `QIBM/ProdData/OS400/CollectionServices/lib` directory path.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 171.

Java sample code

```
import com.ibm.iseries.collectionservices.*;

class testmco2
{
  public static void main( String argv[] )
  {
    String    objectName = null;
    String    libraryName = null;
    String    repoName = null;
    MgtcolObj mco = null;
    int       repoHandle = 0;
    int       argc = argv.length;
    MgtcolObjAttributes
              attr = null;
    MgtcolObjRepositoryEntry
              repoE = null;
    MgtcolObjCollectionEntry
              collE = null;
    int       i,j;

    if ( argc <  3 )
    {
      System.out.println("testmco2  objectName libraryName repoName");
      System.exit(1);
    }

    objectName  = argv[0];
    libraryName = argv[1];
    repoName    = argv[2];

    if ( ! objectName.equals( "*ACTIVE" ) )
      mco = new MgtcolObj( objectName, libraryName );
    else
      try
      {
        mco = MgtcolObj.rtvActive();
      } catch ( Exception e)
      {
        System.out.println("rtvActive(): Exception " + e );
        System.exit(1);
      }
    System.out.println("Object name = " + mco.getName() );
    System.out.println("Library name = " + mco.getLibrary() );

    try
    {
      attr = mco.rtvAttributes( "MCOA0100" );
    } catch ( Exception e)
    {
      System.out.println("rtvAttributes(): MCOA0100: Exception " +
e );
      System.exit(1);
    }

    System.out.println("MCOA0100:  Object " + mco.getLibrary() + "/" + mco.getName() );
    System.out.println("   size = " + attr.size + " retention = " + attr.retentionPeriod +
                       " interval = " + attr.dftInterval + " time created = " +
attr.timeCreated +
                       " time updated = " + attr.timeUpdated );
    System.out.println("   serial = " + attr.logicalPSN + " active = " + attr.isActive +
                       " repaired = " + attr.isRepaired + " summary = " + attr.sumStatus +
                       " repo count = " + attr.repositoryCount );
    if ( attr.repositoryInfo != null )
      for(i = 0; i < attr.repositoryCount; i++ )
      {
        repoE = attr.repositoryInfo[ i ];
        System.out.println("      name = " + repoE.name + " category = " + repoE.categoryName +
             " size = " + repoE.size );
```

```
        for( j = 0; j < repoE.collectionInfo.length; j++ )
        {
          collE = repoE.collectionInfo[ j ];
          System.out.println("         startTime = " + collE.startTime + " endTime = " +
collE.endTime +
                " interval = " + collE.interval );
        }
      }

    try
    {
      attr = mco.rtvAttributes( "MCOA0200" );
    } catch ( Exception e)
    {
      System.out.println("rtvAttributes(): MCOA0200: Exception " + e );
      System.exit(1);
    }

    System.out.println("MCOA0200: Object " + mco.getLibrary() + "/" + mco.getName() );
    System.out.println("   size = " + attr.size + " retention = " + attr.retentionPeriod +
                       " interval = " + attr.dftInterval + " time created = " +
attr.timeCreated +
                       " time updated = " + attr.timeUpdated );
    System.out.println("   serial = " + attr.logicalPSN + " active = " + attr.isActive +
                       " repaired = " + attr.isRepaired + " summary = " + attr.sumStatus +
                       " repo count = " + attr.repositoryCount );
    if ( attr.repositoryInfo != null )
      for(i = 0; i < attr.repositoryCount; i++ )
      {
 repoE = attr.repositoryInfo[ i ];
 System.out.println("      name = " + repoE.name + " category = " + repoE.categoryName +
      " size = " + repoE.size );
 for( j = 0; j < repoE.collectionInfo.length; j++ )
 {
   collE = repoE.collectionInfo[ j ];
   System.out.println("         startTime = " + collE.startTime + " endTime = " + collE.endTime
+
        " interval = " + collE.interval );
 }
      }

    if ( repoName.equals("NONE") )
        return;

    try
    {
      mco.open();
    } catch ( Exception e)
    {
      System.out.println("open(): Exception " + e );
      System.exit(1);
    }

    try
    {
      repoHandle = mco.openRepository( repoName, "MCOD0100" );
    } catch ( Exception e)
    {
      System.out.println("openRepository(): Exception " + e );
      mco.close();
      System.exit(1);
    }
    System.out.println("repoHandle = " + repoHandle );

    MgtcolObjReadOptions  readOptions = new MgtcolObjReadOptions();
    MgtcolObjRecInfo recInfo = new MgtcolObjRecInfo();

    readOptions.option = MgtcolObjReadOptions.READ_NEXT;
    readOptions.recKey = null;
    readOptions.offset = 0;
    readOptions.length = 0;

    while ( recInfo.recStatus == MgtcolObjRecInfo.RECORD_OK )
    {
      try
      {
        mco.readData( repoHandle, readOptions, recInfo, null );
      } catch ( Exception e)
      {
        System.out.println("readData(): Exception " + e );
        mco.close();
        System.exit(1);
```

```
      }

      if( recInfo.recStatus == MgtcolObjRecInfo.RECORD_OK )
      {
        System.out.print("Type = " + recInfo.recType );
        System.out.print(" Key = " + recInfo.recKey );
        System.out.println(" Length = " + recInfo.recLength );
      }

    }/* while ... */

    mco.closeRepository( repoHandle );
    mco.close();

  }/* main() */

}/* class testmco2 */
```

Managing collection objects

When you use Collection Services to collect performance data, each collection is stored in a single object.

You can delete a collection object from the system. If you do not delete the objects manually, Collection Services will delete them automatically after the expiration date and time.

Collection Services deletes the cycled collection objects that have reached their expiration date and time the next time it starts or cycles a collection. The expiration date is associated with the management collection object. Collection Services deletes only expired management collection objects that exist in the configured collection library.

The expiration date for each management collection object is shown in the Properties for that collection object. To keep the object on the system longer, you simply change the date on the Properties page. You can specify **Permanent** if you do not want Collection Services to delete your management collection objects for you.

Related tasks

Manage collections

Select the **Manage Collections** task to launch the collections table to view and work with the collections on your system. The collections from Collection Services, IBM i Job Watcher, IBM i Disk Watcher, Performance Explorer, and Batch Model are visible here.

User-defined transactions

Collection Services and Performance Explorer collect performance data that you define in your applications.

With the provided APIs, you can integrate transaction data into the regularly scheduled sample data collections using Collection Services, and get trace-level data about your transaction by running Performance Explorer.

For detailed descriptions and usage notes, refer to the following API descriptions:

- Start Transaction (QYPESTRT, qypeStartTransaction) API
- End transaction (QYPEENDT, qypeEndTransaction) API
- Log transaction (QYPELOGT, qypeLogTransaction) API (Used only by Performance Explorer)
- Add trace point (QYPEADDT, qypeAddTracePoint) API (Used only by Performance Explorer)

Note: You only need to instrument your application once. Collection Services and Performance Explorer use the same API calls to gather different types of performance data.

Integrating user-defined transaction data into Collection Services

You can select user-defined transactions as a category for collection in the Collection Services configuration. Collection Services then collects the transaction data at every collection interval and stores that data in the collection object. The Create Performance Data (CRTPFRDTA) command exports this data to the user-defined transaction performance database file, QAPMUSRTNS. Collection Services organizes

the data by transaction type. You can specify as many transaction types as you require; however, Collection Services will only report the first 15 transaction types. Data for additional transaction types is combined and stored as the *OTHER transaction type. At every collection interval, Collection Services creates one record for each type of transaction for each unique job. For a detailed description, refer to the usage notes in the Start Transaction API.

Collection Services gathers general transaction data, such as the transaction response time. You can also include up to 16 optional application-defined counters that can track application-specific data like the number of SQL statements used for the transaction, or other incremental measurements. Your application should use the Start Transaction API to indicate the beginning of a new transaction, and should include a corresponding End Transaction API to deliver the transaction data to Collection Services.

Collecting trace information for user-defined transactions with Performance Explorer

You can use the Start, End, and Log Transaction APIs during a Performance Explorer session to create a trace record. Performance Explorer stores system resource utilization, such as CPU utilization, I/O, and seize/lock activity, for the current thread in these trace records. Additionally, you may choose to include application-specific performance data, and then send it to Performance Explorer in each of these APIs. You can also use the Add Trace Point API to identify application-specific events for which Performance Explorer should collect trace data.

To start a Performance Explorer session for your transactions, specify *USRTRNS on the (OSEVT) parameter of your Performance Explorer definition. After entering the ENDPEX command, Performance Explorer writes the data supplied by the application to the QMUDTA field in the QAYPEMIUSR Performance Explorer database file. System-supplied performance data for the start, end, and any log records is stored in the QAYPEMIUSR and QAYPETIDX database files.

Related concepts

Performance Explorer
Performance Explorer collects more detailed information about a specific application, program or system resource, and provides detailed insight into a specific performance problem. This includes the capability both to perform several types and levels of traces and to run detailed reports.

Related reference

QYPESTRT, qypeStartTransaction APIThe Start Transaction (QYPESTRT, qypeStartTransaction) API is used together with the End Transaction (QYPEENDT, qypeEndTransaction) API and the Log Transaction (QYPELOGT, qypeLogTransaction) API to collect performance data for user-defined transactions. The Start Transaction API is called by an application at the beginning of a user-defined transaction.

QYPEENDT, qypeEndTransaction APIThe End Transaction (QYPEENDT, qypeEndTransaction) API is used together with the Start Transaction (QYPESTRT, qypeStartTransaction) API and the Log Transaction (QYPELOGT, qypeLogTransaction) API to collect performance data for user-defined transactions. The End Transaction API is called by an application at the end of a user-defined transaction.

QYPELOGT, qypeLogTransaction APIThe Log Transaction (QYPELOGT, qypeLogTransaction) API is used together with the Start Transaction (QYPESTRT, qypeStartTransaction) API and the End Transaction (QYPEENDT, qypeEndTransaction) API to collect performance data for user-defined transactions. The Log Transaction API is called by an application any time between the calls to the Start Transaction API and the End Transaction API to trace the progress of a user-defined transaction.

QYPEADDT, qypeAddTracePoint APIThe Add Trace Point (QYPEADDT, qypeAddTracePoint) API is used to record application-defined trace data.

Collection Services data files: QAPMUSRTNSThis Collection Services database file contains performance data for the user-defined and Application Response Measurement (ARM) transactions.

Create Performance Data (CRTPFRDTA) commandSee the Create Performance Data (CRTPFRDTA) command for information about creating performance database files.

C++ example: Integrating user-defined transactions into Collection Services

This C++ example program shows how to use the Start Transaction and End Transaction APIs to integrate user-defined transaction performance data into Collection Services.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 171.

```
//*********************************************************************
// tnstst.C
//
// This example program illustrates the use
// of the Start/End Transaction APIs (qypeStartTransaction,
// qypeEndTransaction).
//
//
// This program can be invoked as follows:
//   CALL lib/TNSTST PARM('threads' 'types' 'transactions' 'delay')
//     where
//       threads      = number of threads to create (10000 max)
//       types        = number of transaction types for each thread
//       transactions = number of transactions for each transaction
//                      type
//       delay        = delay time (millisecs) between starting and
//                      ending the transaction
//
// This program will create "threads" number of threads. Each thread
// will generate transactions in the same way. A thread will do
// "transactions" number of transactions for each transaction type,
// where a transaction is defined as a call to Start Transaction API,
// then a delay of  "delay" millisecs, then a call to End Transaction
// API. Thus, each thread will do a total of "transactions" * "types"
// number of transactions. Each transaction type will be named
// "TRANSACTION_TYPE_nnn" where nnn ranges from 001 to "types". For
// transaction type n, there will be  n-1 (16 max) user-provided
// counters reported, with counter m reporting m counts for each
// transaction.
//
// This program must be run in a job that allows multiple threads
// (interactive jobs typically do not allow multiple threads). One
// way to do this is to invoke the program using the SBMJOB command
// specifying ALWMLTTHD(*YES).
//
//*********************************************************************

#define _MULTI_THREADED

// Includes
#include "pthread.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "qusec.h"
#include "lbcpynv.h"
#include "qypesvpg.h"

// Constants
#define maxThreads 10000

// Transaction pgm parm structure
typedef struct
{
  int types;
  int trans;
  int delay;
} tnsPgmParm_t;

// Error code structure
typedef struct
{
  Qus_EC_t error;
  char     Exception_Data[100];
} error_code_t;


//*********************************************************************
//
// Transaction program to run in each secondary thread
//
```

```
//*********************************************************************

void *tnsPgm(void *parm)
{
  tnsPgmParm_t *p = (tnsPgmParm_t *)parm;

  char tnsTyp[] = "TRANSACTION_TYPE_XXX";
  char pexData[] = "PEX";
  unsigned int pexDataL = sizeof(pexData) - 1;
  unsigned long long colSrvData[16] = {1,2,3,4,5,6,7,8,
                                       9,10,11,12,13,14,15,16};
  unsigned int colSrvDataL;
  char tnsStrTim[8];

  struct timespec ts = {0, 0};

  error_code_t errCode;

  _DPA_Template_T target, source; // Used for LBCPYNV MI instr

  unsigned int typCnt;
  unsigned int tnsCnt;
  int rc;


  // Initialize error code
  memset(&errCode, 0, sizeof(errCode));
  errCode.error.Bytes_Provided = sizeof(errCode);

  // Initialize delay time
  ts.tv_sec  = p->delay / 1000;
  ts.tv_nsec = (p->delay % 1000) * 1000000;

  // Loop doing transactions
  for (tnsCnt = 1; tnsCnt <= p->trans; tnsCnt++)
  {
    for (typCnt = 1; typCnt <= p->types; typCnt++)
    {
      // Set number field in transaction type
      source.Type = _T_UNSIGNED;
      source.Length = 4;
      source.reserved = 0;
      target.Type = _T_ZONED;
      target.Length = 3;
      target.reserved = 0;
      _LBCPYNV(tnsTyp + 17, &target, &typCnt, &source);

      // Set Coll Svcs data length in bytes
      colSrvDataL = (typCnt <= 16) ? (typCnt - 1) : 16;
      colSrvDataL = colSrvDataL * 8;

      // Call Start Transaction API
      qypeStartTransaction(tnsTyp,
                           (unsigned int *)&tnsCnt,
                           pexData,
                           (unsigned int *)&pexDataL,
                           tnsStrTim,
                           &errCode);

      // Delay specified amount
      rc = pthread_delay_np(&ts);

      // Call End Transaction API
      qypeEndTransaction(tnsTyp,
                         (unsigned int *)&tnsCnt,
                         pexData,
                         (unsigned int *)&pexDataL,
                         tnsStrTim,
                         (unsigned long long *)&colSrvData[0],
                         (unsigned int *)&colSrvDataL,
                         &errCode);
    }
  }

  return NULL;
}


//*********************************************************************
//
// Main program to run in primary thread
//
```

```
//*********************************************************************

void main(int argc, char *argv[])
{
  // Integer version of parms
  int threads;  // # of threads
  int types;    // # of types
  int trans;    // # of transactions
  int delay;    // Delay in millisecs

  pthread_t threadHandle[maxThreads];
  tnsPgmParm_t tnsPgmParm;
  int rc;
  int i;


  // Verify 4 parms passed
  if (argc != 5)
  {
    printf("Did not pass 4 parms\n");
    return;
  }

  // Copy parms into integer variables
  threads = atoi(argv[1]);
  types   = atoi(argv[2]);
  trans   = atoi(argv[3]);
  delay   = atoi(argv[4]);

  // Verify parms
  if (threads > maxThreads)
  {
    printf("Too many threads requested\n");
    return;
  }

  // Initialize transaction pgm parms (do not modify
  // these while threads are running)
  tnsPgmParm.types = types;
  tnsPgmParm.trans = trans;
  tnsPgmParm.delay = delay;

  // Create threads that will run transaction pgm
  for (i=0; i < threads; i++)
  {
    // Clear thread handle
    memset(&threadHandle[i], 0, sizeof(pthread_t));
    // Create thread
    rc = pthread_create(&threadHandle[i],      // Thread handle
                        NULL,                  // Default attributes
                        tnsPgm,                // Start routine
                        (void *)&tnsPgmParm);  // Start routine parms
    if (rc != 0)
      printf("pthread_create() failed, rc = %d\n", rc);
  }

  // Wait for each thread to terminate
  for (i=0; i < threads; i++)
  {
    rc=pthread_join(threadHandle[i],  // Thread handle
                    NULL);            // No exit status
  }

}  /* end of Main */
```

Java example: Integrating user-defined transactions into Collection Services

This Java example program shows how to use the Start Transaction and End Transaction APIs to integrate user-defined transaction performance data into Collection Services.

Note: By using the code examples, you agree to the terms of the "Code license and disclaimer information" on page 171.

```
import com.ibm.iseries.collectionservices.PerformanceDataReporter;


// parameters:
//   number of TXs per thread
```

```
//   number of threads
//   log|nolog
//   enable|disable
//   transaction seconds

public class TestTXApi
{
  static TestTXApiThread[]    thread;

  static private String[] TxTypeString;
  static private byte[][] TxTypeArray;

  static private String   TxEventString;
  static private byte[]   TxEventArray;

  static
  {
    int i;

    // initialize transaction type strings and byte arrays

      TxTypeString = new String[20];
      TxTypeString[ 0] = "Transaction type  00";
      TxTypeString[ 1] = "Transaction type  01";
      TxTypeString[ 2] = "Transaction type  02";
      TxTypeString[ 3] = "Transaction type  03";
      TxTypeString[ 4] = "Transaction type  04";
      TxTypeString[ 5] = "Transaction type  05";
      TxTypeString[ 6] = "Transaction type  06";
      TxTypeString[ 7] = "Transaction type  07";
      TxTypeString[ 8] = "Transaction type  08";
      TxTypeString[ 9] = "Transaction type  09";
      TxTypeString[10] = "Transaction type  10";
      TxTypeString[11] = "Transaction type  11";
      TxTypeString[12] = "Transaction type  12";
      TxTypeString[13] = "Transaction type  13";
      TxTypeString[14] = "Transaction type  14";
      TxTypeString[15] = "Transaction type  15";
      TxTypeString[16] = "Transaction type  16";
      TxTypeString[17] = "Transaction type  17";
      TxTypeString[18] = "Transaction type  18";
      TxTypeString[19] = "Transaction type  19";

      TxTypeArray = new byte[20][];
      for ( i = 0; i < 20; i++ )
        try
        {
          TxTypeArray[i] = TxTypeString[i].getBytes("Cp037");
        } catch(Exception e)
        {
          System.out.println("Exception \"" + e + "\" when converting");
        }

  }/* static */



  public static void main( String[] args )
  {
    int     numberOfTXPerThread;
    int     numberOfThreads;
    boolean log;
    boolean enable;
    int     secsToDelay;

      // process parameters
      if ( args.length >= 5 )
 try
        {
          numberOfTXPerThread = Integer.parseInt( args[0] );
          numberOfThreads     = Integer.parseInt( args[1] );

          if ( args[2].equalsIgnoreCase( "log" ) )
     log = true;
          else
          if ( args[2].equalsIgnoreCase( "nolog" ) )
            log = false;
          else
          {
            System.out.println( "Wrong value for 3rd parameter!" );
            System.out.println( "\tshould be log|nolog" );
            return;
```

```
          }

          if ( args[3].equalsIgnoreCase( "enable" ) )
     enable = true;
          else
          if ( args[3].equalsIgnoreCase( "disable" ) )
            enable = false;
          else
          {
            System.out.println( "Wrong value for 4th parameter!" );
            System.out.println( "\tshould be enable|disable" );
            return;
          }

          secsToDelay = Integer.parseInt( args[4] );

        } catch (Exception e)
        {
          System.out.println( "Oops! Cannot process parameters!" );
          return;
        }
      else
      {
        System.out.println( "Incorrect Usage." );
        System.out.println( "The correct usage is:" );
        System.out.println( "java TestTXApi numberOfTXPerThread numberOfThreads
    log|nolog enable|disable secsToDelay");
        System.out.println("\tlog will make the program cut 1 log transaction per start / end
pair");
        System.out.println("\tdisable will disable performance collection to minimize
overhead");
        System.out.print("\nExample: \"java TestTXApi 10000 100 log enable 3\" will call " );
        System.out.println("cause 10000 transactions for each of 100 threads");
        System.out.println("with 3 seconds between start and end of transaction");
        System.out.println("Plus it will place additional log call and will enable
reporting." );
        return;
      }

      System.out.println( "Parameters are processed:" );
      System.out.println( "\tnumberOfTxPerThread = " + numberOfTXPerThread );
      System.out.println( "\tnumberOfThreads = " + numberOfThreads );
      System.out.println( "\tlog = " + log );
      System.out.println( "\tenable = " + enable );
      System.out.println( "\tsecsToDelay = " + secsToDelay );

    // cause initialization of a PerformanceDataReporter class
      {
        PerformanceDataReporter pReporter = new PerformanceDataReporter();
 pReporter.enableReporting();
      }

    TestTXApi t = new TestTXApi( );

      System.out.println( "\nAbout to start ..." );
      t.prepareTests( numberOfTXPerThread, numberOfThreads, log, enable, secsToDelay );

    long startTime = System.currentTimeMillis();

      t.runTests( numberOfThreads );

      // wait for threads to complete
      for ( int i = 0; i < numberOfThreads; i++ )
        try
        {
          thread[i].join( );
        } catch(Exception e)
        {
          System.out.println( "***Exception \"" + e + "\" while joining thread " + i );
        }

    long endTime = System.currentTimeMillis();

      System.out.println( "\nTest runtime for " + ( numberOfTXPerThread * numberOfThreads) +
                          " TXs was " + ( endTime - startTime ) + " msec" );

  }/* main() */

  private void prepareTests( int numberOfTxPerThread,
                             int numberOfThreads, boolean log,
boolean enable, int secsToDelay )
  {
```

```
    System.out.println( "Creating " + numberOfThreads + " threads");
    thread = new TestTXApiThread[numberOfThreads];
    for ( int i = 0; i < numberOfThreads; i++ )
      thread[i] = new TestTXApiThread( i, numberOfTxPerThread,
                                       log, enable, secsToDelay );

  }/* prepareTests() */

  private void runTests( int numberOfThreads )
  {
    for ( int i = 0; i < numberOfThreads; i++ )
      thread[i].start( );

  }/* runTests() */

  private class TestTXApiThread extends Thread
  {
    private int     ordinal;
    private int     numberOfTxPerThread;
    private boolean log;
    private boolean enable;
    private int     secsToDelay;

    private PerformanceDataReporter     pReporter;

    private long    timeStamp[];
    private long    userCounters[];


      public TestTXApiThread( int ordinal, int numberOfTxPerThread,
                              boolean log, boolean enable, int secsToDelay )
      {
        super();
        this.ordinal             = ordinal;
        this.numberOfTxPerThread = numberOfTxPerThread;
        this.log                 = log;
        this.enable              = enable;
        this.secsToDelay         = secsToDelay;

          pReporter = new PerformanceDataReporter( false );
          if ( enable )
            pReporter.enableReporting();
          timeStamp = new long[1];
          userCounters = new long[16];
          for ( int i = 0; i < 16; i++ )
              userCounters[i] = i;

      }/* constructor */

      public void run()
      {
        int i;

              for ( i = 0; i < numberOfTxPerThread; i++ )
              {
                pReporter.startTransaction( TxTypeArray[i%20], i, TxTypeArray[i%20], 20,
timeStamp );
//                pReporter.startTransaction( TxTypeArray[i%20], i, TxTypeString[i%20],
timeStamp );
                if ( log )
                  pReporter.logTransaction( TxTypeArray[i%20], i, TxTypeArray[i%20], 20 );
//                  pReporter.logTransaction( TxTypeArray[i%20], i, TxTypeString[i%20] );
                if (secsToDelay > 0)
                  try
                  {
                    Thread.sleep(secsToDelay * 1000);
                  } catch(Exception e) { }
                pReporter.endTransaction( TxTypeArray[i%20], i, TxTypeArray[i%20], 20,
timeStamp,
                                          userCounters );
//                pReporter.endTransaction( TxTypeArray[i%20], i, TxTypeString[i%20], timeStamp,
//                                          userCounters );
              }

      }/* run() */

 }/* class TestTXApiThread */

}/* class TestTXApi */
```

Finding wait statistics for a job, task, or thread

During the running of a job, task, or thread, conditions arise that cause that process to wait (for example, while the system resolves a lock or hold on a required object).

Collection Services can collect data on the cause and duration of the time a process spends waiting. This data is reported in the Collection Services database files QAPMJOBWT and QAPMJOBWTD.

Note: To query the QAPMJOBWTD file, the CCSID of your job must be set to the CCSID of the primary language installed on the system (not to 65535 binary data).

Another tool that shows job wait statistics is IBM i Job Watcher. IBM i Job Watcher returns real-time information about a selected set of jobs, threads, and Licensed Internal Code (LIC) program tasks. At specified time intervals, IBM i Job Watcher samples anywhere from one thread per job to all threads per job. IBM i Job Watcher gathers a variety of performance data, including detailed wait statistics for jobs, tasks, and threads.

There are 32 wait buckets which accumulate wait state data. These static wait buckets, used by both Collection Services and IBM i Job Watcher, provide a stable view of the wait state data. In Collection Services, data from these buckets is reported in files QAPMJOBWT and QAPMJOBWTG. In Job Watcher, data from these buckets is reported in QAPYJWTDE and QAPYJWSTS.

Related concepts

A job's lifeTo understand the basics of IBM i work management, follow a simple batch job as it moves through the system.

IBM i Job Watcher
IBM i Job Watcher provides for the collection of job data for any or all jobs, threads, and tasks on the system. It provides call stacks, SQL statements, objects being waited on, Java JVM statistics, wait statistics and more which are used to diagnose job related performance problems.

The basics of IBM i Wait Accounting
Wait Accounting is the patented technology built into the IBM i operating system that tells you what a thread or task is doing when it appears that it is not doing anything.

Related reference

Collection Services data files: QAPMJOBWTThis Collection Services database file contains information about job, task, and thread wait conditions.

Collection Services data files: QAPMJOBWTDThis Collection Services database file contains a description of the counter sets found in file QAPMJOBWT.

Collection Services data files: QAPMJOBWTGThis Collection Services database file contains information about job, task, and thread current wait conditions that is not available in the QAPMJOBWT file.

Job AccountingThis experience report provides a summary of information to make it easier to determine which Work Management interfaces to use when dealing with job attributes.

Understanding disk consumption by Collection Services

The amount of disk resource Collection Services consumed varies greatly depending on the settings that you use.

For illustration purposes, assume that Collection Services is used daily and cycles at midnight, causing each *MGTCOL object to contain one day's worth of collected data. Next, establish a base size for one day's worth of collected data by using the default properties for Collection Services. A standard plus protocol profile with an interval value of 15 minutes can collect 500 MB of data in the *MGTCOL object. The size actually collected per day using the default properties can vary greatly depending on system size and usage. The 500 MB example might represent a higher-end system that is heavily used.

Interval rate	Intervals per collection	Multiplier	Size in MB
15 minutes	96	1	500

The size of one day's worth of data is directly proportional to the number of intervals collected per collection period. For example, changing the interval rate from 15 minutes to 5 minutes increases the number of intervals by a factor of 3 and increases the size by the same factor.

Interval rate	Intervals per collection	Multiplier	Size in MB
15 minutes	96	1	500
5 minutes	288	3	1500

To continue this example, the following table shows the size of one *MGTCOL object produced each day by Collection Services at each interval rate, using the default standard plus protocol profile.

Interval rate	Intervals per collection	Multiplier	Size in MB
15 minutes	96	1	500
5 minutes	288	3	1500
1 minutes	1440	15	7500
30 seconds	2880	30	15000
15 Seconds	5760	60	30000

The size of the *MGTCOL object, in this example, can vary from 500 MB to 30 GB depending on the rate of collection. You can predict a specific system's disk consumption for one day's collection interval through actual observation of the size of the *MGTCOL objects created, using the default collection interval of 15 minutes and the standard plus protocol profile as the base and then using the multiplier from the above table to determine the disk consumption at other collection intervals. For example, if observation of the *MGTCOL object size reveals that the size of the object for a day's collection is 50 MB for 15-minute intervals, then you could expect Collection Services to produce *MGTCOL objects with a size of 3 GB when collecting data at 15-second intervals.

Note: Use caution when considering a collection interval as frequent as 15 seconds. Frequent collection intervals can adversely impact system performance.

Retention period

The retention period also plays a significant role in the amount of disk resource that Collection Services consumes. The default retention period is one day. However, practically speaking, given the default values, the *MGTCOL object is deleted on the third day of collection past the day on which it was created. Thus, on the third day of collection there is two days' worth of previously collected data plus the current day's data on the system. Using the table above, this translates into having between 1 GB and 1.5 GB of disk consumption at 15-minute intervals, and 60 to 90 GB of disk consumption at 15-second intervals on the system during the third day and beyond.

The formula to calculate disk consumption based on the retention period value is:

```
(Retention period in days + 2.5) * Size of one day's collection =
Total Disk Consumption
```

Note: 2.5 corresponds to two days of previous collection data, and an average of the current day (2 days + 1/2 day).

Using the above tables and formula, a retention period of 2 weeks gives you a disk consumption of 8.25 GB at 15-minute intervals and 495 GB at 15-second intervals for the example system.

It is important to understand the disk consumption by Collection Services to know the acceptable collection interval and retention period for a given system. Knowing this can ensure that disk consumption will not cause system problems. Remember to consider that a system monitor or a job monitor can override a category's collection interval to graph data for a monitor. A system administrator must ensure that monitors do not inadvertently collect data at intervals that cause excess data consumption.

Collecting and displaying CPU utilization for all partitions

When using multiple partitions, it can be important to understand the overall utilization of processing capability, across all partitions, regardless of whether the partition is running IBM i, AIX®, or Linux®. IBM i provides a way to collect and display this data.

Collecting the data

To collect CPU utilization for the physical system, you must meet the following configuration requirements:

- POWER6® hardware with IBM i 6.1 and a firmware level of xx340_061 or later.
- Enabled the collection of performance data for the partition you want it collected on. You only need to collect this data on one partition, and this partition must be an IBM i partition. The CPU utilization information that is collected will reflect work done in partitions that are running AIX and Linux as well as IBM i, but AIX and Linux do not support collecting this data.

  Enabling the collection of this performance data requires the setting of a configuration parameter on the HMC or Integrated Virtualization Manager (IVM). On the HMC, there is an "Allow performance information collection" checkbox on the processor configuration tab. Select this checkbox on the IBM i partition that you want to collect this data. If you are using IVM, you use the chgsyscfg command, specifying the allow_perf_collection (permission for the partition to retrieve shared processor pool utilization) parameter. Valid values for the parameter are 0, do not allow authority (the default) and 1, allow authority.

Once the performance data collection support is enabled, Collection Services will collect this additional information. At each collection interval, Collection Services will collect partition configuration and utilization information from the hypervisor. The data is stored in the Collection Services database file QAPMLPARH. You also have the ability to get physical processor utilization in the Collection Services database file QAPMSYSPRC.

Displaying the data

You can use the Performance Data Investigator tool found in IBM Navigator for i to view the data collected graphically on the web. These charts can be found in the "Physical System" folder under the Collection Services content package. Some examples of charts using this data are:

- Logical Partitions Overview
- Donated Processor Time by Logical Partition
- Uncapped Processor Time Used by Logical Partition
- Physical Shared Processor Pool Utilization
- Physical Processors Utilization by Physical Processor
- Dedicated Processor Utilization by Logical Partition
- Physical Processors Utilization by Processor Status Overview
- Physical Processor Utilization by Processor Status Detail

Related tasks

Performance Data Investigator
Performance Data Investigator provides a web-based GUI over performance data with interactive charts and tables. You can view and analyze performance data for each of the collectors (Collection Services, IBM i Job Watcher, IBM i Disk Watcher, Performance Explorer, Database SQL Plan Cache, and Database SQL Performance Monitor).

Related reference

Collection Services data files: QAPMLPARHThis Collection Services database file contains logical partition configuration and utilization data as it is known to the hypervisor.

Collection Services data files: QAPMSYSPRCThis Collection Services database file reports utilization data for a system's physical processor units that are based on data that is obtained from the hypervisor.

Collecting ARM performance data

You can use Collection Services to collect Application Response Measurement (ARM) performance data.

The ARM APIs collect the performance data for ARM transactions. (The ARM APIs are a set of APIs developed by the Open Group to allow applications to report the progress of application transactions.) These transactions are reported in the QAPMARMTRT and QAPMUSRTNS database files.

To learn more about ARM APIs, visit The Open Group website at http://collaboration.opengroup.org/tech/management/arm/index.php.

Short lifespan threads and tasks

Collection Services captures performance data for every job, task, and secondary thread that used processor time during a sample interval. This data is reported via records in the QAPMJOBMI file.

There are times when secondary threads and tasks are created that do very little work and terminate quickly; lifespans are generally less than one second. This can be a problem when it happens frequently and is on-going. It can significantly increase the size of collection objects and QAPMJOBMI file members. It also results in increased CPU utilization to capture data and generate the files as well as more resource being used by tools that consume this data.

Although resources consumed within a specific short lifespan thread or task is not significant, they as a group, can be a significant factor in total system utilization. Consequently, they cannot simply be ignored.

Beginning in IBM i 7.1, Collection Services will accumulate data for tasks and secondary threads whose lifespan is less than a specific threshold. Short lifespan secondary threads will be accumulated by job for any job that has such threads. Short lifespan tasks are accumulated by resource affinity domain. This accumulated data will be reported at sample time similar to how other tasks or secondary threads are reported.

The QAPMJOBMI file has a new field "Short lifespan entry count". This field will have a value that is greater than zero for records that contain data for short lifespan threads or tasks. Its value is the number of such entities that were accumulated within the interval for the indicated job or resource affinity domain. The QAPMCONF file reports the short lifespan thresholds used during collection. See the description for GKEY = "F1" in the QAPMCONF article.

By default the threshold that will be used for both tasks and threads is 1000 milliseconds (terminating threads and tasks whose lifespans are less than 1000 milliseconds will not be individually reported). Should there be a need to disable this processing or you want different thresholds, environment variables can be used to accomplish that:

Variable QPM_TASK_SL_THRESHOLD can be created to control short lifespan task processing
Variable QPM_THREAD_SL_THRESHOLD can be created to control short lifespan secondary thread processing

The value associated with the environment variable is the threshold in milliseconds that should be applied. A value of 0 or null will cause all threads or tasks to be reported. You must create these environment variables as system level variables so that Collection Services collector job, QYPSPFRCOL, will see them. The values are obtained only once at the start of a collection; you must cycle an active collection after making changes for the new values to be used.

The following is an example of creating the environment variables and setting the default value of 1000 milliseconds:

```
ADDENVVAR ENVVAR(QPM_TASK_SL_THRESHOLD) VALUE(1000) LEVEL(*SYS)
ADDENVVAR ENVVAR(QPM_THREAD_SL_THRESHOLD) VALUE(1000) LEVEL(*SYS)
```

IBM i Job Watcher

IBM i Job Watcher provides for the collection of job data for any or all jobs, threads, and tasks on the system. It provides call stacks, SQL statements, objects being waited on, Java JVM statistics, wait statistics and more which are used to diagnose job related performance problems.

IBM i Job Watcher is similar in sampling function to the CL commands WRKACTJOB and WRKSYSACT in that each refresh computes delta information for the ending interval. The data collected from the jobs, threads, or tasks being watched is done so in a non-intrusive manner.

IBM i Job Watcher can be configured and managed through the IBM Navigator for i Performance interface or CL commands.

Related concepts

The basics of IBM i Wait Accounting
Wait Accounting is the patented technology built into the IBM i operating system that tells you what a thread or task is doing when it appears that it is not doing anything.

Related tasks

Managing IBM i Job Watcher
Manage IBM i Job Watcher by using IBM Navigator for i.

Related reference

Add Job Watcher Definition (ADDJWDFN)See the Add Job Watcher Definition (ADDJWDFN) command for information about specifying the performance data that is to be collected during a Job Watcher collection.

Remove Job Watcher Definition (RMVJWDFN)See the Remove Job Watcher Definition (RMVJWDFN) command for information about removing a Job Watcher definition from the system.

Start Job Watcher (STRJW)See the Start Job Watcher (STRJW) command for information about starting a Job Watcher collection.

End Job Watcher (ENDJW)See the End Job Watcher (ENDJW) command for information about ending a Job Watcher collection.

IBM i Disk Watcher

IBM i Disk Watcher provides for the collection of disk performance data to diagnose disk related performance problems.

IBM i Disk Watcher allows you to obtain data concerning I/O operations to disk units, along with frequently needed run-time data to determine which objects, files, processes, threads, and tasks are being accessed. This tool surfaces data beyond what is provided by such tools as WRKDSKSTS, WRKSYSSTS, and WRKSYSACT. Disk Watcher provides a mechanism to use short and longer duration traces to collect disk I/O data along with the associated task and object name.

Some potential uses of this tool are:

- Evaluating the performance of I/O operations on multi-path disk units
- Evaluating the performance of I/O queuing
- Determining how performance may be improved by re-spreading data across units
- Determining the optimal placement of devices, IOAs, or buses

IBM i Disk Watcher can be configured and managed through the IBM Navigator for i Performance interface or CL commands.

Related tasks

Managing IBM i Disk Watcher
Manage IBM i Disk Watcher by using IBM Navigator for i.

Related reference

Disk Watcher data filesSee the Disk Watcher data files topic for information about the database files that are created by Disk Watcher that contain performance data.

Add Disk Watcher Definition (ADDDWDFN)See the Add Disk Watcher Definition (ADDDWDFN) command for information about specifying the performance data that is to be collected during a Disk Watcher collection.

Remove Disk Watcher Definition (RMVDWDFN)See the Remove Disk Watcher Definition (RMVDWDFN) command for information about removing a Disk Watcher definition from the system.

Start Disk Watcher (STRDW)See the Start Disk Watcher (STRDW) command for information about starting a Disk Watcher collection.

End Disk Watcher (ENDDW)See the End Disk Watcher (ENDDW) command for information about ending a Disk Watcher collection.

Performance Explorer

Performance Explorer collects more detailed information about a specific application, program or system resource, and provides detailed insight into a specific performance problem. This includes the capability both to perform several types and levels of traces and to run detailed reports.

Performance Explorer is a data collection tool that helps the user identify the causes of performance problems that cannot be identified by collecting data using Collection Services or by doing general trend analysis. Two reasons to use Performance Explorer include:

- Isolating performance problems to the system resource, application, program, procedure, or method that is causing the problem
- Analyzing the performance of applications

The AS/400 Performance Explorer Tips and Techniques book provides additional examples of the Performance Explorer functions and examples of the enhanced Performance Explorer trace support.

Performance Explorer is a tool that helps find the causes of performance problems that cannot be identified by using tools that do general performance monitoring. As your computer environment grows both in size and in complexity, it is reasonable for your performance analysis to gain in complexity as well. The Performance Explorer addresses this growth in complexity by gathering data on complex performance problems.

Note: Performance Explorer is the tool you need to use after you have tried the other tools. It gathers specific forms of data that can more easily isolate the factors involved in a performance problem; however, when you collect this data, you can significantly affect the performance of your system.

This tool is designed for application developers who are interested in understanding or improving the performance of their programs. It is also useful for users knowledgeable in performance management to help identify and isolate complex performance problems.

Related reference

Performance Tools PDFThis IBM Redbooks publication explains how to use performance tools to collect data about the performance of a system, job, or program. It also explains how to analyze and print the data to help identify and correct any problems.

Performance Explorer concepts

Performance Explorer works by collecting detailed information about a specified system process or resource. This topic explains how Performance Explorer works, and how best to use it.

Performance Explorer has advantages for people who need detailed performance analysis. Using Performance Explorer you can:

- Determine what is causing a performance problem on the system down to the level of user, job, file, object, thread, task, program, module, procedure, statement, or instruction address.
- Collect performance information on user-developed and system software.
- Do a detailed analysis on one job without affecting the performance of other operations on the system.
- Analyze data on a system other than the one on which it was collected. For example, if you collect data on a managed system in your network, you can send it to the central site system for analysis.

Like Collection Services, Performance Explorer collects data for later analysis. However, they collect very different types of data. Collection Services collects a broad range of system data at regularly schedules intervals, with minimal system resource consumption. In contrast, Performance Explorer starts a session that collects trace-level data. This trace generates a large amount of detailed information about the resources consumed by an application, job, or thread. Specifically, you can use Performance Explorer to answer specific questions about areas like system-generated disk I/O, procedure calls, Java method calls, page faults, and other trace events. It is the ability to collect very specific and very detailed information that makes the Performance Explorer effective in helping isolate performance problems. For example, Collection Services can tell you that disk storage space is rapidly being consumed. You can use Performance Explorer to identify what programs and objects are consuming too much disk space, and why.

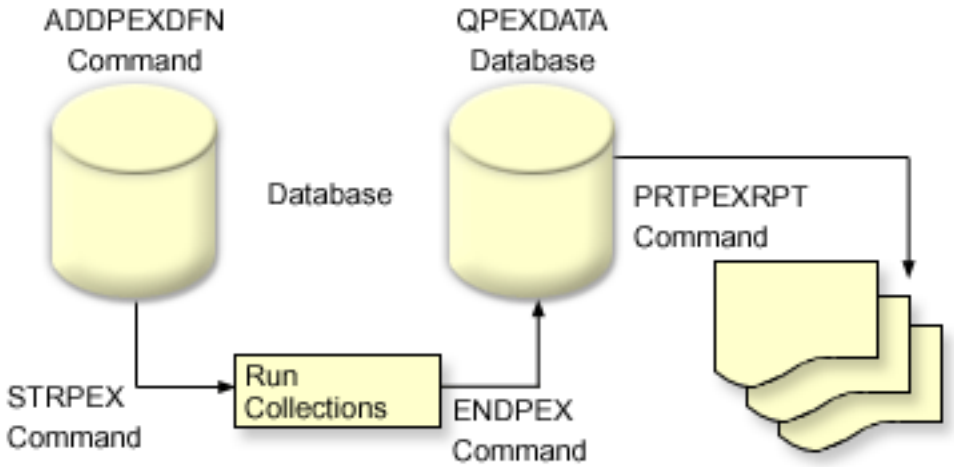
When Performance Explorer is running, it creates only the files that are needed for the collection.

Note: You can collect Performance Explorer data and Collections Services data at the same time.

How Performance Explorer works

The following figure should help you become familiar with the normal path through the Performance Explorer. For details on each of these steps, see Configure Performance Explorer. The figure shows a basic work cycle that consists of the following steps:

1. Define a Performance Explorer data collection. You can also add a filter to limit the amount of data collected by specifying a compare value for specific events.
2. Start the Performance Explorer to collect the data based on your definition.
3. Run your program, command, or workload.
4. End the collection, which saves the collected data to a set of database files.
5. Create and print reports from the database files.



To learn more about Performance Explorer, refer to any of the following Performance Explorer topics.

Related concepts

Collection Services

Collection Services provides for the collection of system and job level performance data. It is the primary collector of performance data.

Related tasks

Configuring Performance Explorer

To collect detailed trace information, you need to tailor Performance Explorer to work optimally with the application process from which the trace is being taken.

Performance Explorer definitions
The parameters and conditions that determine what data Performance Explorer collects and how it collects it are configured and stored using Performance Explorer definitions. This topic explains how to use these definitions and provides a sample illustrating a simple definition.

To collect Performance Explorer data, you need to tell Performance Explorer what data to gather. You do this by using the Add Performance Explorer Definition (ADDPEXDFN) command to create a Performance Explorer definition. After the definition is completed and saved, you are ready to continue to the next task in the cycle of work.

Before creating a new definition, consider what kinds of information you want and the amount of detail you need. The Performance Explorer provides the following types of data collection:

Statistics type definitions

Identifies applications and IBM programs or modules that consume excessive CPU use or that perform a high number of disk I/O operations. Typically, you use the statistical type to identify programs that should be investigated further as potential performance bottlenecks.

- Good for first order analysis of IBM i programs, procedures, and MI complex instructions.
  - Gives number of invocations
  - Gives both inline and cumulative CPU usage in microseconds
  - Gives both inline and cumulative number of synchronous and asynchronous I/O
  - Gives number of calls made
- Works well for short or long runs
- Size of the collected data is fairly small and constant for all runs
- Run time collection overhead of ILE procedures may be a problem due to the frequency of calls. Although run time is degraded, the collected statistics are still accurate because Performance Explorer removes most of the collection overhead from the data.
- Uses combined or separated data areas. The MRGJOB parameter on the ADDPEXDFN command specifies whether all program statistics are accumulated in one data area, or kept separate (for example, one data area for each job).

The statistics can be structured in either a hierarchical or flattened manner.

- A hierarchical structure organizes the statistics into a call tree form in which each node in the tree represents a program procedure run by the job or task.
- A flattened structure organizes the statistics into a simple list of programs or procedures, each with its own set of statistics.

Here is an example of a Performance Explorer statistics definition called MYSTATS that will show CPU and disk resource usage on a per program or procedure level.

```
ADDPEXDFN DFN(MYSTATS) /* The name of the definition. */
TYPE(*STATS) /* The type of definition */
JOB(*ALL) /*All Jobs */
TASKS(*ALL) /*All tasks */
   DTAORG(*FLAT) /* Do not keep track of who calls who */
```

Profile type definitions

Identifies high-level language (HLL) programs, modules, procedures, and statements that consume excessive CPU utilization based on source program statement numbers.

- Program profile (specify TYPE(*PROFILE) and PRFTYPE(*PGM) on the ADDPEXDFN command)

- Gives detailed breakdown of where you are spending time within a set of programs within a specific job.
- Can summarize the data by program, module, procedure, statement, or instruction.
- Size of collection is fairly small and constant regardless of length of run.
- Limit of 16 MI programs means that you should use this as a second order analysis tool.
- Can vary overhead by changing sample interval. An interval of 2 milliseconds seems a good first choice for benchmarks.
- No restrictions on pane size due to the number of programs specified or the size of the programs specified.

Here is an example of a Performance Explorer program profile definition called PGMPROF that will show usage for a particular procedure.

```
ADDPEXDFN DFN(PGMPROF) /* The name of the definition. */
TYPE(*PROFILE) /* The type of definition */
JOB(*ALL) /*All Jobs */
PGM((MYLIB/MYPGM MYMODULE MYPROCEDURE)) /* The name of the program to monitor. */
INTERVAL(1) /* 1-millisecond samples will be taken. */
```

- Job profile (specify the following on the ADDPEXDFN command: TYPE(*PROFILE) and PRFTYPE(*JOB))
  - Gives detailed breakdown of where you are spending time in the set of jobs or tasks of the collection.
  - Size of collection is relatively small but not constant. The size increases as the length of the run increases.
  - Can profile all jobs and tasks on the system or can narrow the scope of data collected to just a few jobs or tasks of interest.
  - Can vary overhead by changing sample interval. An interval of 2 milliseconds seems a good first choice for benchmarks.

  Here is an example of a Performance Explorer job profile definition called ALLJOBPROF that will show usage for all your jobs.

```
ADDPEXDFN DFN(ALLJOBPROF) /* The name of the definition. */
TYPE(*PROFILE) /* The type of definition */
PRFTYPE(*JOB) /* A job profile type will be monitored. */
JOB(*ALL) /*All Jobs */
TASKS(*ALL) /*All tasks */
INTERVAL(1) /* 1-millisecond samples will be taken. */
```

Trace definitions

Gathers a historical trace of performance activity generated by one or more jobs on the system. The trace type gathers specific information about when and in what order events occurred. The trace type collects detailed reference information about programs, Licensed Internal Code (LIC) tasks, IBM i job, and object reference information.

- Some common trace events are:
  - Program and procedure calls and returns
  - Storage, for example, allocate and deallocate.
  - Disk I/O, for example, read operations and write operations.
  - Java method, for example, entry and exit.
  - Java, for example, object create and garbage collection.
  - Journal, for example, start commit and end commit.
  - Synchronization, for example, mutex lock and unlock or semaphore waits.
  - Communications, for example, TCP, IP, or UDP.
- Longer runs collect more data.

Here is an example of a Performance Explorer trace definition called DISKTRACE that will show usage for all disk events.

```
 ADDPEXDFN DFN(DISKTRACE) /* The name of the definition. */
 TYPE(*TRACE) /* The type of definition */
 JOB(*ALL) /*All Jobs */
 TASKS(*ALL) /*All tasks */
 TRCTYPE(*SLTEVT) /* Only selected individual events and machine instructions
are included in the trace definition */
 SLTEVT(*YES) /* *SLTEVT allows you to specify individual machine instructions
and events to be specified in addition to the categories of events
available with the TRCTYPE parameter. */
 DSKEVT((*ALL)) /* All disk events are to be traced. */
```

Here is an example of a Performance Explorer trace definition called HEAPEVENTS.

```
 ADDPEXDFN DFN(HEAPEVENTS) /* The name of the definition. */
 TYPE(*TRACE) /* The type of definition */
 JOB(*ALL) /*All Jobs */
 TASK(*ALL) /*All tasks */
 MAXSTG (100000) /*Maximum storage. Set to 100000 because the default of
10000 KB is often too small for the large number of heap events that can be
generated when tracing all jobs and all tasks.*/
 TRCTYPE(*HEAP) /* Selects all heap events from the STGEVT
(storage events) parameter. */
```

Related concepts

Performance Explorer reports

After you have collected performance data with a Performance Explorer session, you can view it by running the included reports or by querying the database files directly.

Related tasks

Configuring Performance Explorer

To collect detailed trace information, you need to tailor Performance Explorer to work optimally with the application process from which the trace is being taken.

Related reference

Add Performance Explorer Definition (ADDPEXDFN) commandSee the Add Performance Explorer Definition (ADDPEXDFN) command for information about specifying the performance data that is to be collected during a Performance Explorer collection.

Performance Explorer database files

The data that Performance Explorer collects is stored in Performance Explorer database files.

The following table shows the Performance Explorer (PEX) data files collected by the system when using data collection commands. Type the Display File Field Description (DSPFFD) command as follows to view the contents for a single file:

```
DSPFFD FILE(xxxxxxxxxx)
```

where *xxxxxxxxx* is the name of the file that you want to display.

Type of information contained in file	File name
Trace Resources Affinity	QAYPEAFN
Auxiliary storage management event data	QAYPEASM
Auxiliary storage pool (ASP) information data	QAYPEASPI
Base event data	QAYPEBASE
Basic configuration information	QAYPECFGI
Communications event data	QAYPECMN
Disk event data	QAYPEDASD

Type of information contained in file	File name
Disk server event data	QAYPEDSRV
Event type and subtype mapping	QAYPEEVENT
File Serving event data	QAYPEFILSV
Configured filter information	QAYPEFTRI
Performance measurement counter (PMC) selection	QAYPEFQCFG
Heap event data	QAYPEHEAP
Hardware monitor data	QAYPEHMON
Hardware monitor total data	QAYPEHTOT
Performance Explorer Java event data	QAYPEJVA
Performance Explorer Java class information data	QAYPEJVCI
Performance Explorer Java method information data	QAYPEJVMI
Performance Explorer Java name information data	QAYPEJVNI
Licensed Internal Code (LIC) bracketing data	QAYPELBRKT
Machine interface (MI) complex instructions collected on	QAYPELCPLX
Jobs collected on	QAYPELJOB
Licensed Internal Code (LIC) modules to collect data on	QAYPELLIC
Metrics to collect data on	QAYPELMET
Machine interface (MI) program, module, or procedures collected on	QAYPELMI
Task names to collect data on	QAYPELNAMT
Task number to collect data on	QAYPELNUMT
Configured tasks	QAYPELTASK
Machine interface (MI) program bracketing data	QAYPEMBRKT
Machine interface (MI) complex instructions mapping	QAYPEMICPX
Addresses of machine interface (MI) pointer	QAYPEMIPTR
Machine interface (MI) user event data	QAYPEMIUSR
Portable Application Solutions Environment (PASE) event data	QAYPEPASE
Page fault event data	QAYPEPGFLT
Program profile data	QAYPEPPANE
Licensed Internal Code (LIC) address resolution mapping	QAYPEPROCI
Resource management process event data	QAYPERMPM

Type of information contained in file	File name
Resource management seize lock event data	QAYPERMSL
Reference information	QAYPEREF
Miscellaneous resolution data	QAYPERINF
Database level indicator	QAYPERLS
General information	QAYPERUNI
Segment address range (SAR) data	QAYPESAR
Segment address resolution mapping	QAYPESEGI
Basic statistics data	QAYPESTATS
Synchronization event data	QAYPESYNC
Process and task resolution mapping	QAYPETASKI
Trace job equivalent event data	QAYPETBRKT
Common trace data for all events	QAYPETIDX
Trace index data (by time and task)	QAYPETIDXL
Trace index data (by time)	QAYPETID2L
Task switch event data	QAYPETSKSW
User-defined bracketing hook data	QAYPEUSRDF

Migration of Performance Explorer database files

The Performance Explorer (PEX) database files change from release to release, as new events and new data are added to the files. When you migrate to a new release of IBM i, if the system finds incompatible PEX database files, it moves these files to the QPEXD*vrmxx* library, where *vrm*=version. The system displays a status message that indicates that the files are being moved. After the files are moved, the system displays a completion message that indicates whether the move succeeded or failed. If the move fails, the system displays the `Incompatible repository` message.

Related concepts

Performance Explorer reports
After you have collected performance data with a Performance Explorer session, you can view it by running the included reports or by querying the database files directly.

Performance Explorer reports
After you have collected performance data with a Performance Explorer session, you can view it by running the included reports or by querying the database files directly.

Performance Explorer gathers detailed information about a program or job's behavior and performance and stores this information in Performance Explorer database files. You can query these files with SQL, or by running one of several reports. You can generate four different reports with Performance Explorer: Statistics, Profile, Trace, and Base reports. See Performance Explorer definitions for information on why you would use a particular definition to generate one of these reports. Each report is discussed in detail in the Performance Tools.

You can create and print Performance Explorer reports by using the Print Performance Explorer Report (PRTPEXRPT) command. Use the OUTFILE parameter when you want to customize your Trace Report. The following commands are examples for printing reports for each type of Performance Explorer data:

- Print a *STATS report sorting by the CPU time used

```
PRTPEXRPT MBR(MYSTATS) LIB(MYLIB) TYPE(*STATS) STATSOPT(*CPU)
```

- Print a profile report summarized by procedure

```
PRTPEXRPT MBR(MYPROFILE) LIB(MYLIB) TYPE(*PROFILE)
PROFILEOPT(*SAMPLECOUNT *PROCEDURE)
```

- Print a trace sorted by task ID

```
PRTPEXRPT MBR(MYTRACE) LIB(MYLIB) TYPE(*TRACE) TRACEOPT(*TASK)
```

Performance Explorer stores its collected data in the QAVPETRCI file, which is located in the QPFR library. Type the following command to view the contents for a single record:

```
DSPFFD FILE(QPFR/QAVPETRCI)
```

Related concepts

Performance Explorer definitions
The parameters and conditions that determine what data Performance Explorer collects and how it collects it are configured and stored using Performance Explorer definitions. This topic explains how to use these definitions and provides a sample illustrating a simple definition.

Related reference

Performance Explorer database files
The data that Performance Explorer collects is stored in Performance Explorer database files.

Performance Tools PDFThis IBM Redbooks publication explains how to use performance tools to collect data about the performance of a system, job, or program. It also explains how to analyze and print the data to help identify and correct any problems.

Print Performance Explorer Report (PRTPEXRPT) commandSee the Print Performance Explorer Report (PRTPEXRPT) command for information about printing a formatted listing of the data within a Performance Explorer collection.

Configuring Performance Explorer

To collect detailed trace information, you need to tailor Performance Explorer to work optimally with the application process from which the trace is being taken.

To configure Performance Explorer, follow these steps:

1. Create a session definition that informs the system which performance data you want to collect. On the Add Performance Explorer Definition (ADDPEXDFN) display, specify the collection type and a name for the definition. This definition is stored as a database member by that name in the QAPEXDFN file in library QUSRSYS. The name that you specify is used on the Start Performance Explorer (STRPEX) command.
2. (Optional) Add a filter (Add PEX Filter (ADDPEXFTR) command). A Performance Explorer filter identifies the performance data that is to be collected during a Performance Explorer session, and is meant to limit the amount of data collected by specifying a compare value for specific events.
3. Start collecting data (Start Performance Explorer (STRPEX) command). A job may be in more than one Performance Explorer collection if the *PMCO event is not being collected. If the *PMCO event is being collected, then a job can be in more than one collection only if all the collections have the same interval specification (ADDPEXDFN INTERVAL() parameter). You can specify a definition and optional filter on the STRPEX command.
4. Run your command, program, or workload for data that you want to analyze.
5. Stop collecting the data and save it to database files for analysis. Use the End Performance Explorer (ENDPEX) command to stop the collection.
6. Analyze the performance data. The Print Performance Explorer Report (PRTPEXRPT) command provides unique reports for each type of data (statistical, profile, trace profile, or trace).

   The following are other options for analysis:

   - Write your own queries for the set of database files.

- Use iDoctor for IBM i - PEX Analyzer. iDoctor for IBM i is a set of software performance analysis tools and associated services that extend your ability to evaluate the health of your system by gathering detailed information and providing automated, graphical analysis of this data.

7. To end the Performance Explorer session, use the End Performance Explorer (ENDPEX) command.

All of the Performance Explorer commands can be accessed with one of the following methods:

- The command interface. Type the commands from the command line. All the commands are part of the IBM i operating system.
- The Performance Tools menu options.

Related concepts

Performance Explorer definitions
The parameters and conditions that determine what data Performance Explorer collects and how it collects it are configured and stored using Performance Explorer definitions. This topic explains how to use these definitions and provides a sample illustrating a simple definition.

Performance Explorer concepts
Performance Explorer works by collecting detailed information about a specified system process or resource. This topic explains how Performance Explorer works, and how best to use it.

Related reference

Add PEX filter (ADDPEXFTR) commandSee the Add Performance Explorer Filter (ADDPEXFTR) command for information about adding a Performance Explorer (PEX) filter to the system.

Start Performance Explorer (STRPEX) commandSee the Start Performance Explorer (STRPEX) command for information about starting a Performance Explorer collection.

Print Performance Explorer Report (PRTPEXRPT) commandSee the Print Performance Explorer Report (PRTPEXRPT) command for information about printing a formatted listing of the data within a Performance Explorer collection.

Ending Performance Explorer
To end the Performance Explorer session, use the End Performance Explorer (ENDPEX) command.

The End Performance Explorer (ENDPEX) command performs the following actions on the collected data:

- Places the collected data in files QAYPExxx in the library that you specify. Use OPTION(*END) and DTAOPT(*LIB) to do this. The database member name for all the QAYPExxx files uses the session name as the default unless you specify a name for the DTAMBR parameter. You can specify RPLDTA(*NO) to not overwrite existing data with the new data or RPLDTA(*YES) to overwrite the existing data with the new data. Unless you are a very sophisticated user, use RPLDTA(*NO).
- Places the collected data into a single IBM-defined file. Use OPTION(*END) and DTAOPT(*MGTCOL) to do this. Typically, you would use *MGTCOL only under the direction of an IBM service representative. Specifying the *MGTCOL value on the DTAOPT parameter saves the collection information into a management collection object. The management collection object option should be used only if the data is going to be shipped to IBM. The performance tools can analyze only the database files.
- Discards the collected data. Use OPTION(*END) if you want to save the data or DTAOPT(*DLT) to discard any collected data. You do this when you determine the collected data cannot be used. For example, one of the suspected jobs did not start as expected. If you choose the *DLT option, the collected performance data for the session is never saved.
- Suspends the collection session but does not end it. Use OPTION(*SUSPEND) to do this. You can later start the data collection again by issuing the STRPEX command with OPTION(*RESUME) for the specific session ID.

Note: If you forget the active collection session name, use the ENDPEX SSNID(*SELECT) command.

Viewing and analyzing data

The data that is collected by a performance data collector is stored in database files. IBM i has many tools available to help you manage, view and analyze the data.

Related reference

End to End Performance Management on IBM iThis IBM Redbooks publication explains the tasks and tools that are associated with Performance Management on IBM i.

IBM Navigator for i

IBM Navigator for i is a web console interface where you can perform the key tasks to administer your IBM i. The web application is part of the base IBM i operating system, and can be easily accessed by pointing your browser to http://systemName:2001. Function available in IBM Navigator for i can be used to help manage performance of your system.

Related concepts

IBM Navigator for iSee the IBM Navigator for i topic to learn about more functions available within IBM Navigator for i.

IBM Navigator for i Performance interface

Use the IBM Navigator for i Performance interface to view, collect, and manage performance data by bringing together various performance information and tools into one centralized place.

Related concepts

IBM Navigator for iSee the IBM Navigator for i topic to learn about more functions available within IBM Navigator for i.

Performance Data Investigator

Performance Data Investigator provides a web-based GUI over performance data with interactive charts and tables. You can view and analyze performance data for each of the collectors (Collection Services, IBM i Job Watcher, IBM i Disk Watcher, Performance Explorer, Database SQL Plan Cache, and Database SQL Performance Monitor).

It can be accessed by following these steps:

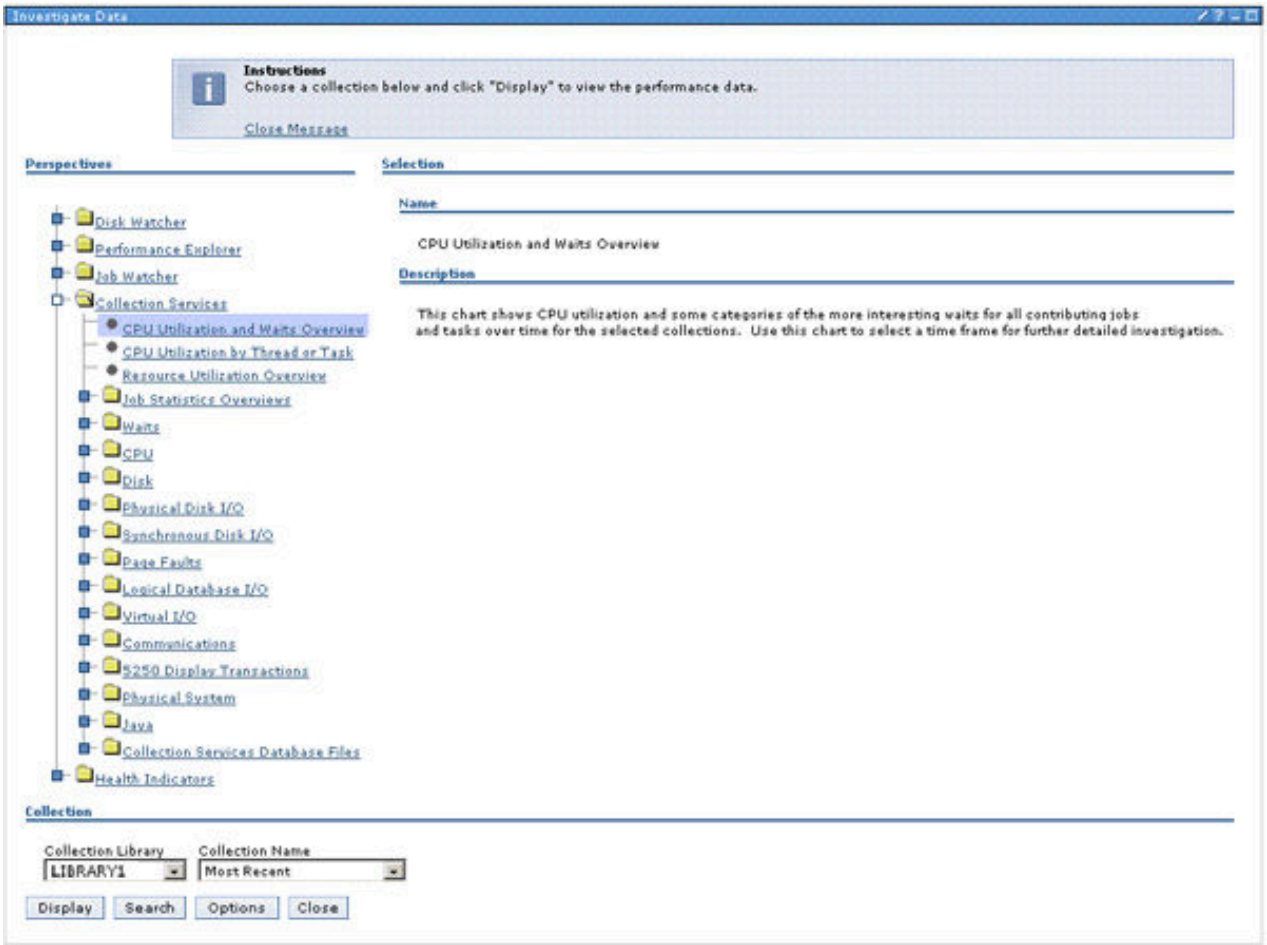
1. Launch IBM Navigator for i.
   a) Access the following URL from a web browser where hostA is your IBM i partition name: http://hostA:2001.
   b) Log in using your system id and password.
2. Select **Performance** > **Investigate Data** from the left panel.

Investigate Data

Selecting the Investigate Data task launches the powerful Performance Data Investigator tool. With this tool, you can view and analyze data that is stored in performance collections in chart or table form.

From the Investigate Data main page, you select the perspective and collection you want to analyze. Each collector (Collection Services, Job Watcher, Disk Watcher, and Performance Explorer) has an associated *content package* that contains predefined perspectives of that data collection type. There are also IBM-shipped content packages for Health Indicators, Monitors, and Database. The Health Indicators package contains perspectives that show the general health of your partition and allows user-defined thresholds to be configured. It is also possible to have custom content packages and perspectives that were created and saved by a user in the list.

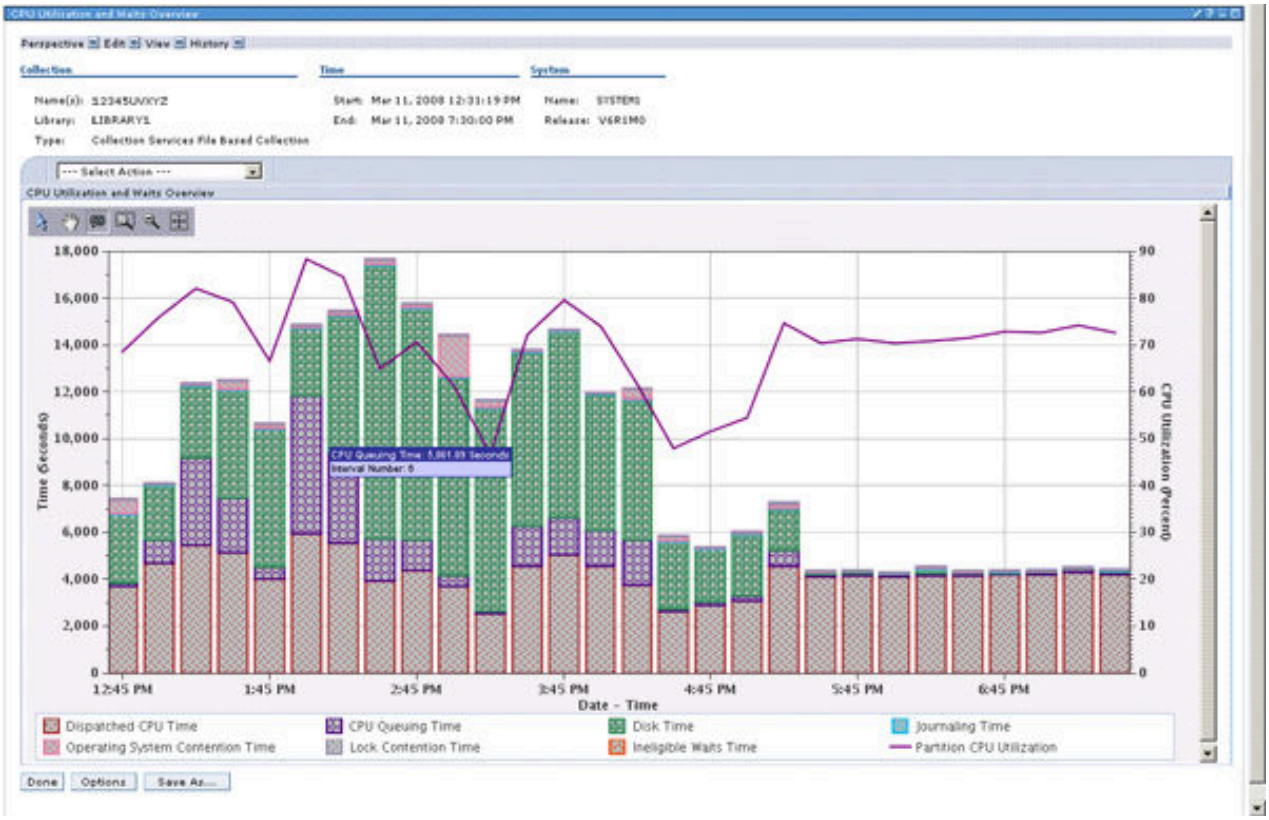
The following image shows an example of the hierarchical format of the list of content packages that are found on the **Investigate Data** page:



Each content package has a list of perspectives under it to provide a different interpretation (rendering) of the data. A *perspective* defines one page of charts or tables that can be used to render the data you want to analyze.

The **Investigate Data** page can be used to select the perspective and collection you want to analyze by following these steps:

1. Expand the content package that you want to work with by clicking the square next to it, or by selecting the content package name directly.
2. Perspectives are stored hierarchically in the content package. To navigate to a subfolder, click the square next to the folder name.
3. When you find a perspective that you want to view, select it by clicking it. On the opposite side of the page, you see a brief description of the perspective. At the bottom of the page, you see two option lists to help you select a collection.
4. Choose the library that you want to work with using the **Collection Library** list. Selecting the library causes the Collection Name list to update with collections in the chosen library. Only collections that are valid for the selected perspective is included in the list.
5. Select the collection that you want to work with using the **Collection Name** list.
6. Click **Display** to view the collection data that is rendered in the chosen perspective.



Only collections that are created in IBM i 6.1 or later or were converted to the 6.1 format (by using the Performance Convert Collection task in IBM Navigator for i or the Convert Performance Collection (CVTPFRCOL) command) are available to analyze in Performance Data Investigator.

You might need to install some or all of the following, depending upon the level of function that you need:

- IBM Performance Tools for i (5770-PT1) Option 1 - Manager Feature
  - Performance Explorer content package
  - IBM i Disk Watcher functions and content package
  - Database content package
  - Batch Model functions and content package
- IBM Performance Tools for i (5770-PT1) Option 3 - Job Watcher
  - IBM i Job Watcher functions and content package

Related concepts

The basics of IBM i Wait Accounting
Wait Accounting is the patented technology built into the IBM i operating system that tells you what a thread or task is doing when it appears that it is not doing anything.

Collecting and displaying CPU utilization for all partitions
When using multiple partitions, it can be important to understand the overall utilization of processing capability, across all partitions, regardless of whether the partition is running IBM i, AIX®, or Linux®. IBM i provides a way to collect and display this data.

Working with a Perspective
A perspective is the rendering (either in chart or table form) of the specified collections data. Once the collection data is displayed, there are several features available to help you interact and analyze the data.

- One of the most powerful features is the ability to drill down into your data collection. By selecting a second perspective from the action list menu of your current perspective view, you can drill down to another view of your data. Each chart can provide different insight to your data, but the power in the Performance Data Investigator tool is the ability that each perspective has to affect the subsequent

perspectives. Each perspective accessed through drilldown allows you to narrow your custom view context. By selecting points on a chart or rows in a table, you are indicating which data sets should be used to narrow the scope of future views. For instance, you may notice a potential performance problem while viewing the CPU Utilization and Waits Overview perspective. You can narrow the scope of interest by zooming in on a specific Date-Time range. You can then select another perspective from the action list menu, such as CPU Utilization by Thread or Task, to gain additional relevant information. The process of selecting data points and drilling down to another perspective is repeatable. You can use the History menu at the top of the perspective to keep track of perspectives you have displayed as well as a way to return to them.

- Because you have the ability to modify the way data is shown, a Save action is provided so that you can reference the modified perspective in the future.
- By clicking Options, you can specify persistent user preferences to be used within Performance Data Investigator. One of these options, "Enable Design Mode" will give you the ability to create your own content packages and perspectives.
- There are many other interactive chart and table features and actions available to help make Performance Data Investigator flexible and easy to use, including the ability to Export the data or Modify the SQL used to produce the perspective.

Save Perspective
The Save Perspective feature allows you to save a modified perspective for future use.

As you investigate data in a collection through drilldown and context modification, the current context perspective is modified. By saving this modified perspective, you will be able to return to it in the future and quickly render any collection data (of the appropriate type) to the specifics of the perspective created.

To perform the Save action from a customized table or chart, click on the "Save As" button at the bottom, or use the "Save As" action from the Perspective menu at the top . The Save perspective page lets you specify a name and description to help identify this new perspective. Once the save is completed successfully, the perspective page is again shown with an additional message indicating a successful save occurred and a URL that will let you return directly to this same perspective in the future.

The following are key points for this feature:

- The URL returned from the save can be shared with others, as long as they are using the IBM Navigator for i on the same system.
- The saved perspective will be available from the Investigate Data page of Performance Data Investigator. The saved perspective can be saved in either an unlocked package on the system and visible to all users or in a user's private custom content package. The custom content package is located under "Custom Perspectives - USERNAME" content package on the main Investigate Data page.
- The content package is stored in IFS at "QIBM/UserData/OS400/iSeriesNavigator/config/PML/CCP" in a file named "CCP_USERNAME.PML". It should be backed up if you want to protect the saved perspectives on the system.
- The saved perspectives can be used to analyze a different performance collection, simply by choosing the collection from the library and name dropdown boxes. The context information must apply to the new collection to be properly rendered. If an empty chart appears, verify and alter your perspective context using the Change Context action.

Related tasks
Change Context
The Change Context action allows you to inspect and modify the current context information for the chart or table view.

Options
From the Options page, you can set persistent user preferences unique to Performance Data Investigator.

The options that can be set on this page are:

- Use Patterns - Specifies whether to use patterns where applicable in charts. This option is selected by default.
- Show Charts - Sets the default to show charts whenever possible rather than tables. This option is selected by default.
- Enable Design Mode - Enables advanced features that allow design and development of new content packages. This option is not selected by default.
- Show Help - Enables help messages for many tasks. This option is selected by default.
- Show SQL error messages - Enables SQL error messages to be shown. This option is selected by default.
- Set Table Size - Allows you to specify the number of visible rows and columns shown for a table.

The default library specifies the library that will be used when a collection is selected. It can be set to one of the following:

- Collection Services configured library
- Last visited library
- A specified library

The System Monitor options that can be set on this page are:

- Show thresholds: Specifies whether to show the thresholds in system monitor charts. This option is not selected by default.

Enable Design Mode
The setting to enable Design Mode is found on the Options page. Enabling Design Mode allows custom development of new content packages.

By selecting Design Mode, additional advanced features become available. This includes the ability to:

- Create new folders and perspectives
- Edit folders and perspectives
- Add, Edit, and Delete views in a perspective
- Delete custom folders and perspectives
- Add, Edit and Delete data series for a chart view
- Add, Edit, and Delete thresholds for a for a chart view
- Use Advanced Edit option to edit the XML used to produce the perspective
- "Lock" custom perspectives so they cannot be edited in the future
- Arrange ordering of perspectives in a folder by using Move Up or Move Down
- Refresh the Perspectives list

Related tasks
Creating a Custom Package
When design mode is enabled, you can create your own custom content packages. These packages can contain perspectives tailored to your needs.

Refresh Perspectives
Use Refresh Perspectives to reload all the content packages on the system.

The Investigate Data page of Performance Data Investigator has a Refresh Perspectives button at the bottom. Click this button will force the reload of all the content packages on the system so that your view is updated. This is valuable when manually developing new content packages on the system.

Refresh Perspectives is only available when design mode is enabled.

Chart Features
There are many features available for Chart Views that allow you to further investigate your data and modify the context of the chart view.

The interactive capability of charts makes them a powerful tool for analyzing performance data. Being able to visualize graphed performance data can make peaks and valleys of activity stand out. The ability to drill down into further details and specific time intervals is very useful. The following chart icons are interactors which can aid you with analysis:

- By clicking the arrow icon, you will be able to select a point or bar on the chart. Selecting data points allows Performance Data Investigator to help narrow future analysis based on the context you find interesting. By selecting points on the chart and then an action from the action list menu, you can drill down for further analysis on that data. This is the default interactor.
- By clicking the hand icon, you will be able to pan the chart by clicking and dragging around the chart image. This interactor is useful when the chart is zoomed in close to a portion of the chart and you want to move around without adjusting your zoom factor.
- By clicking the talk bubble icon, you will enable or disable tooltip information. Tooltips are displayed when hovering over any chart data points and can be defined to show a number of interesting bits of information. It is disabled by default.
- By clicking the icon that looks like a magnifying glass with a dotted rectangle behind it, you will enable the zoom in action interactor. Zooming can be performed by clicking a point and dragging a window around the portion of the chart you wish to further investigate. When you release the mouse button, the perspective chart will be re-rendered to display only the area you have selected.
- By clicking the magnifying glass icon with the minus sign ("-") in it, you will incrementally zoom out from your current zoom level until you return to the entire chart view for the perspective.
- By clicking the square icon with arrows pointing out in all directions, you reset the chart zoom to the maximum level. This will show you the entire chart view for the perspective.

Only one chart interactor can be active at a time.

In addition, the "Show as Table" action is added to the chart action list menu. This action will change your current view to render it as a table.

Table Features
There are many features available for Table Views that help you to further investigate your data.

Tables retain the capability to select interesting portions of your data by selecting entire rows. Tables can easily be filtered, sorted, or searched for specific information. Switching back to a chart will reflect any ordering and filtering changes made to the table.

There are several actions available that are specific to tables. The following are available by either clicking the table icon features or by selecting a specific action from the action list menu.

- Select All - Selects all check boxes for all rows of the Select column. Selected rows can then be manipulated by Performance Data Investigator through the actions on the action list menu.
- Deselect All - Clears all check marks in the Select column.
- Filter Row Show or Hide Toggle - The Filter row is hidden by default. Select this icon to show the filter row to be able to filter the data. This allows you to refine the data displayed by specific parameters of the column values.
- Clear All Filters - Removes any custom filters.
- Edit Sort - Allows you to sort the columns displayed in the table view based on the values in up to three columns. Click this icon to perform the edit sort to the table view. The edit sort query will be displayed.

Select up to three column headings for first, second, and third sort positions. Then select ascending or descending order for each sort to display data sorted in that manner of priority.

- Clear All Sorts - Removes all custom sorts that have been enabled.

The following actions are available only on the action list menu for tables:

- Show as chart - This action will change your current view to render it as a chart.
  - This action is not available if "Show Charts" is not selected on the Options page.
  - If you try to select the action "Show as chart" while viewing a table that does not have a data series defined, this action will require you to define a data series before continuing.
- Columns - Add or remove columns to the table view. You can also rearrange the column order by moving the column headings up or down.
- Show find toolbar - Allows you to perform a search within the table view.
- Restore defaults - Restores the table to the default sort and filtering.

The following actions are available by clicking icons found on the table:

- Select - By clicking the check box next to any row you can select a row in the table. Selecting a row allows Performance Data Investigator to help narrow future analysis based on the context you find interesting. The action selected from the action list menu will be performed on the selected rows.
- Sorting - By clicking the sort indicator (circumflex "^") in any column header, the table can be sorted in ascending or descending order. You can also use the Edit Sort or Clear All Sorts icons to manipulate your current sort criteria.
- Filtering - By clicking the Show Filter Row table icon, a filter row will be shown under the column headings for the table. This allows you to refine the data displayed by specific parameters of the column values. The filter row is hidden by default. To filter the rendered data based on a condition for a column, click the "Filter" link for that column.

Table and Chart Actions
There are many useful actions that can be performed from a table or a chart view.

Export
The Export View function allows you to export a chart or table to a file location for later reference. Data can be exported to an image (charts only), comma delimited, or tab delimited file.

Selecting the Export action from a chart or table view brings you to the Export page. From this page, you can inspect and modify the following fields for the perspective view that you wish to save:

- Title - This is the title that will be used at the top of the file saved.
- Format - Select the format for saving the perspective view:

  For a table, the choices are: comma delimited (*csv) or tab delimited (*.txt) file formats

  For a chart, you may select from: image (*.png), comma delimited (*.csv), or tab delimited (*.txt) formats
- Data Range - Allows you to change the data range in the view exported. The choices available are:

  All data - This will export all of the data available by the current view.

  Displayed data - This will export only the current visible data of the view.

  User-defined range - When this option is selected, you can specify a first and last record number. Record number refers to the index of a data element among all data for that data series. This will export the specified range.

When OK is selected from the Export page, a new browser window may open to download the chart or table view. You may also need to respond to a message bar on your browser to allow the file to be downloaded. The file download window will allow you to select Open or Save the data to your client.

Modify SQL
The Modify SQL action allows you to inspect and modify the SQL statements used to retrieve the data from the performance collection in the current context.

A perspective view is rendered with the selected collections data through a defined SQL statement. Modify SQL allows you the opportunity to change the query used to retrieve the data. Experience with SQL development is advised before modifying the SQL statements. You can use this feature to understand which database fields Performance Data Investigator uses to calculate the complex metrics displayed in a view.

Be aware, the queries rely on SQL aliases to represent the specific database members needed to target the performance collection selected. The aliases are made in QTEMP and are named by concatenating the library, file, and member names together into a single alias name. If you want to run these queries outside of Performance Data Investigator, the aliases will need to be recreated in your own interactive session.

Once the SQL statement has been modified and the perspective has been displayed again, you can then use the "Save As" action to save the updated perspective into a custom content package for future use.

The Modify SQL panel also features a Reset button which will reset the SQL statement to what it was when the panel was loaded. There is also a check box "Allow Collection Choice" which when checked (default behavior) will allow the query to run on any collection, not just your currently selected collection.

Related tasks

Creating and Editing a View
When design mode is enabled, an Edit View action is available if you are currently viewing a table or chart. It is also possible to create, edit, and add new views to an unlocked perspective.

Creating a Perspective
When design mode is enabled, a "New Perspective" button will appear on the main Investigate Data page as well as on the "Saving a custom perspective" panel. You can create a new custom perspective into an unlocked content package or perspective group of your choice.

Size Next Upgrade
Use the Size Next Upgrade action to send data from your current session to Workload Estimator for use in sizing a future system using current performance statistics.

The Size Next Upgrade action is available within a view perspective when investigating a Collection Services file collection. When selected, it will direct you to the IBM Systems Workload Estimator (WLE). The data sent to WLE will be retrieved from your current collection based on your current context. When you select the Size Next Upgrade action, you will be shown the following information:

- Filters - This will show you if filtering by Date/Time has been used to subset the data that will be sent to WLE.
- General information such as system name, operating system, version, and workload ID
- CPU metrics such as Utilization and Interactive Utilization
- Disk storage capacity

The data will be sent to the Workload Estimator (WLE) web application when you click the OK button and the IBM Systems Workload Estimator web application will be opened in a new window.

It is recommended you use PM for Power Systems if you want to size your upgrades based on historical growth trend data or seasonal peaks data. PM for Power Systems also facilitates sizings from multiple partitions.

Related concepts

IBM Performance Management for Power Systems (PM for Power Systems) - support for IBM i

The IBM Performance Management for Power Systems (PM for Power Systems) in support of IBM i offering automates the collection, archival, and analysis of system performance data and returns reports to help you manage system resources and capacity.

Change Context
The Change Context action allows you to inspect and modify the current context information for the chart or table view.

The information displayed on this panel represents the context mined from prior perspectives which affect the current table or chart. Altering the values will affect the current table or chart and future perspectives which mine the same data. It will not, however, affect any prior perspectives. If you exit the current perspective, any changes made using Change Context are lost, and the values will be reset to their prior, mined values. Typically, if you plan to continue analysis and drilldown, closing the current perspective and selecting new data points is more preferable to changing the context directly.

Related tasks

Save Perspective
The Save Perspective feature allows you to save a modified perspective for future use.

Creating and Editing a View
When design mode is enabled, an Edit View action is available if you are currently viewing a table or chart. It is also possible to create, edit, and add new views to an unlocked perspective.

A view is a single chart or table. A perspective is made up of one or more (up to twelve) views. You can create a new view by selecting an unlocked perspective from either the main Investigate Data page or the "Saving a custom perspective panel" and then selecting the "Edit" button. From the Views section on the Edit Perspective page, click on the "Add" button. This will open the Add View panel. From this panel, you can specify the following:

View

- Name
- Type of view (either table or chart)

Data Set

A data set is used to produce the view. It is defined by an SQL query. By clicking "Modify SQL" , another panel will open which will allow you to modify the SQL statement which produces the data for the view.

Drilldown

The Drilldown section displays a list of perspectives that are available to be added (or removed) as drilldown options for the view. Drilldown perspectives are shown in the view's action menu and allow you to examine other perspectives from the same context.

Chart Properties

- Transpose Axes - When the view is a chart, you may select the Transpose Axes checkbox if you want the domain and range to change axes (the domain will be displayed on the left instead of the bottom).
- Data Series - Occasionally the rendering of a chart may not be the way you want to see it. This could happen if data available in a table is not included in a chart, or possibly if no chart series is defined. You can do the following:
  - Add Data Series - Allows you to create a new data series to be added to the current chart view.
  - Edit Data Series - Allows you to view and modify an existing data series for the chart view. You can modify the data series attributes. The changes will be reflected on the chart view when you return to it from the Edit View page.
  - Delete - This will delete a data series from the chart view.
  - Move Up- Moves the selected data series up in the list of views.
  - Move Down - Moves the selected data series down in the list of views.
- Thresholds - Use this option to define a threshold to be used by the view. Thresholds can be added, edited, or deleted from a View using the corresponding buttons.

Related tasks

Modify SQL
The Modify SQL action allows you to inspect and modify the SQL statements used to retrieve the data from the performance collection in the current context.

Data Series
The Data Series panel allows you to view and modify the data series used for a chart view, or define a new data series to be used by a chart.

Thresholds
Thresholds provide a way to quickly glance at a chart and have a visual indicator of whether the values shown are within guidelines, or whether action should be taken.

Creating a Perspective
When design mode is enabled, a "New Perspective" button will appear on the main Investigate Data page as well as on the "Saving a custom perspective" panel. You can create a new custom perspective into an unlocked content package or perspective group of your choice.

Searching for a Perspective: Metric Finder
Use the "Search" button on the Investigate Data page to display the metric finder function which can help you find perspectives that contain a specific metric.

The metric finder format of the Investigate Data page is designed to allow a user to quickly display perspectives based on an interest in a specific metric or type of information. By selecting a metric from the extensive list of defined supported metrics, you will be able to see a list of the perspectives that provide views based on that area of information.

The metric finder format of the Investigate Data page has these sections that vary from the hierarchical format of the Investigate Data page:

- Metric - This metric find list allows you to display the list of perspectives that include a specific information type.
  - Filter - There are a large number of metrics available. The filter field of the Metric section allows you to limit the list of metrics in the Metric list by matching text in the filter to the name of the metric. This filter will help you search for one specific metric, without knowing the exact name of it.
    - After entering filter text, you must click "Apply Filter" to cause the metric name list to be updated.
    - When the metric list is filtered, only metrics matching the filter (partially or entirely) will be included in the dropdown.
  - The list of metrics provided with no filtering is exactly equal to the metrics from all perspectives available from the hierarchical format of perspectives. In other words, you will be able to get to all the top-level perspectives through selection of the metric wanted as you would be able to if the same perspective was found in the hierarchical format of perspectives. This includes the customer package perspectives defined by a user. Perspectives that are only accessible via drilldown are not included in the list of perspectives.
- Perspective - The perspective list is shown as a table. The table includes one or more perspectives which match the metric selected above in the metric section. Here you can select the perspective you want to use to render your collections data.
  - The location of each perspective is shown in the chart entry by the format:

    Content Package –> Folder [–> subfolder] –> Perspective Name

Note: If a perspective contains a metric, it does not imply that a chart view of a data collection will include that metric by default. In some cases, forcing the "Show as Table" action will be required to find the metric once the perspective is displayed.

Creating a Custom Package
When design mode is enabled, you can create your own custom content packages. These packages can contain perspectives tailored to your needs.

The interactive ability to create custom charts and tables can be very useful to an advanced user who has a need to see performance data in a different manner from the content packages provided by IBM. A content package typically contains an independently defined set of perspectives with some common purpose. To create a new content package folder, click on the "New Package..." icon at the top of the main Investigate Data panel, or at the top of the perspectives list in the "Saving a custom perspective panel". This will launch a new panel where you can specify a name and description for your new package.

Once a package is created, you can use the Edit button to edit the package information. While in the Edit mode, you can also select a default perspective for your package (if there are perspectives currently defined in your package). Also, you can select the "Locked" checkbox if you wish to lock your package (and all of its descendants) so that it cannot be edited. If the package is left unlocked after creation, you can use the Edit button to edit the package information or the Delete button to delete it.

Any content packages you create will appear in the main perspective hierarchy list. Once a package is created, you can add new perspective groups and perspectives to it.

Creating a Folder
When design mode is enabled, a "New folder" button will appear on the main Investigate Data page as well as on the "Saving a custom perspective" panel. This will allow you to organize any perspectives created into logical groupings.

To create a perspective group, you must have either an unlocked content package or perspective group selected in the tree. Clicking the "New folder" button will launch a panel that will allow you to type in a name and description for the new folder. If a content package is selected when the "New folder" action is taken, the perspective group will be created under the selected content package. If the "New folder" action is taken while a perspective group is selected, the new perspective group will be nested below the selected perspective group.

Once a perspective group is created, you can use the Edit button to edit the folder information. While in the Edit mode, you can also select a default perspective for your perspective group (if there are perspectives currently defined in your package). Also, you can select the "Locked" check box if you want to lock your folder (and all of its descendants) so that it cannot be edited. If the folder is left unlocked after creation, you can use the Edit button to edit the perspective group information or the Delete button to delete it.

Any perspective groups you create will appear in the main perspective hierarchy list under the associated content package. Once a perspective group is created, you can add new perspective groups and perspectives to it. You may also choose to use the Move Up or Move Down buttons to arrange the order of the perspective groups in your content package.

Creating a Perspective
When design mode is enabled, a "New Perspective" button will appear on the main Investigate Data page as well as on the "Saving a custom perspective" panel. You can create a new custom perspective into an unlocked content package or perspective group of your choice.

A perspective is one rendered panel of data, typically in the form of one or more charts or tables. A perspective contains one or more (up to twelve) views. A view is a single chart or table. To create a perspective, you must have either an unlocked content package or perspective group selected in the perspective tree. You can then select the "New Perspective" button. This will launch a new panel where you can specify a name and description for your perspective. You can select the "Locked" checkbox if you wish to lock your perspective so that it cannot be edited. Also on this panel, you can add views to the perspective.

If a perspective is created unlocked, you can use the "Edit" button to edit the perspective information. While in edit mode, you can change the name, description, or locked status (from unlocked to locked). You can also add, edit, or delete views associated with the perspective. An unlocked perspective can also be deleted using the "Delete" button.

Performance Data Investigator uses XML files to store how perspectives are defined (such as how it mines and renders data). The files are referred to as "Performance Markup Language" (PML). You can view or edit the PML directly for an unlocked perspective by clicking on the "Advanced Edit" button from the main Investigate Data page as well as on the "Saving a custom perspective" panel.

Any perspectives you create will appear in the main perspective hierarchy list under the associated content package or perspective group.

Related tasks

Modify SQL
The Modify SQL action allows you to inspect and modify the SQL statements used to retrieve the data from the performance collection in the current context.

Creating and Editing a View
When design mode is enabled, an Edit View action is available if you are currently viewing a table or chart. It is also possible to create, edit, and add new views to an unlocked perspective.

Data Series
The Data Series panel allows you to view and modify the data series used for a chart view, or define a new data series to be used by a chart.

The Data Series attributes are as follows. Some attributes may be locked in the case where the chart already has the attribute specified.

- Domain - Specifies the field to be used for the independent axis used for this chart. If another data series exists for the current chart, this field is locked to match the existing domain value.
- Range - Allows you to specify Range values to be used for this chart. The Available list shows all possible ranges. The Add button adds the ranges selected in the Available list to the data series. The Remove button removes the ranges selected in the Selected table from the data series. Use the dropdown menus in the Selected table to specify the color, background, and pattern of each range.
- Type - Select the chart type (line or bar with variations).
- Breakdown - Each distinct value of the chosen breakdown field will produce a unique data series which represents all of the range values for that breakdown value. This is the mechanism used to create individual data series for each unique job on a system over time. For example, given the following data set:

  Table 1. Data set example

Interval Number	Job CPU	Job Name
1	10	Job A
1	15	Job B
2	20	Job A
2	30	Job B
3	30	Job A
3	45	Job B

  A line chart with a domain of Interval Number and a range of Job CPU, would result with the following data points: (1, 10), (1, 15), (2, 20), (2, 30), (3, 30), (3, 45).

  Specifying a breakdown dimension of Job Name, would produce two lines with the following data points: Series 1 = (1, 10), (2, 20), (3, 30) and Series 2 = (1, 15), (2, 30), (3, 45).

  It is a complicated, but powerful feature that may take a little experimenting to fully understand.
- Tooltip fields - By selecting fields from this list, each point in the data series will include in its tool tip the value of the fields you select for the current domain.

The data series functions are only available for chart views. It may be necessary to add a data series when switching from a table view to a chart view. If you try to select the action "Show as chart" while viewing a table that has does not have a data series defined, this action will require you to define a data series before continuing.

Related tasks

Creating and Editing a View
When design mode is enabled, an Edit View action is available if you are currently viewing a table or chart. It is also possible to create, edit, and add new views to an unlocked perspective.

Thresholds
Thresholds provide a way to quickly glance at a chart and have a visual indicator of whether the values shown are within guidelines, or whether action should be taken.

A threshold represents a boundary which, once crossed, indicates the data has reached a new state. Threshold values are persistent and are stored according to user profile. The following can be specified for a threshold:

- Name - The name to be displayed for this threshold
- Field - This is the field for which this threshold is defined.
- Color - This specifies the color to be used when drawing this threshold on the chart.
- Current Value - The current value represents the threshold value currently specified by the user. This value can be easily changed, and will persist across application sessions. Future thresholds defined with the same name and field will also use the same value. To reset the current value to the default value, click the "Reset to Default Value" button.
- Default Value - The default value represents the value supplied when the content package was created. This value will be used when the user does not intentionally override it by specifying a Current Value. To force the default value to a new value for this threshold, click "Update to Current Value".

Related tasks

Creating and Editing a View
When design mode is enabled, an Edit View action is available if you are currently viewing a table or chart. It is also possible to create, edit, and add new views to an unlocked perspective.

Graph History
Graph History provides a graphical display of the historical performance data collection created by Collection Services.

The **Performance** > **Graph History** task provides a graphical view of the historical performance data collection created by Collection Services. You can view historical performance data by using the **Graph History** task in IBM Navigator for i. To view the historical data, you must use Collection Services to collect data and enable historical data creation in the Collection Services configuration.

Graph History concepts
Configure and generate historical performance data for use by the **Graph History** task in IBM Navigator for i.

Graph History displays the data contained in the historical performance data collection created by Collection Services. Therefore, the type and amount of data available depends on your Collection Services configuration. The amount of data that is available to be graphed is determined by the historical summary and detail data retention period settings that you selected from the Collection Services properties.

When historical data creation is enabled in the Collection Services configuration, historical data is automatically created for you at Collection Services cycle time. You also can select to create historical data at any time for a selected Collection Services management collection object. The Collection Services management collection object must contain data that was collected on the same partition and that falls within the retention periods you configured for keeping historical data.

The historical performance data collection consists of the following sets of data:

- **Summary data**

Summary data is the system level or summarized metrics that are useful in identifying trends or detecting changes in a system over a long time. Summary data is created when the option to create historical data is turned on.

- **Detail data**

  Detail data is the data from which metrics are derived and other relevant supplementary data. Detail data is created when the historical detail data creation option is also on. The creation of detail data is on by default, but can be turned off. This data is used to look deeper into a problem that was identified by looking at summary historical data. Only the top number of contributors for each metric are stored as historical detail data, as defined by the value of the historical detail data filter.

The amount of historical data that you are allowed to keep depends on whether PM for Power Systems (PM Agent) is active.

- If PM for Power Systems is not active, you can keep up to 7 days of detail data and 1 month of summary data.
- If PM for Power Systems is active, then you can keep up to 60 days of detail data and 50 years of summary data.

Related concepts

Collection Services support for historical data
Collection Services can be configured to create and maintain a historical performance data collection. The **Performance** > **Graph History** task in IBM Navigator for i can be used to analyze the historical performance data in chart or table form.

Related tasks

Configuring historical data for Graph History
Configure the historical performance data settings in Collection Services to generate historical performance data for use by the **Graph History** task in IBM Navigator for i.

Creating historical data for Graph History
To create historical performance data immediately for use in **Graph History**, follow these steps.

Viewing historical data with Graph History
Use the **Graph History** task in IBM Navigator for i to view historical performance data.

Activating PM Agent
PM Agent is a part of the operating system and you must activate it to use its collecting capabilities.

Related reference

Historical data files

Configuring historical data for Graph History
Configure the historical performance data settings in Collection Services to generate historical performance data for use by the **Graph History** task in IBM Navigator for i.

The **Graph History** task is included in IBM Navigator for i and is used to display the historical performance data collection created by Collection Services. To configure the historical performance data settings in Collection Services, follow these steps:

1. Select **Performance** > **All Tasks** > **Collectors** > **Collection Services** from your IBM Navigator for i window.
2. Click **Configure Collection Services**.
3. On the **General** tab, select **Create historical data when collection is cycled**.
4. Click the **Data Retention** tab.
5. Select **PM Agent on** to turn on PM for Power Systems (PM Agent).
6. Make any wanted changes to the historical data retention periods.
7. Click the **Historical Data** tab.
8. Make any wanted changes to other historical data settings, including interval, detail data creation, or detail data filter.

9. Click **OK**.

Historical data is now automatically created for you at every Collection Services cycle time.

Related concepts

Graph History concepts
Configure and generate historical performance data for use by the **Graph History** task in IBM Navigator for i.

Collection Services support for historical data
Collection Services can be configured to create and maintain a historical performance data collection. The **Performance** > **Graph History** task in IBM Navigator for i can be used to analyze the historical performance data in chart or table form.

Related tasks

Creating historical data for Graph History
To create historical performance data immediately for use in **Graph History**, follow these steps.

Viewing historical data with Graph History
Use the **Graph History** task in IBM Navigator for i to view historical performance data.

Activating PM Agent
PM Agent is a part of the operating system and you must activate it to use its collecting capabilities.

Creating historical data for Graph History
To create historical performance data immediately for use in **Graph History**, follow these steps.

Historical data for the currently active Collection Services collection is not put into the historical performance data collection automatically until the collection cycles. By manually creating historical data, you can view the latest collected data in **Graph History** by manually creating historical data. Manually create historical data by following these steps:

1. Select **Performance** > **Manage Collections** from your IBM Navigator for i window.
2. Select a Collection Services *MGTCOL object based collection that you want to create historical performance data for.
3. From the **Actions** menu, select **Create Historical Data**.

Cycling Collection Services also allows all data up to the time of the cycle to be added to the historical performance data collection. Cycle Collection Services by following these steps:

1. Select **Performance** > **All Tasks** > **Collectors** > **Collection Services** from your IBM Navigator for i window.
2. Click **Cycle Collection Services**.

Related concepts

Graph History concepts
Configure and generate historical performance data for use by the **Graph History** task in IBM Navigator for i.

Collection Services support for historical data
Collection Services can be configured to create and maintain a historical performance data collection. The **Performance** > **Graph History** task in IBM Navigator for i can be used to analyze the historical performance data in chart or table form.

Related tasks

Configuring historical data for Graph History
Configure the historical performance data settings in Collection Services to generate historical performance data for use by the **Graph History** task in IBM Navigator for i.

Viewing historical data with Graph History
Use the **Graph History** task in IBM Navigator for i to view historical performance data.

Related reference

Historical data files

Viewing historical data with Graph History
Use the **Graph History** task in IBM Navigator for i to view historical performance data.

The **Graph History** task is included in IBM Navigator for i and is used to display the historical performance data collection created by Collection Services. To view the historical performance data, follow these steps:

1. Select **Performance** > **Graph History** from your IBM Navigator for i window. An initial chart is displayed based on the default settings.
2. Select what you want to display by changing the fields in the context pane.
3. Click **Refresh** to display the graph.

After you launch **Graph History**, the chart displays a series of graphed collection points. These collection points on the graph line are identified by two different graphics that correspond to the two levels of data:

- A square collection point represents a data point where detail data is available for displaying the top contributors and properties information.
- A circular collection point represents a data point where only summary data is available, so no top contributors or properties information can be shown.

To view the top contributors of a particular data point, select a square data point to display the **Top Contributors** chart. To view the properties of one of the top contributors, select the bar next to a top contributor to display the **Properties** pane for that top contributor.

Related concepts

Graph History concepts
Configure and generate historical performance data for use by the **Graph History** task in IBM Navigator for i.

Collection Services support for historical data
Collection Services can be configured to create and maintain a historical performance data collection. The **Performance** > **Graph History** task in IBM Navigator for i can be used to analyze the historical performance data in chart or table form.

Related tasks

Configuring historical data for Graph History
Configure the historical performance data settings in Collection Services to generate historical performance data for use by the **Graph History** task in IBM Navigator for i.

Creating historical data for Graph History
To create historical performance data immediately for use in **Graph History**, follow these steps.

Performance Data Reports
Create a group of printable or online tables and graphs of performance data by using IBM Navigator for i.

A Performance Data Report can be generated for a set of PDI perspectives. You can view the report in IBM Navigator for i or generate a PDF or zip file. By using Performance Data Reports, you can efficiently export multiple charts or tables for a collection at one time.

Performance report definitions
A performance report definition is a predefined set of Performance Data Investigator perspectives used to create a report.

A performance report definition can be used to generate a PDF or zip file that contains graphs of the data that is contained in the collection. Generating a PDF or zip file makes it easy to export multiple charts or tables for a collection at one time. Several predefined performance data report definitions are provided and you can also create your own definitions that are tailored to your specific needs.

Viewing report definitions
To view performance data report definitions, follow these steps.

1. Select **Performance** > **All Tasks** > **Performance Data Reports** from your IBM Navigator for i window.
2. Click **Report Definitions**.

Adding a report definition
To add a performance data report definition, follow these steps.

1. Select **Performance** > **All Tasks** > **Performance Data Reports** from your IBM Navigator for i window.
2. Click **Add Definition**.

Deleting a report definition
To delete a performance data report definition, follow these steps.

1. Select **Performance** > **All Tasks** > **Performance Data Reports** from your IBM Navigator for i window.
2. Click **Delete Definition**.

Creating a performance data report
To create a performance data report, follow these steps.

1. Select **Performance** > **All Tasks** > **Performance Data Reports** > **Report Definitions** from your IBM Navigator for i window.
2. Select the report definition that you want to create.
3. From the **Actions** menu, select **Create Performance Data Report**.

Manage collections
Select the **Manage Collections** task to launch the collections table to view and work with the collections on your system. The collections from Collection Services, IBM i Job Watcher, IBM i Disk Watcher, Performance Explorer, and Batch Model are visible here.

Related reference

CL commands for performanceThe operating system includes several CL commands to help you manage and maintain system performance.

Viewing a collection
To view a collection in IBM Navigator for i, follow one of the following series of steps.

1. Select **Performance** > **Investigate Data** from your IBM Navigator for i window.
2. Expand the content package that you are interested in.
3. Keep expanding the nodes in the tree until you navigate to the perspective that you would like to use.
4. Select the perspective.
5. Select the collection library.
6. Select the collection name.
7. Click **Display**.

You can also view a Collection Services, Disk Watcher, Job Watcher, or Performance Explorer file based collection by doing the following steps:

1. Select **Performance** > **Manage Collections** from your IBM Navigator for i window. This action launches the list of data collections on your system.
2. Select the collection that you want to view.
3. From the **Actions** menu, select **Investigate Data**. This action launches the Performance Data Investigator tool. The selected collection data is rendered using the default perspective that is defined by the content package.

You can also view a Batch Model file based collection by doing the following steps:

1. Select **Performance** > **All Tasks** > **Sizing** > **Batch Model** > **Batch Models** from your IBM Navigator for i window. This action launches the list of data collections on your system.
2. Select a Batch Model file based collection with a status of **Complete** that you want to view.
3. From the **Actions** menu, select **Investigate Results**. This action launches the Performance Data Investigator tool. The selected collection data is rendered using the default perspective that is defined by the content package.

Related concepts

Investigate Data

Selecting the Investigate Data task launches the powerful Performance Data Investigator tool. With this tool, you can view and analyze data that is stored in performance collections in chart or table form.

Copying a collection

To copy a collection, follow these steps.

1. Select **Performance** > **Manage Collections** from your IBM Navigator for i window.
2. Select the collection that you want to copy.
3. From the **Actions** menu, select **Copy**.

Related reference

Copy Performance Collection (CPYPFRCOL) commandSee the Copy Performance Collection (CPYPFRCOL) command for information about copying a performance collection.

Deleting a collection

To delete a collection, follow these steps.

1. Select **Performance** > **Manage Collections** from your IBM Navigator for i window.
2. Select the collection that you want to delete.
3. From the **Actions** menu, select **Delete**.

Related reference

Delete Performance Collection (DLTPFRCOL) commandSee the Delete Performance Collection (DLTPFRCOL) command for information about deleting a performance collection.

Saving a collection

To save a collection, follow these steps.

1. Select **Performance** > **Manage Collections** from your IBM Navigator for i window.
2. Select the collection that you want to save.
3. From the **Actions** menu, select **Save**.

Related reference

Save Performance Collection (SAVPFRCOL) commandSee the Save Performance Collection (SAVPFRCOL) command for information about saving a performance collection.

Restoring a collection

To restore a collection, follow these steps.

1. Select **Performance** > **Manage Collections** from your IBM Navigator for i window.
2. From the **Actions** menu, select **Maintain Collections** > **Restore**.

Related reference

Restore Performance Collection (RSTPFRCOL) commandSee the Restore Performance Collection (RSTPFRCOL) command for information about restoring a performance collection.

Converting a collection

To convert a collection that was collected in a previous release, follow these steps.

1. Select **Performance** > **Manage Collections** from your IBM Navigator for i window.
2. Select the collection that you want to convert.
3. From the **Actions** menu, select **Maintain Collections** > **Convert**.

Related reference

Convert Performance Collection (CVTPFRCOL) commandSee the Convert Performance Collection (CVTPFRCOL) command for information about converting a performance collection.

Viewing collection properties
To view collection properties, follow these steps.

1. Select **Performance** > **Manage Collections** from your IBM Navigator for i window.
2. Select the collection that you want to view the properties for.
3. From the **Actions** menu, select **Properties**.

Rebuilding the collection table
To rebuild the collection table, follow these steps.

1. Select **Performance** > **Manage Collections** from your IBM Navigator for i window.
2. From the **Actions** menu, select **Maintain Collections** > **Rebuild Collection Table**.

Managing Collection Services
Manage Collection Services by using IBM Navigator for i.

Related reference

CL commands for performanceThe operating system includes several CL commands to help you manage and maintain system performance.

Starting Collection Services
Start Collection Services by doing the following.

To start Collection Services, follow these steps:

1. Select **Performance** > **All Tasks** > **Collectors** > **Collection Services** from your IBM Navigator for i window.
2. Click **Start Collection Services**.

Related reference

Start Performance Collection (STRPFRCOL) commandSee the Start Performance Collection (STRPFRCOL) command for information about starting Collection Services.

Stopping Collection Services
Stop Collection Services by doing the following.

To stop Collection Services, follow these steps:

1. Select **Performance** > **All Tasks** > **Collectors** > **Collection Services** from your IBM Navigator for i window.
2. Click **Stop Collection Services**.

Related reference

End Performance Collection (ENDPFRCOL) commandSee the End Performance Collection (ENDPFRCOL) command for information about stopping Collection Services.

Configuring Collection Services
Configure Collection Services by doing the following.

To configure Collection Services, follow these steps:

1. Select **Performance** > **All Tasks** > **Collectors** > **Collection Services** from your IBM Navigator for i window.
2. Click **Configure Collection Services**.

   Specifying summary data to be generated when the collection is cycled facilitates quicker processing of the performance database data by analysis tools.

Related tasks

Customizing data collections

When you use Collection Services to collect performance data, you control what data is collected and how often it is collected.

Related reference

Configure Performance Collection (CFGPFRCOL) commandSee the Configure Performance Collection (CFGPFRCOL) command for information about configuring Collection Services.

Creating a Collection Services custom profile

Create a custom Collection Services profile by doing the following.

When you create a custom Collection Services profile, you can select from a list of available data categories, such as system, job, disk storage, or memory. You can specify how often the data is collected for each category of data that you collect in your custom profile.

To create a custom Collection Services profile, follow these steps:

1. Select **Performance** > **All Tasks** > **Collectors** > **Collection Services** from your IBM Navigator for i window.
2. Click **Configure Collection Services**.
3. Click the **Data to Collect** tab.
4. Select **Customize collection profile**.
5. Select the categories that you want to add to your custom profile from **Available categories**. To add the categories, click **Add**. The categories are then listed under **Categories to collect**.
6. Configure a category's data collection frequency by selecting **Configure** next to the category name in the **Categories to collect** list. This action opens the **Configure Category** page.
   a) On the **Configure Category** page, specify the frequency to collect the category data.
   b) Click **OK**.
7. Click **OK** to complete creating the Collection Services custom profile.

The Collection Services custom profile that is created is not used by Collection Services immediately. You must wait until Collection Services cycles. If you want to use the custom profile immediately, force Collection Services to cycle by following these steps:

1. Select **Performance** > **All Tasks** > **Collectors** > **Collection Services** from your IBM Navigator for i window.
2. Click **Cycle Collection Services**.

Related tasks

Customizing data collections

When you use Collection Services to collect performance data, you control what data is collected and how often it is collected.

Cycling Collection Services

Cycle Collection Services by doing the following.

To cycle Collection Services, follow these steps:

1. Select **Performance** > **All Tasks** > **Collectors** > **Collection Services** from your IBM Navigator for i window.
2. Click **Cycle Collection Services**.

Creating database files

To create database files, follow these steps.

1. Select **Performance** > **Manage Collections** from your IBM Navigator for i window.
2. Select a Collection Services *MGTCOL object based collection that you want to create performance data for.
3. From the **Actions** menu, select **Create data**.

Related tasks

Creating database files from Collection Services data
Use this information to manually or automatically create database files from Collection Services data.

Related reference

Create Performance Data (CRTPFRDTA) commandSee the Create Performance Data (CRTPFRDTA) command for information about creating Collection Services database files.

Displaying Collection Services status
Display Collection Services status by doing the following.

To display Collection Services status, follow these steps:

1. Select **Performance** > **All Tasks** > **Collectors** > **Collection Services** from your IBM Navigator for i window.
2. Click **Collection Services Status**.

Related reference

Check Performance Collection (CHKPFRCOL) commandSee the Check Performance Collection (CHKPFRCOL) command for information about displaying the current status of Collection Services.

Managing IBM i Disk Watcher
Manage IBM i Disk Watcher by using IBM Navigator for i.

The IBM i Disk Watcher functions and content package require the installation of IBM Performance Tools for i (5770-PT1) Option 1 - Manager Feature.

Related concepts

IBM i Disk Watcher
IBM i Disk Watcher provides for the collection of disk performance data to diagnose disk related performance problems.

Related reference

CL commands for performanceThe operating system includes several CL commands to help you manage and maintain system performance.

Starting Disk Watcher
Start Disk Watcher by doing the following.

To start Disk Watcher, follow these steps:

1. Select **Performance** > **All Tasks** > **Collectors** > **Disk Watcher** from your IBM Navigator for i window.
2. Click **Start Disk Watcher**.

Related reference

Start Disk Watcher (STRDW)See the Start Disk Watcher (STRDW) command for information about starting a Disk Watcher collection.

Stopping Disk Watcher
Stop Disk Watcher by doing the following.

To stop Disk Watcher, follow these steps:

1. Select **Performance** > **All Tasks** > **Collectors** > **Disk Watcher** from your IBM Navigator for i window.
2. Click **Stop Disk Watcher**.

Related reference

End Disk Watcher (ENDDW)See the End Disk Watcher (ENDDW) command for information about ending a Disk Watcher collection.

Adding a Disk Watcher definition
Add a Disk Watcher definition by doing the following.

To add a Disk Watcher definition, follow these steps:

1. Select **Performance** > **All Tasks** > **Collectors** > **Disk Watcher** from your IBM Navigator for i window.
2. Click **Add Disk Watcher Definition**.

Related reference

Add Disk Watcher Definition (ADDDWDFN)See the Add Disk Watcher Definition (ADDDWDFN) command for information about adding a Disk Watcher definition from the system.

Deleting a Disk Watcher definition

Delete a Disk Watcher definition by doing the following.

To delete a Disk Watcher definition, follow these steps:

1. Select **Performance** > **All Tasks** > **Collectors** > **Disk Watcher** from your IBM Navigator for i window.
2. Click **Disk Watcher Definitions**.
3. Select the Disk Watcher definition that you want to delete.
4. From the **Actions** menu, select **Delete**.

Related reference

Remove Disk Watcher Definition (RMVDWDFN)See the Remove Disk Watcher Definition (RMVDWDFN) command for information about removing a Disk Watcher definition from the system.

Displaying the properties of a Disk Watcher definition

Display the properties of a Disk Watcher definition by doing the following.

To display the properties of a Disk Watcher definition, follow these steps:

1. Select **Performance** > **All Tasks** > **Collectors** > **Disk Watcher** from your IBM Navigator for i window.
2. Click **Disk Watcher Definitions**.
3. Select the Disk Watcher definition that you want to display the properties for.
4. From the **Actions** menu, select **Properties**.

Managing IBM i Job Watcher

Manage IBM i Job Watcher by using IBM Navigator for i.

The IBM i Job Watcher functions and content package require the installation of IBM Performance Tools for i (5770-PT1) Option 3 - Job Watcher.

Related concepts

IBM i Job Watcher

IBM i Job Watcher provides for the collection of job data for any or all jobs, threads, and tasks on the system. It provides call stacks, SQL statements, objects being waited on, Java JVM statistics, wait statistics and more which are used to diagnose job related performance problems.

Related reference

CL commands for performanceThe operating system includes several CL commands to help you manage and maintain system performance.

Starting Job Watcher

Start Job Watcher by doing the following.

To start Job Watcher, follow these steps:

1. Select **Performance** > **All Tasks** > **Collectors** > **Job Watcher** from your IBM Navigator for i window.
2. Click **Start Job Watcher**.

Related reference

Start Job Watcher (STRJW)See the Start Job Watcher (STRJW) command for information about starting a Job Watcher collection.

Stopping Job Watcher

Stop Job Watcher by doing the following.

To stop Job Watcher, follow these steps:

1. Select **Performance** > **All Tasks** > **Collectors** > **Job Watcher** from your IBM Navigator for i window.
2. Click **Stop Job Watcher**.

Related reference

End Job Watcher (ENDJW)See the End Job Watcher (ENDJW) command for information about ending a Job Watcher collection.

Adding a Job Watcher definition

Add a Job Watcher definition by doing the following.

To add a Job Watcher definition, follow these steps:

1. Select **Performance** > **All Tasks** > **Collectors** > **Job Watcher** from your IBM Navigator for i window.
2. Click **Add Job Watcher Definition**.

Related reference

Add Job Watcher Definition (ADDJWDFN)See the Add Job Watcher Definition (ADDJWDFN) command for information about specifying the performance data that is to be collected during a Job Watcher collection.

Deleting a Job Watcher definition

Delete a Job Watcher definition by doing the following.

To delete a Job Watcher definition, follow these steps:

1. Select **Performance** > **All Tasks** > **Collectors** > **Job Watcher** from your IBM Navigator for i window.
2. Click **Job Watcher Definitions**.
3. Select the Job Watcher definition that you want to delete.
4. From the **Actions** menu, select **Delete**.

Related reference

Remove Job Watcher Definition (RMVJWDFN)See the Remove Job Watcher Definition (RMVJWDFN) command for information about removing a Job Watcher definition from the system.

Displaying the properties of a Job Watcher definition

Display the properties of a Job Watcher definition by doing the following.

To display the properties of a Job Watcher definition, follow these steps:

1. Select **Performance** > **All Tasks** > **Collectors** > **Job Watcher** from your IBM Navigator for i window.
2. Click **Job Watcher Definitions**.
3. Select the Job Watcher definition that you want to display the properties for.
4. From the **Actions** menu, select **Properties**.

Batch Model

The Batch Model tool models the system utilization and run times of IBM i batch workloads.

You can use the Batch Model tool to help analyze and predict batch job performance on the IBM i and help answer the question: “What can I do to my system to meet my overnight batch runtime requirements (also known as the batch window)?”

A Batch Model collection is created based on existing performance data that is collected by IBM i Collection Services. A Batch Model collection can then be changed and analyzed for various “what if” scenarios, such as workload growth and hardware upgrades.

Investigate the results of Batch Model to view the batch window in a form that shows workload start/stop times, dependencies between workloads, and amount of resources used. View the Batch Model perspectives to locate times in the batch window when more efficient job scheduling can improve total system throughput.

The Batch Model functions and content package require the installation of IBM Performance Tools for i (5770-PT1) Option 1 - Manager Feature.

Creating a batch model
A Batch Model collection is created based on existing performance data that is collected by IBM i Collection Services. During the process of creating a batch model, the measured data is analyzed and a model is built over the measured data.

To create a batch model, follow these steps.

1. Select **Performance** > **Manage Collections** from your IBM Navigator for i window.
2. Select a Collection Services file based collection that you want to create a batch model for.
3. From the **Actions** menu, select **Create Batch Model**.

You can also create a batch model by following these steps:

1. Select **Performance** > **All Tasks** > **Sizing** > **Batch Model** from your IBM Navigator for i window.
2. Click **Create Batch Model**.

Note: The **Create Batch Model** action submits a batch job to create the model. Creating a batch model can sometimes be a long running operation. The creation is done when the status of the Batch Model collection is in **Complete** state.

Investigating Batch Model results
Investigate Batch Model results to view the batch window in a form that shows workload start/stop times, dependencies between workloads, and amount of resources used.

To view the results of a Batch Model file based collection, follow these steps.

1. Select **Performance** > **Investigate Data** > **Batch Model** from your IBM Navigator for i window.
2. Select the perspective that you would like to view.
3. Select the collection library.
4. Select the collection name.
5. Click **Display**.

You can also view the results of a Batch Model file based collection by doing the following steps:

1. Select **Performance** > **All Tasks** > **Sizing** > **Batch Model** > **Batch Models** from your IBM Navigator for i window. This action launches the list of data collections on your system.
2. Select a Batch Model file based collection with a status of **Complete** that you want to view.
3. From the **Actions** menu, select **Investigate Results**. This action launches the Performance Data Investigator tool. The selected collection data is rendered using the default perspective that is defined by the content package.

Changing the calibration of a batch model
Change the calibration of a batch model to make corrections to the model if the measured data was modeled incorrectly.

To change the calibration of a batch model, follow these steps.

1. Select **Performance** > **Manage Collections** from your IBM Navigator for i window.
2. Select a Batch Model file based collection with a status of **Complete** that you want to change the calibration of.
3. From the **Actions** menu, select **Change Calibration**.

You can also change the calibration of a batch model by doing the following steps:

1. Select **Performance** > **All Tasks** > **Sizing** > **Batch Model** from your IBM Navigator for i window.
2. Click **Change Batch Model Calibration**.

Calibrating a batch model
Calibrating a batch model must be done to re-create the model after changes were made to the batch model calibration.

To calibrate a batch model, follow these steps.

1. Select **Performance** > **Manage Collections** from your IBM Navigator for i window.
2. Select a Batch Model file based collection with a status of **Calibration Changed** that you want to calibrate.
3. From the **Actions** menu, select **Calibrate**.

You can also calibrate a batch model by doing the following steps:

1. Select **Performance** > **All Tasks** > **Sizing** > **Batch Model** from your IBM Navigator for i window.
2. Click **Calibrate Batch Model**.

Note: The **Calibrate Batch Model** action submits a batch job to re-create the model. Calibrating a batch model can sometimes be a long running operation. The calibration is done when the status of the Batch Model collection is in **Complete** state.

Changing a batch model
Change a batch model to model various “what if” scenarios.

You can change a batch model to model the following scenarios:

- Overall system workload growth
- Processor upgrade
- Disk upgrade
- Changes to individual workloads

To change a batch model, follow these steps.

1. Select **Performance** > **Manage Collections** from your IBM Navigator for i window.
2. Select a Batch Model file based collection with a status of **Complete** that you want to change.
3. From the **Actions** menu, select **Change Model**.

You can also change a batch model by doing the following steps:

1. Select **Performance** > **All Tasks** > **Sizing** > **Batch Model** from your IBM Navigator for i window.
2. Click **Change Batch Model**.

Analyzing a batch model
Analyzing a batch model must be done to re-create the model after changes were made to the batch model.

To analyze a batch model, follow these steps.

1. Select **Performance** > **Manage Collections** from your IBM Navigator for i window.
2. Select a Batch Model file based collection with a status of **Model Changed** that you want to analyze.
3. From the **Actions** menu, select **Analyze**.

You can also analyze a batch model by doing the following steps:

1. Select **Performance** > **All Tasks** > **Sizing** > **Batch Model** from your IBM Navigator for i window.
2. Click **Analyze Batch Model**.

Note: The **Analyze Batch Model** action submits a batch job to re-create the model. Analyzing a batch model can sometimes be a long running operation. The analysis is done when the status of the Batch Model collection is in **Complete** state.

Merging batch models
Merge batch models to merge all the workloads from two different data collections. This action is helpful when data needs to be merged that was collected during different time periods or from different systems.

To merge two batch models, follow these steps.

1. Select **Performance** > **Manage Collections** from your IBM Navigator for i window.
2. Select a Batch Model file based collection that you want to merge.
3. From the **Actions** menu, select **Merge**.

You can also merge two batch models by doing the following steps:

1. Select **Performance** > **All Tasks** > **Sizing** > **Batch Model** from your IBM Navigator for i window.
2. Click **Merge Batch Model**.

Resetting a batch model
Reset a batch model to change the status of your Batch Model collection to Reset.

If your Batch Model collection gets into an **Error** status, you must reset your batch model to set the status to **Reset**. Set the status to **Reset** to try the operation that caused your Batch Model collection to end with an error again.

To reset a batch model, follow these steps.

1. Select **Performance** > **Manage Collections** from your IBM Navigator for i window.
2. Select a Batch Model file based collection with a status of **Error** that you want to reset.
3. From the **Actions** menu, select **Reset**.

You can also reset a batch model by doing the following steps:

1. Select **Performance** > **All Tasks** > **Sizing** > **Batch Model** from your IBM Navigator for i window.
2. Click **Reset Batch Model**.

IBM Navigator for i Monitors

IBM Navigator for i monitors track current information about the performance of your system. Additionally, you can use them to carry out predefined actions when a specific event occurs. Monitors continue to monitor and perform any threshold commands or actions you specified until you stop the monitor.

IBM Navigator for i provides the following types of monitors:

System monitor

Collect and display performance data as it happens. System monitors use Collection Services data to track the elements of system performance of specific interest to you. Detailed graphs help you visualize what is going on with your system as it happens. Choose from various metrics (performance measurements) to pinpoint specific aspects of system performance.

Message monitor

Find out whether your application completes successfully or monitor for specific messages that are critical to your business needs. You can see the details of a message, reply to a message, send a message, delete a message, and view or change the properties of a message.

Management Central Monitors (7.1)

If you are still using the System i Navigator client, you can find out more information about Management Central monitors at the link above. Release 7.1 was the final release of System i Navigator.

Related concepts

Collection Services

Collection Services provides for the collection of system and job level performance data. It is the primary collector of performance data.

IBM Navigator for iSee the IBM Navigator for i topic to learn about more functions available within IBM Navigator for i.

Monitor concepts
Monitors track near real-time performance data. Additionally, they continually monitor your system to run a selected command when a specified threshold is reached. Find out how monitors work, what they can monitor, and how they can respond to a performance situation.

System monitors use performance metrics that are stored in database files that are generated and maintained by Collection Services. You can use Performance Data Investigator to view the performance data that is gathered by the monitor. You can change the frequency of the data collection in the monitor properties. The settings in the monitor properties override the settings in Collection Services if the monitor requires the data to be collected more frequently.

Note: When monitors override the settings of Collection Services to gather performance data more frequently, the settings are not undone when a monitor stops. You must manually change the Collection Services settings if you no longer want to collect the data as often.

You can use monitors to track and research many different elements of system performance and can have many different monitors that run simultaneously. Using multiple monitors together can provide a sophisticated tool for observing and managing system performance. For example, when a new interactive application is implemented, you might use a system monitor to prioritize a job's resource utilization and a message monitor to alert you if a specified message occurs.

Monitors must be manually restarted after a partition IPL, they will not automatically restart.

Setting thresholds and actions

When you create a new monitor, you can specify actions that you want to occur when a system metric reaches a specified threshold level or a message occurs. When threshold levels or messages occur, you can choose to run an IBM i command on the target system, such as sending a message or holding a job queue.

Related concepts
Collection Services
Collection Services provides for the collection of system and job level performance data. It is the primary collector of performance data.

Collection Services support for system monitoring
Collection Services now fully supports the concept of system monitoring. Collection Services can be configured to collect system monitor performance metrics, even without configuring and running a IBM Navigator for i System Monitor.

Related tasks
Performance Data Investigator
Performance Data Investigator provides a web-based GUI over performance data with interactive charts and tables. You can view and analyze performance data for each of the collectors (Collection Services, IBM i Job Watcher, IBM i Disk Watcher, Performance Explorer, Database SQL Plan Cache, and Database SQL Performance Monitor).

Related reference
Collection Services data files: QAPMSMCMNThis Collection Services database file contains summarized metrics from communication protocol data (*CMNBASE collection category) that may be used in support of system monitoring.
Collection Services data files: QAPMSMDSKThis Collection Services database file contains summarized metrics from disk data (*DISK collection category) that may be used in support of system monitoring.
Collection Services data files: QAPMSMJMIThis Collection Services database file contains summarized metrics from job data (*JOBMI collection category) that may be used in support of system monitoring.

Collection Services data files: QAPMSMJOSThis Collection Services database file contains summarized metrics from job data (*JOBOS collection category) that may be used in support of system monitoring.

Collection Services data files: QAPMSMPOLThis Collection Services database file contains summarized metrics from pool data (*POOL collection category) that may be used in support of system monitoring.

Collection Services data files: QAPMSMSYSThis Collection Services database file contains summarized metrics from system data (*SYSLVL collection category) that may be used in support of system monitoring.

Creating a system monitor

Use this information to learn how to set up a system monitor in IBM Navigator for i and how to configure it to take the best advantage of the available options.

System monitors are highly interactive tools that you can use to gather and display near real-time performance data from your target system. To create a new system monitor, perform the following steps:

1. In IBM Navigator for i, select **Monitors** > **System Monitors**. From the **Actions** menu, select **Create New System Monitor...**.
2. Specify a monitor name. Use the **Create New System Monitor-General** page specify a name for your monitor. Provide a brief description so you can find the monitor in a list of monitors. Click **Next**.
3. Select metrics. Use the **Create New System Monitor-Metrics** page to select your metrics. To add a metric to the monitor, select it from the list of Available Metrics, and click **Add**. The metric is listed under **Metrics to monitor**.
4. View and change your metric properties. Use the **Create New System Monitor-Metrics** page to edit the properties of each metric by clicking the metric name in the **Metrics to monitor** list. This action opens the **Configure Metric** page where you can edit the properties of the metric.
   a) Set collection interval. Use the **Configure Metric** page to select the collection interval for the metric.
   b) Set thresholds. Use the **Configure Metric** page to enable thresholds and specify trigger and reset values.
   c) Set threshold actions. Use the **Configure Metric** page to specify the actions you want to occur when a metric threshold is triggered or reset.
   d) Click **OK** to save the metric properties.
5. Click **Next** to see the monitor summary page.
6. Click **Finish** to save the monitor.

After you create your monitor, right-click the monitor name and select **Start** to run the monitor and begin working with monitor graphs. The monitor will continue to run and perform any threshold commands or actions you specified until you stop the monitor.

Related tasks

Investigate system monitor data

Use this information to learn how to view system monitor performance data in Performance Data Investigator in IBM Navigator for i.

Monitor metrics

To effectively monitor system performance, you must decide which aspects of system performance you want to monitor. IBM Navigator for i offers various performance measurements, which are known as *metrics*, to help you pinpoint different aspects of system performance.

When you configure a monitor, you can use any metric, a group of metrics, or all the metrics from the list to be included in your monitor. Metric types that you can use in your monitor include the following.

Table 2.	
Metric groups	**Metric description**
CPU Utilization	The percentage of available processing unit time that is consumed by jobs on your system. Choose from the following types of CPU Utilization metrics for use in your monitors: • **CPU Utilization (Average):** Configured CPU percent unscaled. • **CPU Utilization (Interactive Jobs):** Configured CPU percent that is consumed by interactive jobs. • **CPU Utilization (Uncapped):** Uncapped CPU percent unscaled. The amount of unscaled system CPU consumed relative to the maximum uncapped CPU the partition could consume based on the number of the virtual processors that are assigned to the partition and the capacity of the shared virtual pool. • **CPU Utilization (SQL):** SQL CPU percent unscaled. The amount of unscaled system CPU consumed performing work that is done on behalf of SQL operations relative to the configured CPU time (processor units) available to the partition.
Interactive Response Time (Average and Maximum)	The response time that interactive jobs experience on your system.
Transaction Rate (Interactive)	The number of transactions per second completed on your system by interactive jobs (Job Type = 'I').
Batch Logical Database I/O	The average number of logical database input/output (I/O) operations that are currently performed by batch jobs (Job Type = 'B') on the system.
Disk Arm Utilization (Average and Maximum)	The disk unit percent busy for all disks.
Disk Arm Utilization for System ASP (Average and Maximum)	The disk unit percent busy for disks in the system ASP.
Disk Arm Utilization for User ASP (Average and Maximum)	The disk unit percent busy for all disks in user ASPs.
Disk Arm Utilization for Independent ASP (Average and Maximum)	The disk unit percent busy for disks in independent ASPs.
Disk Storage Utilization (Average and Maximum)	The percentage of disk storage that is full on your system during the time you collect the data.
Disk Storage Utilization for System ASP (Average and Maximum)	The percentage of disk storage that is full in the system ASP during the time you collect the data.
Disk Storage Utilization for User ASP (Average and Maximum)	The percentage of disk storage that is full in user ASPs during the time you collect the data.
Disk Storage Utilization for Independent ASP (Average and Maximum)	The percentage of disk storage that is full in independent ASPs during the time you collect the data.

Table 2. (continued)	
Metric groups	**Metric description**
Communications Line Utilization (Average and Maximum)	The amount of data that was sent and received on all your system communication lines.
LAN Utilization (Maximum and Average)	The amount of data that was sent and received on all your local area network (LAN) communication lines.
Machine Pool Faults	The number of faults per second occurring in the machine pool on the system.
User Pool Faults (Maximum and Average)	The number of faults per second per pool.
Temporary Storage Used	The total amount of temporary storage (megabytes) in use within the system. This metric includes both system and user temporary storage.
Spool File Creation Rate	The number of spool files that are created per second.
Shared Processor Pool Utilization (Virtual and Physical)	The amount of CPU consumed in the shared pool by all partitions that are using the pool relative to the CPU available within the pool.
Disk Response Time (Read and Write)	The response time that disk units experienced on your system.
HTTP Requests Received Rate	The number of requests received per second for all HTTP servers.
HTTP Requests Received (Maximum)	The largest number of HTTP requests received by a single server.
HTTP Responses Sent Rate	The number of responses sent per second for all HTTP servers.
HTTP Responses Sent (Maximum)	The largest number of HTTP responses sent by a single server.
HTTP Non-Cached Requests Processed (Average and Maximum)	The number of non-cached requests processed for HTTP servers.
HTTP Error Responses Sent (Average and Maximum)	The number of error responses sent for HTTP servers.
HTTP Non-Cached Requests Processing Time (Total and Highest Average)	The processing time for non-cached requests for HTTP servers.
HTTP Cached Requests Processing Time (Total and Highest Average)	The processing time for cached requests for HTTP servers.

If you need more help, click the **Help** button on the **Create New System Monitor-Metrics** page. After you become familiar with the IBM Navigator for i metrics, which metrics you select depend on the information needs of your computing environment. After you select metrics that target the information you are trying to see, you are ready to view and change detailed metric information for each metric you selected for your monitor.

Investigate system monitor data
Use this information to learn how to view system monitor performance data in Performance Data Investigator in IBM Navigator for i.

You can use system monitors to gather and display near real-time performance data from your system. System monitors use Performance Data Investigator to chart the performance data that is gathered by the monitor. To view system monitor performance data, perform the following steps:

1. In IBM Navigator for i, select **Monitors** > **System Monitors**.
2. Select the name of the monitor whose performance data you want to view. From the **Actions** menu, select **Investigate Monitor Data**.
3. Select the name of the perspective you want to view. This action launches the perspective in Performance Data Investigator.
4. Use the **Refresh** button to update the perspective to show new data as it is collected for a monitor that is active.
5. Click **Done** when you are done viewing the data.

Related tasks

Performance Data Investigator
Performance Data Investigator provides a web-based GUI over performance data with interactive charts and tables. You can view and analyze performance data for each of the collectors (Collection Services, IBM i Job Watcher, IBM i Disk Watcher, Performance Explorer, Database SQL Plan Cache, and Database SQL Performance Monitor).

Related reference

Monitor metrics
To effectively monitor system performance, you must decide which aspects of system performance you want to monitor. IBM Navigator for i offers various performance measurements, which are known as *metrics*, to help you pinpoint different aspects of system performance.

Creating a message monitor
Use this information to learn how to set up a message monitor in IBM Navigator for i and how to configure it to take the best advantage of the available options.

Message monitors can be used to find out whether your application completes successfully or monitor for specific messages that are critical to your business needs. To create a new message monitor, perform the following steps:

1. In IBM Navigator for i, select **Monitors** > **Message Monitors**. From the **Actions** menu, select **Create New Message Monitor**.
2. Specify a monitor name. Use the **Create New Message Monitor-General** page specify a name for your monitor. Provide a brief description so you can find the monitor in a list of monitors. Click **Next**.
3. Select message queue. Use the **Create New Message Monitor-Message Queue** page to select the message queue you want to monitor. Click **Next**.
4. Add message sets. Use the **Create New Message Monitor-Message Set** page to add message sets to the monitor. To add a message set, click **Add**.
5. Define actions. Use the **Create New Message Monitor-Message Set** page to define what actions to take when a message in the message set occurs. Click **Next**.
6. Configure actions. Use the **Create New Message Monitor-Actions** page to configure when to apply thresholds and actions.
7. Click **Next** to see the monitor summary page.
8. Click **Finish** to save the monitor.

After you create your monitor, right-click the monitor name and select **Start** to run the monitor.

Scenarios: IBM Navigator for i monitors
Use this information to see how you can use some of the different types of monitors to look at specific aspects of your system's performance.

The monitors included in IBM Navigator for i provide a powerful set of tools for researching and managing system performance. For an overview of the types of monitors that are provided by IBM Navigator for i, see "IBM Navigator for i Monitors" on page 90.

See the following scenarios for detailed usage examples and sample configurations:

Scenario: System monitor
See an example system monitor that alerts you if the CPU utilization gets too high and temporarily holds any lower priority jobs until more resources become available.

Situation

As a system administrator, you need to ensure that the system has enough resources to meet the current demands of your users and business requirements. For your system, CPU utilization is an important concern. You would like the system to alert you if the CPU utilization gets too high and to temporarily hold any lower priority jobs until more resources become available.

To accomplish this, you can set up a system monitor that sends you a message if CPU utilization exceeds 80%. Moreover, it can also hold all the jobs in the QBATCH job queue until CPU utilization drops to 60%, at which point the jobs are released, and normal operations resume.

Configuration example

To set up a system monitor, you need to define what metrics you want to track and what you want the monitor to do when the metrics reach specified levels. To define a system monitor that accomplishes this goal, complete the following steps:

1. In IBM Navigator for i, select **Monitors** > **System Monitors**. From the **Actions** menu, select **Create New System Monitor...**
2. On the **General** page, enter a name and description for this monitor. Click **Next**.
3. Add and edit the **CPU Utilization (Average)** metric properties by performing the following steps:
   a. To add the metric, select **CPU Utilization (Average)** from the list of Available Metrics, and click **Add**. CPU Utilization (Average) is now listed under Metrics to monitor.
   b. To edit the metric properties, click the **CPU Utilization (Average)** metric in the **Metrics to monitor** list. This action opens the **Configure Metric** page where you can edit the properties of the metric.
   c. For **Collection interval**, specify how often you would like to collect the data. This action overrides the Collection Services setting. For this example, specify **30 seconds**.
   d. For **Threshold 1**, enter the following values to send an inquiry message if the CPU Utilization is greater than or equal to 80%:
      i) Select **Enable threshold**.
      ii) For the threshold trigger value, specify **>= 80** (greater than or equal to 80 percent busy).
      iii) For **Duration**, specify **1** interval.
      iv) For the **IBM i command**, specify the following:
      ```
      SNDMSG MSG('Warning,CPU...') TOUSR(*SYSOPR) MSGTYPE(*INQ)
      ```
      v) For the threshold reset value, specify **< 60** (less than 60 percent busy). This action resets the monitor when CPU utilization falls below 60%.
   e. For **Threshold 2**, enter the following values to hold all the jobs in the QBATCH job queue when CPU utilization stays above 80% for five collection intervals:
      i) Select **Enable threshold**.
      ii) For the threshold trigger value, specify **>= 80** (greater than or equal to 80 percent busy).

iii) For **Duration**, specify **5** intervals.

iv) For the **IBM i command**, specify the following:

```
HLDJOBQ JOBQ(QBATCH)
```

v) For the threshold reset value, specify **< 60** (less than 60 percent busy). This action resets the monitor when CPU utilization falls below 60%.

vi) For **Duration**, specify **5** intervals.

vii) For the **IBM i command**, specify the following:

```
RLSJOBQ JOBQ(QBATCH)
```

This command releases the QBATCH job queue when CPU utilization stays below 60% for five collection intervals.

4. Click **OK** to save the metric properties.
5. Click **Next** to view the monitor summary page.
6. Click **Finish** to save the monitor.
7. From the list of system monitors, right-click the new monitor and select **Start**.

Results

The new monitor collects the CPU utilization, with new data points added every 30 seconds, according to the specified collection interval. The monitor automatically carries out the specified threshold actions whenever CPU utilization reaches 80%. The monitor will continue to run and perform threshold actions until you stop the monitor.

Note: This monitor tracks only CPU utilization. However, you can include any number of the available metrics in the same monitor, and each metric can have its own threshold values and actions. You can also have several system monitors that run at the same time.

Scenario: Message monitor
This example describes a monitor that displays any inquiry messages in your message queue that occur on your system.

Situation

As a system administrator, you need to be aware of inquiry messages as they occur across your system. You can set up a message monitor to display any inquiry messages in your message queue that occur on your system.

Configuration example

To set up a message monitor, you need to define the types of messages you would like to watch for and what you would like the monitor to do when these messages occur. To set up a message monitor that accomplishes this goal, complete the following steps:

1. In IBM Navigator for i, select **Monitors** > **Message Monitors**. From the **Actions** menu, select **Create New Message Monitor**.
2. On the **General** page, enter a name and description for this monitor. Click **Next**.
3. On the **Message Queue** page, enter the following values:
   a. For **Message Queue to Monitor**, specify **QSYSOPR**.
   b. For **Library**, specify **QSYS**.
   c. Click **Next**.
4. On the **Message Set** page, perform the following steps:
   a. On the **Message Set 1** tab, click **Add**.
   b. On the **Add A Message Set** page, enter the following values:

i) Select **Add a user defined set of messages**.
ii) For **Message Type**, select **Inquiry**.
iii) Click **OK**.

c. Select **Set the message trigger and reset**.
d. For **Trigger at the following message count**, specify **1**.
e. Click **Next**.

5. Click **Next** to view the monitor summary page.
6. Click **Finish** to save the monitor.
7. From the list of message monitors, right-click the new monitor and select **Start**.

Results

The new message monitor displays any inquiry messages sent to QSYSOPR.

Note: This monitor responds to only inquiry messages sent to QSYSOPR. However, you can include two different sets of messages in a single monitor, and you can have several message monitors that run at the same time. Message monitors can also carry out IBM i commands when specified messages are received.

IBM Performance Management for Power Systems (PM for Power Systems) - support for IBM i

The IBM Performance Management for Power Systems (PM for Power Systems) in support of IBM i offering automates the collection, archival, and analysis of system performance data and returns reports to help you manage system resources and capacity.

The PM for Power Systems offering includes the Performance Management Agent (PM Agent). The PM Agent is a function of the operating system that provides automated collection of nonproprietary Collection Services data, reduces the data, and sends the data to IBM. When you send your data to IBM, you eliminate the need to store all the trending data yourself. IBM stores the data for you and provides you with a series of reports and graphs that show your server's growth and performance. You can access your reports electronically using a traditional browser.

This offering, when used with the IBM Systems Workload Estimator, allows you to better understand how your business trends relate to the timing of required hardware upgrades, such as central processing unit (CPU) or disk. The IBM Systems Workload Estimator can size a systems consolidation or evaluate upgrading a system with logical partitions, by having PM Agent send the data for multiple systems or partitions to the IBM Systems Workload Estimator.

Related concepts

Collection Services
Collection Services provides for the collection of system and job level performance data. It is the primary collector of performance data.

Related tasks

Size Next Upgrade
Use the Size Next Upgrade action to send data from your current session to Workload Estimator for use in sizing a future system using current performance statistics.

Related reference

PM for Power Systems web siteFor more information, see the PM for Power Systems website.

PM Agent concepts

Learn about the functions and benefits PM Agent can provide and about important implementation considerations.

PM Agent uses Collection Services to gather the nonproprietary performance and capacity data from your server and then sends the data to IBM. This information can include CPU utilization and disk capacity, response time, throughput, application and user usage. When you send your data to IBM, you eliminate the need to store all the trending data yourself. IBM stores the data for you and provides you with a series

of reports and graphs that show your server's growth and performance. You can access your reports electronically using a traditional browser.

The most important requirement for establishing an accurate trend of the system utilization, workload, and performance measurements is consistency. Ideally, performance data should be collected 24 hours per day. Because of the relationship between PM Agent and Collection Services, you need to be aware of the implications that can occur when you are using PM Agent.

Here are some guidelines to help you define your collections when you are using PM Agent:

- **Collect data continuously with Collection Services.**

  PM Agent satisfies this requirement by collecting data 24 hours a day with Collection Services. PM Agent collects performance data at 15-minute intervals. PM Agent uses the 15-minute interval default, but does not change what the Collection Services interval is set to. A 15-minute interval is the recommended interval.

- **Select the Standard plus protocol profile.**

  Standard plus protocol is the default value for the collection profile. The collection profile indicates what data is collected. The collection does not cycle (unless required to do so for other reasons). This action is done to gather information for PM Agent reports.

- **Avoid making interim changes to collection parameters when PM Agent is active.**

  For example, when you configure Collection Services in IBM Navigator for i, the **Create database files during collection** field is checked as the default value. The change takes effect immediately. The collection does not cycle (unless required to do so for other reasons).

Related reference

Collection Services collection profiles
Descriptions of the Collection Services collection profiles. The collection profiles define what is collected.

Configuring PM Agent

To start using PM Agent, you need to activate it, set up a transmission method, and customize the data collection and storage.

PM Agent automates the collection of performance data through Collection Services. You can specify which library to put the data in as long as the library resides on the base auxiliary storage pool (ASP). The library should not be moved to an independent auxiliary storage pool because an independent auxiliary storage pool can be varied off, which stops the PM Agent collection process. PM Agent creates the library during activation if the library does not already exist.

To begin using PM Agent, you need to perform the following tasks:

Activating PM Agent

PM Agent is a part of the operating system and you must activate it to use its collecting capabilities.

You must start PM Agent to take advantage of its data collecting capabilities. You can start PM Agent by using the following method:

Issue the Configure PM Agent (CFGPMAGT) command

Run the Configure PM Agent (CFGPMAGT) command from the command line.

You can proceed to the next step in the setup process, which is to set up the Service Agent transmission used to send data to IBM.

Related concepts

Setting up PM Agent transmission of data
You gather and send the performance data using the inventory function of the Electronic Service Agent.

Related tasks

Deactivating PM Agent

Learn how you can stop PM Agent.

Setting up PM Agent transmission of data
You gather and send the performance data using the inventory function of the Electronic Service Agent.

Once you have implemented the transmission method, you are ready to do the other tasks to manage PM Agent.

Related reference
Managing PM Agent
Now that you have set up your network, you can perform a variety of tasks with PM Agent.

Sending PM Agent data with Electronic Service Agent

PM Agent must be active in order to collect performance data. PM Agent uses Collection Services to gather the nonproprietary performance and capacity data from your server. After you have collected this data, use Electronic Service Agent to send the data to IBM.

Related concepts
Configuring Universal ConnectionLearn how to use the Universal Connection wizard and other related tasks that you need when you create a Universal Connection to IBM services.
Related tasks
Setting up a connection to IBMIf you are using a Hardware Management Console (HMC), you need to complete these steps on your HMC to set up a connection to IBM.
Activating PM Agent
PM Agent is a part of the operating system and you must activate it to use its collecting capabilities.

Related reference
Managing PM Agent
Now that you have set up your network, you can perform a variety of tasks with PM Agent.

Configuring PM Agent network for a single server
A single server sends its data directly to IBM.

Here are the steps that you need to follow to configure PM Agent for a single server:

1. Type **CFGPMAGT** from the command line.
2. Specify **1= Send data with Service Agent** for the **Select connectivity option for sending performance data to IBM** field.
3. Specify **0=No** for the **Receive performance data** field.
4. Type your company's contact information on the Work with Contact Information display making sure that you fill out all the mandatory fields, otherwise the configuration will be incomplete.

Related reference
Managing PM Agent
Now that you have set up your network, you can perform a variety of tasks with PM Agent.

Configuring PM Agent network for a host server
A host server receives performance data from other servers and then forwards the data to IBM .

Here are the steps that you need to follow to configure PM Agent for a host server:

1. Type **CFGPMAGT** from the command line on your host server. From the Configure PM Agent display, do the following:
   - Specify **1= Send data with Service Agent** for the **Select connectivity option for sending performance data to IBM** field.
   - Specify **1=Yes** for the **Receive performance data** field.
2. Type **GO PMAGT** from the command line on your host server. From the PM Agent Menu display, do the following:
   - Type a **5** at the command line to Work with Remote IBM i systems then press **Enter**.

- Press **F6** (Create) to identify which servers will send their data to your host server.
- Complete the fields and press Enter.

PM Agent automatically schedules the transmission of data from the primary server to IBM the day after data is received from a remote server. If the automatic scheduling does not fit your work management scheme, you can manually schedule the transmission of the data from the primary server.

Here is a tip that you should keep in mind when scheduling the transmission of your data. Throughout the week, evenly schedule the transmission of data to the host server. This action minimizes the performance impact on the host server. For example, in a network of twelve servers, you might have three groups of four systems. You can schedule each group to send their data on Monday, Wednesday, and Friday. This evenly distributes the amount of data that is sent to the host server.

Once you have configured your servers, you are ready to do the other tasks to manage PM Agent.

Related reference

Managing PM Agent
Now that you have set up your network, you can perform a variety of tasks with PM Agent.

Configuring PM Agent network for a remote server
A remote server sends its performance data to a host server.

Here are the steps that you need to follow to configure PM Agent for a remote server:

1. Type **CFGPMAGT** from the command line.
2. Specify **2=This is a remote IBM i using SNA** for the **Select connectivity option for sending performance data to IBM** field.
3. Specify **0=No** for the **Receive performance data** field.

Once you have configured your servers, you are ready to do the other tasks to manage PM Agent.

Related reference

Managing PM Agent
Now that you have set up your network, you can perform a variety of tasks with PM Agent.

Working with remote servers
In some sites, there will be a network composed of a host server and one or more remote servers. In this case, the host server will send PM data to IBM for processing.

When you use a host server network, you have the other servers in the network send their performance data to this host server for transmission to IBM. To set up your network to use a host server, you must identify the other remote servers and set the schedule for their data transmission. The Work with Remote IBM i systems display (type **GO PMAGT**, then type a **5**) enables you to define these other servers.

Follow these steps to define your remote servers:

1. Type **GO PMAGT** from the command line.
2. Type a **5 (Work with Remote IBM i systems)** from the PM Agent Menu and press Enter. You do not see a remote server displayed initially. You must create a new remote location.
3. Create a new remote location by pressing **F6 (Create)**.
4. Record the values for the following information. Use the **Display Network Attributes (DSPNETA)** command to display these values from the remote system.
   - Local network ID
   - Default local location

   The Work with Remote IBM i systems display shows a list of remote servers. This list includes the status for the servers (active or inactive) and the descriptions for each server.
5. Create or change the description for a remote site server by using the PM Agent Work with Remote IBM i systems display. The remote location name must be unique between remote servers.

The PM Agent software assumes that you defined the Advanced Peer-to-Peer Networking (APPN) link between the server that receives data (the host server) and the server that sends data (the remote server). If your system has the system value QCRTAUT (Create default public authority) set to *EXCLUDE or *USE, you should see Create a device description for a remote server for information on how to define your controller descriptions. If your network does not meet these assumptions, see Non-APPN network considerations for information about creating device pairs to support the connection to each remote server.

Once you have defined your remote servers, you are ready to customize PM Agent to use a specific line connection.

Related tasks

Scheduling jobs with PM Agent
Learn how to schedule jobs with PM Agent.

Creating a device description for PM Agent
You can create a device description for PM Agent.

Working with remote servers in an APPC network
The host server receives PM Agent data from other servers and then sends the data to IBM. The remote server sends PM Agent data to the host server.

Customizing PM Agent
Now that you have set up your network, you may need to customize PM Agent to fit your needs.

Working with remote servers in an APPC network
The host server receives PM Agent data from other servers and then sends the data to IBM. The remote server sends PM Agent data to the host server.

The following information assumes that the controllers that are referred to have previously been defined.

You need to create device pairs to support the connection to each remote server only if PM Agent gathers data.

1. Use the Create Device Description (APPC) (CRTDEVAPPC) command. On the remote server, type CRTDEVAPPC. Press F4 to prompt for the parameters, and define the values with the following information:

   Table 3. Remote system

Keyword(Value)	Description
DEVD(Q1PLOC)	Specifies the name of the device description.
RMTLOCNAME(Q1PLOC)	Specifies the name of the remote location.
ONLINE(*YES)	Specifies whether this device is varied online when the system is started or restarted.
LCLLOCNAME(Q1PRMxxx)	Specifies the local location name. Q1PRMxxx matches the RMTLOCNAME of the primary server, where xxx is unique for each remote location.
CTL(yyyyyy)	Specifies the name of the attached controller, where yyyyyy is a controller that attaches to the primary server.
MODE(Q1PMOD)	Specifies the mode name.
APPN(*NO)	Specifies if the device is APPN-capable.

2. Specify the following information on the host server. At the command line, type CRTDEVAPPC. Press F4 to prompt for the parameters, and define the values with the following information:

Table 4. Primary server

Keyword(Value)	Description
DEVD(Q1PRMxxx)	Specifies the name of the device description. The name that is used here matches the device description name for the remote system.
RMTLOCNAME(Q1PRMxxx)	Specifies the name of the remote location. The name that is used here matches the LCLLOCNAME value of the remote server, where xxx is unique for each remote location.
ONLINE(*YES)	Specifies whether this device is varied online when the system is started or restarted.
LCLLOCNAME(Q1PLOC)	Specifies the local location name. This value matches the RMTLOCNAME of the remote server.
CTL(aaaaaa)	Specifies the name of the attached controller, where aaaaaa is a controller that attaches to the remote server.
MODE(Q1PMOD)	Specifies the mode name.
APPN(*NO)	Specifies if device is APPN-capable.

3. Vary on the devices (Vary Configuration (VRYCFG) command) after you define the APPC devices. On the remote server, type VRYCFG. Press F4 to prompt for the parameters.

Table 5. Vary on remote system

Keyword(Value)	Description
CFGOBJ(Q1PLOC)	Specifies the configuration object.
CFGTYPE(*DEV)	Specifies the type of configuration object.
STATUS(*ON)	Specifies the status

4. Type option 5 on the PM Agent Menu to add Q1PRMxxx as a remote server. See Working with remote servers for instructions on how to add a remote server.

Now that you have finished configuring PM Agent, see Managing PM Agent for other tasks that you can perform with PM Agent.

Related tasks

Working with remote servers
In some sites, there will be a network composed of a host server and one or more remote servers. In this case, the host server will send PM data to IBM for processing.

Related reference

Managing PM Agent
Now that you have set up your network, you can perform a variety of tasks with PM Agent.

Create Device Description (APPC) (CRTDEVAPPC) commandSee the Create Device Description (APPC) (CRTDEVAPPC) command for information about creating a device description for an advanced program-to-program communications (APPC) device.

Vary Configuration (VRYCFG) commandSee the Vary Configuration (VRYCFG) command for information about varying on or off one or more configuration objects, with the capability of also varying on the downline attached configuration objects.

Creating a device description for PM Agent
You can create a device description for PM Agent.

The following steps are necessary on each remote server that has the Create default public authority (QCRTAUT) system value set to *EXCLUDE or *USE. If QUSER does not have *CHANGE authority to device description Q1PLOC, remote transmissions will fail. These steps ensure that the device will not be created or deleted automatically.

Note: This task is necessary only if PM Agent gathers data.

If you allow the device to be created automatically, the device description is created with PUBLIC *EXCLUDE or *USE authority, depending on the value set for QCRTAUT. Whether a device can be created or deleted automatically is controlled by the controller.

For systems that are not configured to use APPN, see Work with remote servers in a non-APPN environment for information on how to create the device description.

The following information assumes that the controller that will be used to communicate with the host server was defined previously on the remote server.

On the *remote server*, re-create device description Q1PLOC:

```
VRYCFG  CFGOBJ(Q1PLOC)
        CFGTYPE(*DEV)
        STATUS(*OFF)
DLTDEVD   DEVD(Q1PLOC)

CRTDEVAPPC DEVD(Q1PLOC)
           RMTLOCNAME(Q1PLOC)
           ONLINE(*NO)
           LCLLOCNAME(name of remote system)
           RMTNETID(remote netid of primary (or central) system)
           CTL(name of controller that the device will be attached to)
           AUT(*EXCLUDE)

CRTOBJAUT OBJ(Q1PLOC)
          OBJTYPE(*DEVD)
          USER(QUSER)
          AUT(*CHANGE)

 VRYCFG    CFGOBJ(Q1PLOC)
           CFGTYPE(*DEV)
           STATUS(*ON)
```

Related tasks

Working with remote servers in an APPC network
The host server receives PM Agent data from other servers and then sends the data to IBM. The remote server sends PM Agent data to the host server.

Related reference

Create Controller Description (APPC) (CRTCTLAPPC) commandSee the Create Controller Description (APPC) (CRTCTLAPPC) command for information about creating a controller description for an advanced program-to-program communications (APPC) controller.

Change Controller Description (APPC) (CHGCTLAPPC) commandSee the Change Controller Description (APPC) (CHGCTLAPPC) command for information about changing a controller description for an advanced program-to-program communications (APPC) controller.

Display Controller Description (DSPCTLD) commandSee the Display Controller Description (DSPCTLD) command for information about displaying a controller description.

Managing PM Agent
Now that you have set up your network, you can perform a variety of tasks with PM Agent.

Customizing PM Agent
Now that you have set up your network, you may need to customize PM Agent to fit your needs.

The Work with PM Agent Customization display provides you with the ability to:

Establish global parameters for the operation of PM Agent software

The global parameters allow you to customize the following items. See the online help for a description of these fields:

- Priority limits
- Trend and shift schedules

Define your PM Agent Global Parameters

To customize the global parameters, do the following steps:

1. Type **GO PMAGT** from the command line.
2. Type a 3 from the PM Agent menu to display the Work with PM Agent Customization display and press Enter.

See Managing PM Agent for other tasks that you can perform with PM Agent.

Related reference

Managing PM Agent
Now that you have set up your network, you can perform a variety of tasks with PM Agent.

Managing PM Agent

Now that you have set up your network, you can perform a variety of tasks with PM Agent.

After you have set up your network to use PM Agent, you can perform the following tasks:

Related reference

End PM Agent (Q1PENDPM) APIThe End PM Agent (Q1PENDPM) API is used to end the Performance Management Agent (PM Agent) jobs.

Start PM Performance Data Transmission

You can use the PM Agent menu to start the Service Agent transmission of the performance data.

To start PM Performance Data Transmission, do these steps:

1. Type **GO PMAGT** from the command line.
2. Type a **9** from the command line and press Enter. Check the joblog for any messages returned.

Deactivating PM Agent

Learn how you can stop PM Agent.

To stop PM Agent from running, use the following method:

Run the `End PM Agent (Q1PENDPM)` API from the command line to deactivate PM Agent.

Related tasks

Activating PM Agent
PM Agent is a part of the operating system and you must activate it to use its collecting capabilities.

Changing PM Agent contact information

Learn how to change your contact information from the original settings.

During the configuration of the PM Agent software, you identified the contact person and provided mailing information for your organization. If at a later time, you need to update the information, use the Work with Contact Information option to change that information. To change the contact information, do the following steps:

1. Type **GO PMAGT** from the command line.
2. Type a 1 from the PM Agent Menu and press Enter. The Work with Contact Information display appears.
3. Change the contact information, as appropriate, and press Enter.

Scheduling jobs with PM Agent

Learn how to schedule jobs with PM Agent.

Integral to the PM Agent software is a scheduler that automatically starts the jobs that are necessary to support the PM Agent performance data collection and analysis.

Part of the PM Agent software activation process includes starting a job that is called Q1PSCH. This job, in turn, starts other jobs as shown in the following table:

To access the PM Agent scheduled jobs, do the following:

1. Type **GO PMAGT** from the command line.

2. Type a **2** from the PM Agent Menu and press Enter. The Work with Automatically Scheduled Jobs display appears.
3. You can change the status for each job from active to inactive. Type a **2** (Change) next to the job that you want to change and press Enter. You are shown the Change Automatically Scheduled Jobs display.

The following table shows you a list of the possible PM Agent jobs.

PM Agent scheduled jobs		
Job	**Schedule**	**Function**
Q1PTEST	At activation	Verifies that PM Agent is activated and then goes to inactive status.
Q1PPMSUB	Hourly	Ensures that Collection Services is collecting data.
Q1PDR	Daily	Performs data reduction and purges performance data.
Q1PPG	Monthly	Purges reduced performance data.
Q1PCM4	As needed	Accesses the PM Agent from remote servers. This job is started only if you have added remote systems by using option 5 from the PM Agent Menu.
Q1PPMCHK	Every 4 hours	Verifies that data collection is active.

Omitting items from IBM Performance Management for Power Systems (PM for Power Systems) analysis
Learn how to omit jobs, users, and communications lines when performing an analysis with IBM Performance Management for Power Systems (PM for Power Systems))

The PM for Power Systems software application summary includes an analysis of items for batch jobs, users, and communication lines. However, some jobs, users, or communication lines are not appropriate for such an analysis. For example, you may want to exclude jobs with longer than normal run times, such as autostart jobs, in the run-time category.

You can omit groups of batch jobs and users from the analysis by using the generic omit function. For example, to omit all jobs starting with MYAPP specify: MYAPP*

To work with omissions, do the following steps:

1. Type **GO PMAGT** from the command line.
2. Type a **4** from the PM Agent Menu and press Enter. The Work with Omissions display appears.
3. Type the appropriate option number depending on which item you want to omit.
   - Type a **1** to work with jobs.
   - Type a **2** to work with users.
   - Type a **3** to work with communications lines.
4. Type a **1** in the appropriate field to omit either a user or a job from a particular category. In the case of a communications line, type the name of the line and then type a **1** in the appropriate field.

Stopping PM Agent momentarily
Learn how you can stop PM Agent momentarily.

If you need to stop PM Agent from verifying that Collection Services is collecting data, you can use the scheduler job to change the date to a future date for the Q1PPMSUB job.

1. Type **GO PMAGT** from the command line.
2. Type a 2 (Work with automatically scheduled jobs).
3. Type a 2 (Change) next to the Q1PPMSUB job.
4. Change the date or time to a future date and time.
5. Press Enter. This change will momentarily stop PM Agent from verifying that Collection Services is collecting data. You must end what is currently being collected.

Note: PM Agent will not start, cycle, or change Collection Services until the date and time to which you set the Q1PPMSUB job has been reached.

Related tasks

Scheduling jobs with PM Agent
Learn how to schedule jobs with PM Agent.

Verify Service Agent Connection
You can use the PM Agent menu to verify the status of the Service Agent connection from your server to IBM.

To verify the Service Agent connection, do the following steps:

1. Type **GO PMAGT** from the command line.
2. Type an **8** from the command line and press Enter. Check the joblog for any messages returned.

Viewing PM Agent status
Learn how to use the PM Agent menu to display PM Agent status.

You can use the PM Agent Menu on your system to display the status of PM Agent. Use the PM Agent Menu to view the Collection Services status, PM Agent scheduler status, the performance data release, the last transmission attempt, performance data members, and the performance data size.

To view the detailed status for PM Agent from the PM Agent menu, do the following steps:

1. Type **GO PMAGT** from the command line.
2. Type a 6 from the command line and press Enter. See the online help for descriptions of each field.

Viewing IBM Performance Management for Power Systems (PM for Power Systems) reports
See examples of the IBM Performance Management for Power Systems (PM for Power Systems) reports and explanations of how to interpret those reports.

The output of the IBM Performance Management for Power Systems (PM for Power Systems) offering is a set of management reports and graphs. The purpose of the reports and graphs is to give management a clear understanding of the current performance of their servers and an accurate growth trend. To view the reports and learn about some of their benefits and uses, visit the IBM Performance Management for Power Systems (PM for Power Systems) web site.

Related reference

PM for Power Systems web siteFor more information, see the PM for Power Systems website.

IBM Systems Workload Estimator

The IBM Systems Workload Estimator is a Web-based sizing tool for System i, System p, and System x. You can use this tool to size a new system, to size an upgrade to an existing system, or to size a consolidation of several systems.

The Workload Estimator allows measurement input to best reflect your current workload; one method is by using data from IBM Performance Management for Power Systems (PM for Power Systems). The Workload Estimator also provides a variety of built-in workloads to reflect your emerging application requirements. Virtualization can be used to yield a more robust solution. The Workload Estimator will provide current and growth recommendations for processor, memory, and disk that satisfy the overall client performance requirements.

Related reference

IBM Systems Workload EstimatorSee the IBM Systems Workload Estimator website to run the online version of the Workload Estimator.

IBM Performance Tools for i

The IBM Performance Tools for i licensed program product includes many supplemental features that supplement or extend the capabilities of the basic performance tools that are available in the operating system.

Related concepts

Collection Services
Collection Services provides for the collection of system and job level performance data. It is the primary collector of performance data.

Manager and Agent feature comparison
You can use the Manager and Agent features to efficiently divide required functions of Performance Tools over a distributed environment. This topic contains a description of these two features, the functions they each contain, and information about how to use them most effectively.

Related reference

Performance Tools PDFThis IBM Redbooks publication explains how to use performance tools to collect data about the performance of a system, job, or program. It also explains how to analyze and print the data to help identify and correct any problems.

CL commands for performanceThe operating system includes several CL commands to help you manage and maintain system performance.

Performance Tools concepts

Describes a variety of tools to help you collect and analyze performance information. Find detailed information about exactly which tools perform which functions and how they work.

Related concepts

Collection Services
Collection Services provides for the collection of system and job level performance data. It is the primary collector of performance data.

Functions provided in IBM Performance Tools for i

IBM Performance Tools for i includes a variety of applications for collecting, analyzing, and reporting performance data. Knowing which functions are available, and which are best suited for a given task can be complex. This topic describes the functions included in this licensed program product.

IBM Performance Tools for i includes the following functions:

Tool	Description
Display Performance Data	Display Performance Data allows you to interactively display sample performance data. To interactively display sample performance data, you can use the Display Performance Data (DSPPFRDTA) command or select the Display performance data option on the IBM Performance Tools menu. Display Performance Data is discussed in detail in the Performance Tools PDF.
Reports	The reports organize Collection Services performance data and trace data in a logical and useful format. The reports are discussed in detail in the Performance Tools PDF.
Graphics function	The Performance Tools graphics function allows you to work with performance data in a graphical format. You can display the graphs interactively, or you can print, plot, or save the data to a graphics data format (GDF) file for use by other utilities. This tool is discussed in detail in the Performance Tools PDF.

Tool	Description
IBM i Job Watcher	The Job Watcher functions and content package in the IBM Navigator for i Performance interface is included in IBM Performance Tools for i (5770-PT1) Option 3 - Job Watcher.
IBM i Disk Watcher	The Disk Watcher functions and content package in the IBM Navigator for i Performance interface is included in IBM Performance Tools for i (5770-PT1) Option 1 - Manager Feature.
Performance Explorer	The Performance Explorer content package in the IBM Navigator for i Performance interface is included in IBM Performance Tools for i (5770-PT1) Option 1 - Manager Feature.
Database	The Database content package in the IBM Navigator for i Performance interface is included in IBM Performance Tools for i (5770-PT1) Option 1 - Manager Feature.
Batch Model	The Batch Model functions and content package in the IBM Navigator for i Performance interface is included in IBM Performance Tools for i (5770-PT1) Option 1 - Manager Feature.

Related concepts

Performance Tools reports
Performance Tools reports provide information on data that has been collected over time. Use these reports to get additional information about the performance and use of system resources.

IBM Navigator for i Performance interface
Use the IBM Navigator for i Performance interface to view, collect, and manage performance data by bringing together various performance information and tools into one centralized place.

Related reference

Work with System Activity (WRKSYSACT) commandSee the Work with System Activity (WRKSYSACT) command for information about how to interactively work with the jobs and tasks currently running in the system.

Performance Tools PDFThis IBM Redbooks publication explains how to use performance tools to collect data about the performance of a system, job, or program. It also explains how to analyze and print the data to help identify and correct any problems.

Manager and Agent feature comparison
You can use the Manager and Agent features to efficiently divide required functions of Performance Tools over a distributed environment. This topic contains a description of these two features, the functions they each contain, and information about how to use them most effectively.

Performance Tools is available with two separately installable features. This topic explains the differences between the two features to help you decide which feature is more appropriate for your applications.

IBM Performance Tools for i (5770-PT1) Option 1 - Manager Feature

The Performance Tools Manager feature is a full-function package, intended to be used on the central site system in a distributed environment or on a single system. If you require analysis of trace data, viewing data graphically, viewing system activity in real time, or managing and tracking system growth, the Manager feature of the Performance Tools licensed program is more useful.

The Manager feature also contains the following functions in the IBM Navigator for i Performance interface:

- IBM i Disk Watcher functions and content package
- Performance Explorer content package
- Database content package
- Batch Model functions and content package

IBM Performance Tools for i (5770-PT1) Option 2 - Agent feature

The Performance Tools Agent feature, with a subset of the Manager function, is a lower-priced package with the more basic functions. In a distributed environment, the Agent feature works well for managed systems in the network because the data can be sent to the Manager if detailed analysis is required. It is also an effective tool for sites that need a reasonable level of self-sufficiency but have no expert skills available.

The Agent feature of Performance Tools provides functions to simplify the collection, management, online display, data reduction, and analysis of performance data. The major Performance Tools functions **not** contained in the Agent feature are performance and trace reports, performance utilities (job traces and the select file utilities), system activity monitoring, and performance graphics.

Related concepts

IBM Navigator for i Performance interface
Use the IBM Navigator for i Performance interface to view, collect, and manage performance data by bringing together various performance information and tools into one centralized place.

Reporting CPU utilization
Find out how the total CPU that is consumed across virtual processors is reported.

Prior to V5R3, processor utilization was calculated as a percentage of the available CPU time. Collection Services reported, in the performance database files, the time used on each processor along with elapsed interval time. Users of this data, such as the Performance Tools reports and displays, needed to add up the time used on each processor to get the total system CPU that was consumed. The available CPU time was calculated as the number of processors in the partition multiplied by the duration of the data collection interval. Finally, the CPU time was divided by the calculated available time to get the utilization percentages.

The problem with the previous methodology is that all users of the data assumed whole virtual processors and depended on no changes to the configured capacities. Logical partitions with partial processor capacities would require all tools to scale the calculated capacity based on processor units allocated. The capability to do dynamic configuration would cause incorrect utilizations when the configured processor units changed. Temporary solutions for minimizing the impacts of these problems included scaling the utilization of the system processors to what would be reported for a whole number of processors and cycling Collection Services when the configuration changed. Because the individual job CPU time was not scaled, the additional time was accounted for by reporting it as being consumed by HVLPTASK. The HVLPTASK task did not actually use CPU, but CPU time was shown to be consumed by HVLPTASK for accounting purposes. The CPU time charged to HVLPTASK scaled the amount of work that was done by real jobs, which resulted in the system CPU percent utilization going from 0 to 100 in direct proportion to the amount of customer work that was performed.

Collection Services now reports the total processor time that is consumed by the partition along with the amount of processor time that was available to be consumed within the partition, regardless of the number of virtual processors that are configured, the partition units that are configured, or how they changed during the interval. To calculate utilization, users of this data divide the reported CPU consumed by the available capacity. This method of calculating CPU utilization eliminates the increasingly error-prone task of computing available CPU time. CPU utilization that is calculated with these new metrics is accurate regardless of how many processing units (whole or fractional) exist, when the processing units changed, or how often the units changed.

Several reasons account for this change in calculating CPU utilization. One reason is that with scaling utilization for jobs or groups of jobs appeared to be much smaller than would be anticipated. This concept is demonstrated in the example that follows. Another reason is that a configuration change could make CPU reporting not valid. Traditionally, the number of CPUs was based on the value that was configured at the beginning of a collection and an IPL was needed to change it. When dynamic configuration was introduced, Collection Services cycled the collection to handle the configuration changes, which assumed that changes would be infrequent. However, the more frequent the change, the more cycling occurs. If the changes are too frequent, collecting performance data is not possible. Lastly, even if the proper configuration data were reported and used for every interval, you would not know what happened between the time the interval started and until it completed. Utilization would still be calculated incorrectly in any interval where there was one or more configuration changes.

Example

Partition A has a capacity of 0.3 processor units and is defined to use one virtual processor. The collection interval time is 300 seconds. The system is using 45 seconds of CPU (15 seconds by interactive jobs and 30 seconds by batch jobs). In this example, the available CPU time is 90 seconds (.3 of 300 seconds). The total CPU utilization is 50%.

Prior to V5R3, when the numbers were scaled, system CPU usage is reported as 150 seconds. 150 seconds divided by 300 seconds of interval time results in 50% utilization. The interactive utilization is 15 seconds divided by 300 seconds, which is 5%. The batch utilization is 30 seconds divided by 300 seconds, which is 10%. The HVLPTASK is getting charged with 35% utilization (150 seconds minus 45 seconds), or 105 seconds divided by 300 seconds. These percentages give us a total of 50%.

Beginning in V5R3, the 45 seconds of usage is no longer scaled but is reported as is. The calculated CPU time that is derived from the reported consumed CPU time divided by the reported available capacity is 50% (45 seconds divided by 90 seconds). The interactive utilization percentage is 17% (15 seconds divided by 90 seconds). The batch utilization percentage is 33% (30 seconds divided by 90 seconds).

IBM i Release	Total CPU	Interactive	Batch	HVLPTASK
V5R2 or earlier	50%	5%	10%	35%
V5R3 or later	50%	17%	33%	N/A

Reporting configured capacity
Find out where the information for configured capacity is recorded.

The partition capacity values are determined initially when the partition is started and depends on the capacity resources available at the time. These initial values can be altered through configuration changes while the partition is active.

Logical partitions (LPARs) enable some partitions to exceed their configured capacity in certain situations. During these times, the processor utilization metrics of these partitions can be greater than 100% of the configured capacity.

The usage and capacity information is recorded in the QAPMSYSTEM database file. The virtual processor information is recorded in the QAPMSYSCPU database file. The following values summarize this information:

Virtual processors
The number of processors that are assigned to a logical partition that is sharing processor capacity of the shared processor pool. This value determines the number of concurrent processors that could be active in the logical partition. This value is included in the QAPMSYSCPU performance database file in the field (or column) named SCTACT.

Shared processor pool capacity available
Total processor capacity in the shared processor pool available for use by shared processor logical partitions. This value is included in the QAPMSYSTEM performance database file in a column named SYSPLA. If partitions configured as uncapped compete for available shared pool capacity in excess of the guaranteed amount, the distribution of processor capacity is determined by the uncapped weight assigned to the logical partition.

Shared processor capacity used
Total amount of shared processor capacity used by all active shared processor logical partitions. Total amount of CPU used within the shared pool by all partitions that share the pool. This value is included in the QAPMSYSTEM performance database file in a column named SYSPLU.

Partition guaranteed capacity
Processor capacity configured to a shared processor logical partition from the shared processor pool. This value is included in the QAPMSYSTEM performance database file in a column named SYSCTA. The 5250 OLTP capacity configured is recorded in column named SYIFTA.

Partition processor utilization
Total CPU time used by a logical partition. In a shared processor logical partition with uncapped capacity, this value may exceed the guaranteed capacity if there is unused capacity in the shared processor pool. This value is included in the QAPMSYSTEM performance database file in a column named SYSPTU. The 5250 OLTP capacity used is recorded in column named SYIFUS. The maximum processor capacity in a partition is determined by the number of virtual processors configured.

Partition available capacity
The amount of processor capacity that could have been used by the logical partition. This value is included in the QAPMSYSTEM performance database file in a column named SYSUTA. This is the processor capacity utilized (SYSPTU) plus the unused capacity in the shared processor pool (SYSPLA), subject to the following limits:

- The minimum is the configured (guaranteed) capacity.
- The maximum is the capacity based on the number of virtual processors assigned to the partition and pool.

Related reference
Collection Services data files: QAPMSYSTEMThis Collection Services database file reports system-wide performance data.
Collection Services data files: QAPMSYSCPUThis Collection Services database file reports utilization for virtual processor units.

5250 online transaction processing (OLTP)
This topic describes 5250 online transaction processing and what jobs or threads are associated with this work.

Online transaction processing (OLTP) refers to a type of interactive application in which requests submitted by users are processed as soon as they are received. The following are examples of OLTP processing:

- The system interactions through a 5250 session, a pass-through job, or a Telnet job.
- A workstation-based request from a Domino mail or calendar, or a browser-based application.

IBM i Access jobs use both interactive and batch, depending on the function. Before V5R3, these jobs were included in the CA4 category and listed as interactive. The distributed data management (DDM) server jobs were also listed as interactive.

After V5R3, the Performance Tools licensed program is updated to better distribute the workloads, depending on which processor capacity feature that the CPU cycles were charged against. The interactive CPU reporting refers to those jobs whose CPU is allocated to the 5250 OLTP processor capacity. The IBM i Access jobs are listed in the appropriate sections of the Performance Tools reports. In addition, the DDM jobs moved from the Interactive workload section of the reports to the Non-interactive workload section.

Installing and configuring Performance Tools
See this topic for installation and setup instructions.

To install Performance Tools, you need a user profile with save system (*SAVSYS) authority. You can use the system operator profile to obtain this authority.

Performance Tools must run in a library named QPFR. If you have a library by this name on your system, use the Rename Object (RNMOBJ) command to rename it before you install Performance Tools. This step will ensure the proper operation of Performance Tools.

Use the following command to place the Performance Tools in library QPFR:

```
RSTLICPGM LICPGM(xxxxPT1) DEV(NAME) OPTION(*BASE)
```

You must then perform one of the following:

- If you have purchased the Manager feature, use the following command:

```
RSTLICPGM LICPGM(xxxxPT1) DEV(tape-device-name) OPTION(1)
```

- If you have purchased the Agent feature, use the following command:

```
RSTLICPGM LICPGM(xxxxPT1) DEV(NAME) OPTION(2)
```

- In addition to installing either the Manager or the Agent feature, if you have purchased IBM i5/OS Job Watcher, use the following command:

```
RSTLICPGM LICPGM(xxxxPT1) DEV(tape-device-name) OPTION(3)
```

If you have several CD-ROMs to install, the following situation may occur. After installing the first one, you may receive a message saying that the licensed program is restored but no language objects were restored. If this occurs, insert the next CD-ROM and enter the following:

```
RSTLICPGM LICPGM(xxxxPT1) DEV(NAME) RSTOBJ(*LNG) OPTION(*BASE)
```

Another method for installing the Performance Tools program is to type GO LICPGM and use the menu options.

Performance Tools is a processor-based program. The usage type is concurrent, and the program is installed with a usage limit *NOMAX.

This program is discussed in detail in the Performance Tools book.

Related reference

Performance Tools PDFThis IBM Redbooks publication explains how to use performance tools to collect data about the performance of a system, job, or program. It also explains how to analyze and print the data to help identify and correct any problems.

Performance Tools reports

Performance Tools reports provide information on data that has been collected over time. Use these reports to get additional information about the performance and use of system resources.

The Performance Tools reports provide an easy way for you to look at your collected data and isolate performance problems. After you have collected performance data over time, you can print the reports to see how and where system resources are being used. The reports can direct you to specific application programs, users, or inefficient workloads that are causing slower overall response times.

Collection Services provides data for most of the Performance Tools reports with the exception of the Transaction, Lock, and Trace reports. You must use the Start Performance Trace (STRPFRTRC) and End Performance Trace (ENDPFRTRC) commands to collect the trace information for those three reports.

Overview of Performance Tools reports
The following list describes each report and gives a brief overview as to why you would use a particular report.

Table 6. Overview of Performance Tools reports

Report	Description	What is shown	How you use the information
System Report	Uses Collection Services data to provide an overview of how the system is operating. The report contains summary information on the workload, resource use, storage pool utilization, disk utilization, and communications. Run and print this report often to give you a general idea of your system use.	System workload. The report includes the database capabilities data.	Workload projection
Component Report	Uses Collection Services data to provide information about the same components of system performance as a System Report, but at a greater level of detail. This report helps you find which jobs are consuming high amounts of system resources, such as CPU, disk, and so on.	Resource use, communications, system and user jobs. The report includes the database capabilities data and the Interactive Feature utilization.	Hardware growth and configuration processing trends
Transaction Reports	Uses trace data to provide detailed information about the transactions that occurred during the performance data collection.	Workload and utilization of CPU, disk, main storage, transaction workload, object contention	Workload projection, pool configuration, application design, file contention, and program use

Table 6. Overview of Performance Tools reports (continued)			
Report	**Description**	**What is shown**	**How you use the information**
Lock Report	Uses trace data to provide information about lock and seize conflicts during system operation. With this information you can determine if jobs are being delayed during processing because of unsatisfied lock requests or internal machine seize conflicts. These conditions are also called waits. If they are occurring, you can determine which objects the jobs are waiting for and the length of the wait.	File, record, or object contention by time; the holding job or object name; the requesting job or object name	Problem analysis. Reduction or elimination of object contention.
Batch Job Trace Report	Uses trace data to show the progression of different job types (for example, batch jobs) traced through time. Resources utilized, exceptions, and state transitions are reported.	Job class time-slice end and trace data	Problem analysis and batch job progress
Job Interval Report	Uses Collection Services data to show information on all or selected intervals and jobs, including detail and summary information for interactive jobs and for noninteractive jobs. Because the report can be long, you may want to limit the output by selecting the intervals and jobs you want to include.	Jobs by interval	Job data

Table 6. Overview of Performance Tools reports (continued)			
Report	**Description**	**What is shown**	**How you use the information**
Pool Interval Report	Uses Collection Services data to provide a section on subsystem activity and a section on pool activity. Data is shown for each sample interval. Because the report can be long, you may want to limit the output by selecting the intervals and jobs you want to include.	Pools by interval	Pool data
Resource Interval Report	Uses Collection Services data to provide resource information on all or selected intervals. Because the report can be long, you may want to limit the output by selecting the intervals you want to include.	Resources by interval	System resource use

Performance explorer and Collection Services are separate collecting agents. Each one produces its own set of database files that contain grouped sets of collected data. You can run both collections at the same time.

Sample System Report - Workload
The Workload section of the system report displays the interactive and non-interactive workload of the system.

The first part of the Workload section of the System Report displays the Interactive Workload of the system. The second part of the Workload section displays the Non-Interactive Workload of the system.

```
                                                                System Report                                                6/26/04
16:06
                                                                     Workload
Page 0
    Member  . . . : PNT6PERF   Model/Serial  . : 825/10-5M0FM        Main storage . . : 8192.0 MB  Started  . . . . : 04/07/04
19:11
      Library . . : CARR098R01 System name . . : CARREGT             Version/Release  :   5/ 4.0   Stopped  . . . . : 04/07/04
20:15
    Partition ID  : 000        Feature Code  . : 7415-2472-7415      Int Threshold  . : 100.00
%
    Virtual Processors:  32    Processor Units :
32.0
    QPFRADJ . . . : 0          QDYNPTYSCD  . . : 1                   QDYNPTYADJ . . . :
1
    Interactive
Workload
        Job                Number          Average      Logical DB      -------- Printer ----------      Communications
MRT
        Type            Transactions       Response      I/O Count        Lines          Pages             I/O Count
Max ime
    -------------     ----------------     --------  -----------------  -------------    ----------      --------------
-------
    Interactive                  3,242         .65            16,734          12,910           339
0              0
    DDM Server                       0         .00           864,667             443            23
1,596,096              0
    PassThru                     6,645         .68           343,262       1,119,009        27,769
240              0
    Total                        9,887                     1,224,663       1,132,362        28,131
1,596,336
    Average
.67

-------------------------------------------------------------------------------------------------------------------------------
-
    Non-Interactive
Workload
       Job         Number          Logical DB      --------- Printer ---------    Communications            CPU Per
```

```
Logical
      Type        Of Jobs         I/O Count           Lines           Pages        I/O Count             Logical I/O      I/O/
Second
    ----------    ---------    ----------------    -------------   -----------    --------------          -----------
----------
    Batch            18,151       1,030,253,068       18,656,603       544,032       1,531,738                 .0001
95,526.4
    Spool                70              1,066           14,933           369               0
.0285             .0
    AutoStart            56             426,047        1,692,060        41,502         178,288                 .0008
39.5
    COLLECTION            1              2,910                0             0               0
.0171             .2
    SQL                 192           3,252,232            3,519            88               0                 .0003
301.5
    MGMTCENTRAL           2              12,229                0             0               0
.0046            1.1
    Total            18,903       1,033,969,357       20,367,115       585,991
1,713,007
    Average                                                                                                      .0003
95,871.0
    Average CPU Utilization . . . . . . . . . . . . .:  61.0
    CPU 1 Utilization . . . . . . . . . . . . . . . .:  55.4
    CPU 2 Utilization . . . . . . . . . . . . . . . .:  57.9
    CPU 3 Utilization . . . . . . . . . . . . . . . .:  61.5
    CPU 4 Utilization . . . . . . . . . . . . . . . .:  62.2
    CPU 5 Utilization . . . . . . . . . . . . . . . .:  62.0
    CPU 6 Utilization . . . . . . . . . . . . . . . .:  60.1
    CPU 7 Utilization . . . . . . . . . . . . . . . .:  61.7
    CPU 8 Utilization . . . . . . . . . . . . . . . .:  63.1
    CPU 9 Utilization . . . . . . . . . . . . . . . .:  55.4
    CPU 10 Utilization. . . . . . . . . . . . . . . .:  56.0
    CPU 11 Utilization. . . . . . . . . . . . . . . .:  59.9
    CPU 12 Utilization. . . . . . . . . . . . . . . .:  60.6
    CPU 13 Utilization. . . . . . . . . . . . . . . .:  60.9
    CPU 14 Utilization. . . . . . . . . . . . . . . .:  62.5
    CPU 15 Utilization. . . . . . . . . . . . . . . .:  63.7
    CPU 16 Utilization. . . . . . . . . . . . . . . .:  64.1
    CPU 17 Utilization. . . . . . . . . . . . . . . .:  54.7
    CPU 18 Utilization. . . . . . . . . . . . . . . .:  57.3
    CPU 19 Utilization. . . . . . . . . . . . . . . .:  59.8
    CPU 20 Utilization. . . . . . . . . . . . . . . .:  60.6
    CPU 21 Utilization. . . . . . . . . . . . . . . .:  61.6
    CPU 22 Utilization. . . . . . . . . . . . . . . .:  62.9
    CPU 23 Utilization. . . . . . . . . . . . . . . .:  63.9
    CPU 24 Utilization. . . . . . . . . . . . . . . .:  64.7
    CPU 25 Utilization. . . . . . . . . . . . . . . .:  57.0
    CPU 26 Utilization. . . . . . . . . . . . . . . .:  55.2
    CPU 27 Utilization. . . . . . . . . . . . . . . .:  66.2
    CPU 28 Utilization. . . . . . . . . . . . . . . .:  61.1
    CPU 29 Utilization. . . . . . . . . . . . . . . .:  62.4
    CPU 30 Utilization. . . . . . . . . . . . . . . .:  63.2
    CPU 31 Utilization. . . . . . . . . . . . . . . .:  66.2
    CPU 32 Utilization. . . . . . . . . . . . . . . .:  66.4

    Total CPU Utilization (Interactive Feature) . .:
.0
    Total CPU Utilization (Database Capability) . .:
51.6
```

Sample Component Report - Job Workload Activity

The Job Workload Activity section of the Component Report gives the total number of transactions, the transactions per hour, the average response time, the number of disk operations, the number of communications operations, the number of PAG faults, the number of arithmetic overflows, and the number of permanent writes for each job.

The values that display in the report header reflect the configuration metrics obtained from the QAPMCONF file when the collection started. These values might change for each interval within a collection period due to dynamic changes in logical partition configuration.

```
                                                        Component Report                                        10/02/03
17:12:15
                                                      Job Workload Activity
Page    6
                                                Perf data from 14:00 to 16:00 at 1 min
Member  . . . : Q275140000 Model/Serial  . : 890/10-3907F
Main storage . . :   56.4  GB Started . . . . :  10/02/03 14:00:00
  Library . . : PTLIBV5R3  System name . . :ABSYSTEM
Version/Release  :    5/ 3.0  Stopped . . . . :  10/02/03 16:00:00
Partition ID  : 003        Feature Code  . :7427-2498-7427
Int Threshold  . :    .00 %
Virtual Processors:   4    Processor Units :   4.0
                            T     P         DB
   Job     User Name/ Job    y     t   CPU   Cpb             Tns            -------- Disk I/O --------   Cmn     PAG  Arith
Perm
   Name    Thread     Number p  Pl y   Util  Util   Tns      /Hour    Rsp     Sync    Async   Logical    I/O    Fault Ovrflw
Write
---------- ---------- ------ -- -- -- ----- ----- ------- --------- ------- -------- -------- -------- -------- ----- ------
------
ADMIN      QTMHHTTP   955725 B  02 25   .02    .0       0         0    .000    14771      615        0        0     0      0
2787
ADMIN      QTMHHTTP   955727 B  02 25   .00    .0       0         0    .000       24        0        0        0     0
0      2
ADMIN      QTMHHTTP   955728 B  02 25   .00    .0       0         0    .000        0        0      165        0     0
0      0
ADMIN      QTMHHTTP   956347 B  02 25   .14    .0       0         0    .000      959      343     1349        0     0      0
736
AMQALMPX   QMQM       955751 B  02 35   .00    .0       0         0    .000        0        0        0        0     0
```

```
0      0
AMQPCSEA   QMQM       955757 B  02 35   .00    .0       0         0    .000        0        0        0        0     0
0      0
AMQRMPPA   QMQM       955773 B  02 35   .01    .0       0         0    .000       14        0        2        0     0
0      0
AMQRRMFA   QMQM       955752 B  02 35   .00    .0       0         0    .000        1        0        0        0     0
0      0
AMQZDMAA   QMQM       955753 B  02 35   .00    .0       0         0    .000        0        0        0        0     0
0      0
AMQZLAA0   QMQM       955755 B  02 20   .02    .0       0         0    .000        7        0        0        0     0
0      0
AMQZLAA0   QMQM       955774 B  02 20   .00    .0       0         0    .000        2        0        0        0     0
0      0
AMQZXMA0   QMQM       955749 B  02 20   .00    .0       0         0    .000        1        0        0        0     0
0      0
CFINT01                     L  01 00   .26    .0       0         0    .000        0        0        0        0     0
0      0
CFINT02                     L  01 00   .06    .0       0         0    .000        0        0        0        0     0
0      0
CFINT03                     L  01 00   .08    .0       0         0    .000        0        0        0        0     0
0      0
CFINT04                     L  01 00   .08    .0       0         0    .000        0        0        0        0     0
0      0
CFINT05                     L  01 00   .00    .0       0         0    .000        0        0        0        0     0
0      0
CFINT06                     L  01 00   .00    .0       0         0    .000        0        0        0        0     0
0      0
COLDQT                      L  01 82   .00    .0       0         0    .000        0        0        0        0     0
0      0
CPUTEST    WLCPU      953645 B  02 51   .00    .0       0         0    .000        0        0        0        0     0
0      0
CPUTEST    WLCPU      953647 B  02 51   .00    .0       0         0    .000        0        0        0        0     0
0      0
CPUTEST    WLCPU      953648 B  02 51   .00    .0       0         0    .000        0        0        0        0     0
0      0
CPUTEST    WLCPU      953649 B  02 51   .00    .0       0         0    .000        0        0        0        0     0
0      0
CPUTEST    WLCPU      953650 B  02 51   .00    .0       0         0    .000        0        0        0        0     0
0      0
Job Name                --  Job name
User Name/Thread        --  User name or secondary thread identifier
Job Number              --  Job number
Typ                     --  Job type
Pl                      --  Pool that the job ran in
Pty                     --  Priority of the job
CPU Util                --  Percentage of available CPU time used by the job.  This is the average of all processors
DB Cpb Util             --  Percentage of database capability used by the job to perform database processing
Tns                     --  Total number of transactions for the job
Tns /Hour               --  Transactions per hour
Rsp                     --  Average interactive transaction response time in seconds
Sync Disk I/O           --  Number of synchronous disk operations (reads and writes)
Async Disk I/O          --  Number of asynchronous disk operations (reads and writes)
Logical Disk I/O        --  Number of logical disk operations (Get, Put, Upd, Other)
Cmn I/O                 --  Number of communications operations (Get, Put)
PAG Fault               --  Number of faults involving the Process Access Group
Arith Ovrflw            --  Number of arithmetic overflow exceptions
Perm Write              --  Number of permanent writes
.
.
.
Column                        Total               Average
        --------------------------  ----------------   ----------------
        CPU Util                                                98.740 *
        DB Cpb Util                                             82.3
        Tns                                     2,099
        Tns /Hour                               1,043
        Resp                                                    1.610
        Sync Disk I/O                         304,001
        Async Disk I/O                      1,906,898
        Logical Disk I/O                    6,257,174
        Cmn I/O                                     0
        PAG Fault                                   0
        Arith Ovrflw                                3
        Perm Write                          1,980,564
        *  ----  Average based on the total elapsed time for the selected intervals
```

Printing the performance reports

You can print reports using the performance data that you collected.

Note: If your trace data and sample data are both in the current library, you can use F20 to toggle between the two Print Performance Report displays.

After you have collected your data, you must create a set of performance data files from the performance information stored in a management collection (*MGTCOL) object. Use the Create Performance Data (CRTPFRDTA) command. After you have created the data files, you can request to print your reports.

Use the following commands to print reports for sample data that you collected with Collection Services:

- Print System Report (PRTSYSRPT)
- Print Component Report (PRTCPTRPT)
- Print Job Interval Report (PRTJOBRPT)
- Print Pool Report (PRTPOLRPT)
- Print Resource Report (PRTRSCRPT)

Use the following commands to print reports for trace data that you collected with the Start Performance Trace (STRPFRTRC) and Trace Internal (TRCINT) commands:

- Print Transaction Report (PRTTNSRPT)
- Print Lock Report (PRTLCKRPT)
- Print Job Trace Report (PRTTRCRPT)

Note: You must use the End Performance Trace (ENDPFRTRC) command to stop the collection of performance trace data and then optionally write performance trace data to a database file before you can print the Transaction reports.

Related reference

CL commands for performanceThe operating system includes several CL commands to help you manage and maintain system performance.

Performance Report columns

Each report includes columns of information. Look here for descriptions of that information.

>8.0
(Component) The number of times the response time was greater than 8 seconds.

%Write Cache Overruns
(Component) Percent of Write Cache Overruns during the collection interval.

---------- (pgmname)
(Transaction) The transaction totals record. For example, ---------- QUYLIST,. This report line occurs each time the job has an active-to-wait transaction. Totals are created for Rsp* (response time), *CPU Secs*, and I/O counts for the transaction.

A-I Wait /Tns
(Transaction) The average time, in seconds, of active-to-ineligible wait time per transaction. If this value is high, it may be because the time-slice value is set too low for many of the interactive jobs. Consider increasing the time slice-value.

Aborts Recd
(Resource Interval) The number of frames received that contained HDLC abort indicators. This indicates that the remote equipment ended frames before they were complete.

Act Jobs
(Job Interval) The number of selected jobs (interactive or noninteractive, depending on the report section) that were active during the interval.

Act Level
(Component) Initial pool activity level.

Act Lvl
(System, Pool Interval) Activity level. For the Pool Activity section of the Pool Interval Report, the activity level of the pool during the interval. For the Storage Pool Utilization section of the System Report, the activity level at the time of the first sample interval.

Act-Inel
(System, Component) Average number of active-to-ineligible job state transitions per minute.

Act-Wait
(System, Component) Number of transitions per minute from active state to wait state by processes assigned to this pool.

ACTIVE
(Job Trace) The time the job was processing.

Active Devices
(System) Average number of active devices on the line.

Active display stations (local or remote)
(System) The number of local or remote display stations entering transactions during the measurement period.

Active Jobs
(Transaction) The number of interactive jobs that were active during the interval.

Active Jobs Per Interval
(System) Average number of jobs of this type that were active per sample interval.

Active K/T /Tns
(Transaction) An average think time and keying time (or the delay time between the end of one transaction and the start of the next transaction), in seconds, for the active work stations (described under Est of AWS). Active K/T /TNS delay time differs from Key/Think /TNS delay time in that any delay time greater than 600 seconds has been rounded to 600 seconds. This technique is used to reduce the effect of very casual users (those who may do intermittent work or leave their work stations for long periods of time) on the estimate of active work stations.

Active Wrk Stn
(Resource Interval) The number of work stations with activity.

Active/Rsp
(Transaction) The time the job spends (either waiting or active) during transaction processing, while it holds an activity level.

Activity level
(System) The sum of activity levels for all interactive pools that had interactive job activity running in them.

Activity Level Time
(Transaction) A breakdown of the transaction time spent *ACTIVE*, waiting on a *SHORT WAIT*, and waiting on a *SEIZE/CFT* (seize conflict). The *SHORT WAIT* and *SEIZE CFT* time are included under *ACTIVITY LEVEL TIME*, because the activity-level slot is not given up during these times. Note that the seize conflict time is included in the active time, not added to it to get transaction/response time, as is the case for waiting time.

Arith Ovrflw
(Component, Job Interval) The number of arithmetic overflow exceptions that occurred for the selected interactive jobs during the interval.

ASP ID
(System, Resource Interval) Auxiliary storage pool identifier.

ASP Rsc Name
(System, Resource) Identifies the ASP resource name to which the disk unit was allocated at collection time.

Async
(System, Component, Transaction, Job Interval) The number of asynchronous disk I/O operations started by the selected interactive jobs during the interval. The job that starts the I/O operation may continue processing without having to wait for the I/O operation to complete. The I/O operation is completed by a background system test.

Async DIO /Tns
(Transaction) The sum of the averages of the asynchronous DB READ, DB WRITE, NDB READ, and NDB WRITE requests (the average number of asynchronous I/O requests per transaction for the job).

Async Disk I/O
(System, Component, Transaction) Number of asynchronous disk input/output operations per transaction.

Async Disk I/O per Second
(Component) Average asynchronous disk I/O operations per second.

Async Disk I/O Requests
(Transaction) The total number of asynchronous disk I/O requests for the given combination of priority, job type, and pool.

Async I/O /Sec
(Job Interval) The average number of asynchronous disk I/O operations started per second by the job during the interval. This is calculated by dividing the asynchronous disk I/O count by the elapsed time.

Async I/O Per Second
(Job Interval) The average number of asynchronous disk I/O operations started per second by the selected noninteractive jobs during the interval.

Async Max
(Transaction) Listed under Average DIO/Transaction, the maximum number of asynchronous DBR, NDBR, and WRT I/O requests encountered for any single transaction by that job. If the job is not an interactive or autostart job type, the total disk I/O for the job is listed here.

Async Sum
(Transaction) Listed under Average DIO/Transaction, the sum of the averages of the asynchronous DBR, NDBR, and WRT requests (the average number of asynchronous I/O requests per transaction for the job).

Asynchronous DBR
(System, Job Interval, Pool Interval) The average number of asynchronous database read operations on the disk per transaction for the job during the intervals. This is calculated by dividing the asynchronous database read count by the transactions processed. This field is not printed if the jobs in the system did not process any transactions. For the Resource Utilization section of the System Report, it is the number of asynchronous database read operations per second.

Note: The asynchronous I/O operations are performed by system asynchronous I/O tasks.

Asynchronous DBW
(System, Job Interval) The average number of asynchronous database write operations on the disk per transaction for the selected jobs during the interval. This is calculated by dividing the asynchronous database write count by the transactions processed. This field is not printed if the jobs in the system did not process any transactions. For the Resource Utilization section of the System Report, it is the number of asynchronous database read operations per second.

Note: The asynchronous I/O operations are performed by system asynchronous I/O tasks.

Asynchronous disk I/O per transaction
(System) The average number of asynchronous physical disk I/O operations per interactive transaction.

Asynchronous NDBR
(System, Job Interval, Pool Interval) The average number of asynchronous nondatabase read operations per transaction for the jobs in the system during the interval. This is calculated from the asynchronous nondatabase read count divided by the transactions processed. This field is not printed if the jobs in the system did not process any transactions. For the Resource Utilization section of the System Report, it is the asynchronous nondatabase read operations per second.

Note: The asynchronous I/O operations are performed by system asynchronous I/O tasks.

Asynchronous NDBW
(System, Job Interval, Pool Interval) The average number of asynchronous nondatabase write operations per transaction for the jobs in the system during the interval. This is calculated from the asynchronous nondatabase write count divided by the transactions processed. This field is not printed if the jobs in the system did not process any transactions. For the Resource Utilization section of the System Report, it is the number of asynchronous nondatabase write operations per second.

Note: The asynchronous I/O operations are performed by system asynchronous I/O tasks.

Avail Local Storage (K)
(Resource Interval) The number of kilobytes of free local storage in the IOP.

Available Storage
(Component) Available local storage (in bytes). The average number of bytes of available main storage in the IOP. The free local storage is probably not joined because it has broken into small pieces.

Average
(Transaction) The average value of the item described in the column for all transactions.

AVERAGE
(Job Trace) Averages for the fields. The entry on the AVERAGE line in the SEQUENCE column shows the number of STRTNS and ENDTNS pairs encountered. For an interactive job, this is the number of transactions entered while the trace was on if the default STRTNS and ENDTNS values were used.

Average Disk Activity Per Hour
(Component) See Disk Arm Seek Distance

Average DIO/Transaction
(Transaction) Seven columns of information about physical disk I/O counts. Physical I/O contrasts with logical I/O shown elsewhere in these reports. A logical I/O is a request sent at the program level that might result in an access to auxiliary storage (DASD). A physical I/O refers to those requests that actually result in access to auxiliary storage.

- Synchronous DBR
- Synchronous NDBR
- Synchronous Wrt
- Synchronous Sum
- Synchronous Max
- Async Sum
- Async Max

Average K per I/O
(Resource Interval) The average number of kilobytes transferred during each disk read or write operation.

Average Phys I/O /Sec
(Resource Interval) The average number of physical disk read and write operations per second made on all disks on the system.

Average Reads/Sec
(Resource Interval) The average number of physical disk read operations per second made on all disks on the system.

Average Response
(System) Average response time (in seconds) for interactive transactions. The Total/Average interactive response time does not include transactions for DDM server jobs.

Average Response Time
(System) Average disk response time per I/O operation.

Average Response Time (seconds)
(System) The average interactive response time.

Average Service Time
(System) Average disk service time per I/O operation. This is the amount of time a request would take if there were no contention.

Average Wait Time
(System) Average disk wait time per I/O operation. Normally due to contention.

Average Writes/Sec
(Resource Interval) The average number of physical disk write operations per second made on all disks on the system.

Avg CPU /Tns
(Transaction) The average number of processing unit seconds per transaction that fell in the given category.

Avg K/T /Tns
(Transaction) The average think time and keying time (or the delay time between transaction boundaries), in seconds, for the interactive jobs.

Avg Length
(Lock) The average number of milliseconds a lock or seize was held.

Avg Rsp (Sec)
(Transaction) The average transaction response time in seconds.

Avg Rsp /Tns
(Transaction) The average response per transaction (in seconds) for the transactions that fell into the given category.

Avg Rsp Time
(Component) Average transaction response time.

Avg Sec Locks
(Transaction) The average length of a lock in seconds attributed to interactive or noninteractive waiters.

Avg Sec Seizes
(Transaction) The average length of a seize in seconds attributed to interactive or noninteractive waiters.

Avg Time per Service
(Resource Interval) The amount of time a disk arm uses to process a given request.

Avg Util
(System, Resource Interval) On the Disk Utilization Summary of the Resource Report, the average percentage of available time that disks were busy. It is a composite average for all disks on the system. On the Communications Summary of the System Report, the average percentage of line capacity used during the measured time interval.

Batch asynchronous I/O per second
(System) The average number of asynchronous physical disk I/O operations per second of batch processing.

Batch CPU seconds per I/O
(System) The average number of system processing unit seconds used by all batch jobs for each I/O performed by a batch job.

Batch CPU Utilization
(Component) Percentage of available processing unit time used by the jobs that the system considers to be batch.

Note: For a multiple-processor system, this is the average use across all processors.

Batch impact factor
(System) Batch workload adjustment for modeling purposes.

Batch permanent writes per second
(System) The average number of permanent write operations per second of batch processing.

Batch synchronous I/O per second
(System) The average number of synchronous physical disk I/O operations per second of batch processing.

BCPU / Synchronous DIO
(Transaction) The average number of batch processor unit seconds per synchronous disk I/O operation.

Bin
(Transaction) The number of binary overflow exceptions.

Binary Overflow
(Component) Number of binary overflows per second.

BMPL - Cur and Inl
(Transaction) The number of jobs currently in the activity level (beginning current multiprogramming level), and the number of jobs on the ineligible queue (beginning ineligible multiprogramming level) for the storage pool that the job ran in when the job left the wait state (the beginning of the transaction).

Note: Multiprogramming level (MPL) is used interchangeably with activity level.

Bundle Wait Count
(Component) Total number of times the tasks and jobs waited for journal bundles to be written to disk.

Bundle Wait Pct
(Component) Percentage of time (relative to the interval elapsed time) spent waiting for journal bundles to be written to disk.

Bundle Writes System
(Component) Number of bundle writes to internal system journals. A bundle write is a group of journal entries which are deposited together by the system.

Bundle Writes User
(Component) Number of bundle writes to user-created journals. A bundle write is a group of journal entries which are deposited together by the system.

Bytes per Second Received
(System) Average number of bytes received per second.

Bytes per Second Transmitted
(System) Average number of bytes transmitted per second.

Bytes Recd per Sec
(Resource Interval) The average number of bytes received per second.

Bytes Trnsmitd per Sec
(Resource Interval) The average number of bytes transmitted per second.

Category
(Transaction) A group of transactions categorized together. In the Analysis by Interactive Transaction Category, the transactions are categorized by the processing unit model. The boundary values that are used to separate the transactions are given in the *Avg CPU /Tns* column. For the Analysis by Interactive Response Time, they are categorized by their response time. For the Analysis by Interactive Key/Think Time, they are categorized by their key/think time.

Cache Hit Statistics
(Component) Statistics data about use of cache including:

- The percent of Device Cache Read Hit for each arm.
- The percent of Controller Cache Read Hit for each arm.
- The percent of efficiency of write cache

Device read
Device Read is the number of Device Cache Read Hits (DSDCRH) divided by number of Device Read Operations (DSDROP), expressed as a percent

Controller read
Controller Read is the number Controller Cache Read Hits (DSCCRH) divided by number of Read Commands (DSRDS), expressed as a percent.

Write efficiency
Write efficiency is the difference between Write Commands (DSWRTS) and Device Write Operations (DSDWOP) divided by Write Commands (DSWRTS), expressed as a percent.

EACS Read
The percent of read hits by the Extended Adaptive Cache Simulator.

EACS Resp
The percent of response time improvement by the Extended Adaptive Cache Simulator.

Capped
(System) Indicates whether the partition was capped or uncapped at the end of each interval. This column is only printed for the IBM i partition collecting performance data.

Channel
(Resource Interval) The B-channel used by the IDLC line. (special condition)

Cmn
(Job Interval) The number of communications I/O operations performed by the selected interactive jobs during the interval.

Cmn I/O
(Component) Number of communications operations (Get, Put).

Cmn I/O Per Second
(Job Interval) The average number of communications I/O operations performed per second by the selected noninteractive jobs during the interval.

Collision Detect
(Resource Interval) The number of times that the terminal equipment (TE) detected that its transmitted frame had been corrupted by another TE attempting to use the same bus.

Commit Ops
(Component) Commit operations performed. Includes application and system-provided referential integrity commits.

Communications I/O Count
(System) Number of communications I/O operations.

Communications I/O Get
(System) Number of communication get operations per transaction.

Communications I/O Put
(System) Number of communication put operations per transaction.

Communications Lines
(System, Component, Job Interval, Pool Interval) For the Report Selection Criteria, the list of communications lines selected to be included (SLTLINE parameter) or excluded (OMTLINE parameter). These are the communications line names you specify.

Control Units
(System, Component, Job Interval, Pool Interval) The list of control units selected to be included (SLTCTL parameter) or excluded (OMTCTL parameter). These are the controller names you specify.

Count
(Transaction, Lock) The number of occurrences of the item in the column. For example, in a lock report, it is the number of locks or seizes that occurred.

CPU
(Transaction) The total processing unit seconds used by the jobs with a given priority.

CPU
(Job Trace) The approximation of the CPU used on this trace entry. This is a calculated value based on the time used and the CPU model being run.

CPU /Tns
(Transaction, Job Interval) The amount of available processing unit time per transaction in seconds.

CPU Model
(System) The processing unit model number.

CPU per I/O Async
(System) CPU use per asynchronous I/O.

CPU per I/O Sync
(System) CPU use per synchronous I/O.

CPU per Logical I/O
(System) Processing unit time used for each logical disk I/O operation.

CPU QM
(Transaction) The simple processing unit queuing multiplier.

CPU Sec
(Transaction) The processing unit time used by the job in this state.

CPU Sec /Sync DIO
(Transaction) The ratio of CPU seconds divided by synchronous disk I/O requests for each type of job.

CPU Sec Avg and Max
(Transaction) The average processing unit time per transaction for the job and the largest processing unit time used for a transaction in the job. If the job is not an interactive or autostart job type, then only the total processing unit time for the job is listed under the MAX column heading.

CPU Sec per Tns
(Transaction) The processing unit time per transaction.

CPU Seconds
(System, Transaction, Component) Average processing unit seconds used per transaction. For System Summary Data, it is the total available processing unit time used by the jobs during the trace period. For Priority-Jobtype-Pool Statistics, it is the total processing unit seconds used by the jobs with a given combination of priority, job type, and pool. For Batch Job Analysis, it is the amount of available processor unit time used by the job in seconds. For Concurrent Batch Job Statistics, it is the amount of available processor unit time used by the jobs in the job set in seconds.

CPU SECONDS
(Job Trace) The approximate processing unit time used for the transaction.

CPU seconds per transaction
(System) The average processing unit seconds per transaction.

CPU Util
(System, Component, Transaction, Job Interval, Pool Interval, Batch Job Trace) Percentage of available processing unit time used. For multiple-processor systems, this is the total utilization divided by the number of processors.

CPU Util per Transaction
(Component) The result of the CPU Utilization divided by the total number of transactions for the job.

CPU Utilization (Batch)
The percentage of available CPU time that is used by batch jobs. This is the average of all processors.

CPU Utilization (Interactive)
The percentage of available CPU time that is used by interactive jobs. This is the average of all processors.

CPU Utilization (Total)
The percentage of available CPU time that is used by interactive and batch jobs. This is the average of all processors.

Note: For uncapped partitions, the Total CPU utilization might exceed 100 percent.

CPU/Async I/O
(Job Interval) The average number of milliseconds of processing unit time taken for each asynchronous disk I/O operation. This is calculated by dividing the milliseconds of the processing unit time the job used by the asynchronous disk I/O count.

CPU/Sync I/O
(Job Interval) The average number of milliseconds of processing unit time taken for each synchronous disk I/O operation. This is calculated from the milliseconds of the processing unit time used by the job divided by the synchronous disk I/O count.

CPU/Tns
(Transaction) The average number of processing seconds per transaction for the job during the interval. This is calculated from the amount of processing unit time used divided by the number of transactions processed.

Cpu/Tns (Sec)
(Transaction) The number of processing unit seconds per transaction.

Ctl
(Component) Controller identifier.

Cum CPU Util
(Transaction) The cumulative percentage of available processing unit time used by the transactions that have an average response time per transaction equal to or less than the given category. For

example, in CPU by Priority for All Jobs for Total Trace Period (System Summary Data), it is the unit time used by the jobs with a priority higher or equal to the given priority.

Cum Pct Tns
(Transaction) Cumulative CPU percent per transaction. For system summary data, it is the cumulative CPU percentage of all transactions that have an average response time per transaction equal to or less than the given category. For Interactive Program Transactions Statistics, it is the cumulative CPU percentage of all transactions through the listed program. For Job Statistics section, it is the cumulative CPU percentage of total transactions through the listed job. For Interactive Program Statistics section, it is the cumulative CPU percentage of all transactions through the listed program.

Cum Util
(System) Cumulative CPU use (a running total).

Note: This is taken from the individual jobs and may differ slightly from the total processing unit use on the workload page.

Cur Inl MPL
(Transaction) The number of jobs waiting for an activity level (ineligible) in the storage pool.

Cur MPL
(Transaction) The number of jobs holding an activity level in the storage pool.

Current User
(Job) The user under which the job was running at the end of each interval.

DASD Ops/Sec
(Component) Disk operations per second.

DASD Ops Per Sec Reads
(Resource) Number of reads per second

DASD Ops Per Sec Writes
(Resource) Number of writes per second

Datagrams Received
(Component) The total number of input datagrams received from interfaces. This number includes those that were received in error.

DB
(Job Trace) The number of physical database reads that occurred for the entry.

DB Fault
(System, Component) Average number of database faults per second.

DB Pages
(System, Component) Average number of database pages read per second.

DB Read
(Transaction) When listed in Physical I/O Counts column, it is the number of database read requests while the job was in that state. When listed in the Sync Disk I/O Rqs/Tns column, it is the average number of synchronous database read requests per transaction.

DB READS
(Job Trace) The number of physical database reads that occurred.

DB Write
(Transaction) When listed in the Sync Disk I/O Rqs/Tns column, it is the average number of synchronous database write requests per transaction.

DB Wrt
(Transaction) When listed in the Physical I/O Counts column, it is the number of database write requests while the job was in that state. When listed in the Synchronous Disk I/O Counts column, it is the number of synchronous database write requests per transaction.

DDM I/O
(Component, Job Interval) The number of logical database I/O operations for a distributed data management (DDM) server job.

DDM Svr Wait /Tns
(Transaction) The average time, in seconds, that a source distributed data management (DDM) server job spent waiting for the target system to respond to a request for data per transaction. This value includes line time and time spent by the target system responding to the request for data.

Dec
(Transaction) The number of decimal overflow exceptions.

Decimal Data
(Component) Data exception count per second. A data exception occurs when data that is not valid is detected by arithmetic instructions. Examples are signs or digit codes that are not valid in decimal instructions, or an insufficient number of farthest left zeros in multiply instructions.

Decommit Ops
(Component) Decommit operations performed. Includes application and system-provided referential integrity decommits.

Decimal Overflow
(Component) Number of decimal overflows per second.

Description
(Component) More detailed description of the exception type.

Detected Access Transmission Error (DTSE) In
(Resource Interval) The number of times the network termination 1 (NT1) end point notified the terminal equipment (TE) of an error in data crossing the ISDN U interface from the line transmission termination (LT) to the NT1 end point. The NT1 end point reports the errors to the TE through the maintenance channel S1.

Detected Access Transmission Error (DTSE) Out
(Resource Interval) The number of times the network termination 1 (NT1) end point notified the terminal equipment (TE) of an error in data crossing the ISDN U interface from the NT1 end point to the LT. The NT1 end point reports the errors to the TE through the maintenance channel S1.

Device
(Component) Device identifier.

DIO/Sec Async
(System) Number of asynchronous I/O operations per second.

DIO/Sec Sync
(System) Number of synchronous I/O operations per second.

Disk Arm Seek Distance
(Component) Average seek distance distributions per hour:

0
Number of zero seeks

1/12
Number of seeks between 0 and 1/12 of the disk

1/6
Number of seeks between 1/12 and 1/6 of the disk

1/3
Number of seeks between 1/6 and 1/3 of the disk

2/3
Number of seeks between 1/3 and 2/3 of the disk

>2/3
Number of seeks greater than 2/3 of the disk

Disk Arms
(System) The number of disk arms for this IOP.

Disk Capacity
(Component) Average amount of disk space used or available.

MB
Millions of bytes available on the disk.

Percent
Percent of space available on the disk.

Disk Controllers
(System) The number of disk storage controllers for this IOP.

Disk Feature
(System) The type of disk (9332, 9335, and so on).

Disk I/O Async
(System, Component) Total number of asynchronous disk I/O operations.

Disk I/O Logical
(Component) The number of logical disk operations, such as gets and puts.

Disk I/O per Second
(System) Average number of physical disk I/O operations per second.

Disk I/O Reads /Sec
(Resource Interval) The average number of disk read operations per second by the disk IOP.

Disk I/O Requests
(Transaction) The total number of synchronous and asynchronous disk I/O requests issued by the jobs during the trace period.

Disk I/O Sync
(System, Component) Total number of synchronous disk I/O operations.

Disk I/O Writes /Sec
(Resource Interval) The average number of disk write operations per second by the disk IOP.

Disk IOPs
(System) The number of disk IOP controllers.

Disk mirroring
(System) Indicates whether disk mirroring is active.

Disk Space Used
(Resource Interval) The total disk space used in gigabytes for the entire system.

Disk transfer size (KB)
(System) The average number of kilobytes transferred per disk operation.

Disk utilization
(System) The fraction of the time interval that the disk arms were performing I/O operations.

Dsk CPU Util
(System, Resource Interval) The percentage of CPU used by the disk unit.

Dtgm Req Transm Dscrd
(Component) The percentage of IP datagrams that are discarded because of the following reasons:

- No route was found to transmit the datagrams to their destination.
- Lack of buffer space.

Dtgm Req for Transm Tot
(Component) The total number of IP datagrams that local IP user-protocols supplied to IP in requests for transmission.

Elapsed Seconds
(Transaction, Component) The elapsed time in seconds. For the Batch Job Analysis section of the Transaction Report, it is the number of seconds elapsed from when the job started to when the job ended. For the Concurrent Batch Job Statistics section of the Transaction Report, it is the total elapsed time of all jobs in that job set.

Elapsed Time
(Job Interval) The amount of time (minutes and seconds) for which the job existed during the interval. This is the same as the interval length unless the job started or ended during the interval, in which case it is less.

Elapsed Time--Seconds
(Transaction) Shows the time spent by the job, in the following columns:

Long Wait
Elapsed times in the state (such as waiting for the next transaction or lock-wait time).

Active/Rsp
During transaction processing, the time the job spends (either waiting or active) while it holds an activity level. At the end of a transaction (on the transaction totals line), this is the time the job spent processing the transaction in an activity level, for long waits caused by locks, and in the ineligible state.

Inel Wait
The time the job spent in the ineligible wait state waiting for an activity level.

EM3270 Wait /Tns
(Transaction) The average, in seconds, of the time spent waiting on the host system communications for Systems Network Architecture (SNA) and binary synchronous communications (BSC) 3270DE per transaction. Program logic is required to determine if the emulation program is communicating with the display or the host processing unit. Because there are requirements on event-wait processing, not all transition combinations can be detected.

ENTRY
(Job Trace) The instruction in the program where the program was given control.

EORn
(Transaction) Listed in the Wait Code column, End of response time for transaction n. These codes are in the wait code column, but they are not wait codes. They indicate transaction boundary trace records.

EOTn
(Transaction) Listed in the Wait Code column, End of transaction for transaction for type n. These codes are in the wait code column, but they are not wait codes. They indicate transaction boundary trace records.

Estimated Exposr AP Not Jrnld
(Component) System-estimated access path recovery time exposure in minutes if no access paths were being journaled by the system.

Estimated Exposr Curr System
(Component) System-estimated access path recovery time exposure in minutes.

Est Of AWS
(Transaction) An estimate of the number of active work stations for the trace period or interval. Any delay time greater than 600 seconds has been rounded to 600 seconds. This technique is used to reduce the effect of very casual users (those who may do intermittent work or leave their work stations for long periods of time) on the estimate of active work stations.

Event Wait /Tns
(Transaction) The average time, in seconds, of the event-wait time per transaction. Often requests made by a job that runs on the system are made to asynchronous jobs. These asynchronous jobs use an event to signal completion of the request back to the requester. The event-wait time is the time the requesting job waits for such a signal.

EVT
(Transaction) Listed in the Wait Code column, Event Wait. This is a long wait that occurs when waiting on a message queue.

Exception Type
(Component) Type of program exception that results from the internal microprogram instructions being run in internal microprogram instructions procedure. Because these exceptions are monitored at a low level within the system, it is difficult to associate these exceptions with specific end-user

operations. The counts are meaningful when the processing unit time required to process them affects system performance. A variation in the counts may indicate a system change that could affect performance. For example, a large variation in seize or lock counts may indicate a job scheduling problem or indicate that contention exists between an old application and a new one that uses the same resources.

Note: To see the seize and lock counts, you should collect the trace data by using the Start Performance Trace (STRPFRTRC) command. Run the Print Transaction Report (PRTTNSRPT) to list the objects and jobs that are holding the locks.

Exceptional wait
(System) The average exceptional wait time, in seconds, per transaction. An *exceptional wait* is that portion of internal response time that cannot be attributed to the use of the processor and disk. An exceptional wait is caused by contention for internal resources of the system, for example, waiting for a lock on a database record.

Constant
The portion of exceptional wait time held constant as throughput increases.

Variable
The portion of exceptional wait time that varies as throughput increases.

Excp
(Component, Transaction) For the Component Report, it is the total number of program exceptions that occurred per second. For the Transaction Report, a Y in this column means that the transaction had exceptions. The types of exceptions that are included are process access group exceptions, and decimal, binary, and floating point overflow. See the Transition Report to see which exceptions the transaction had.

Excp Wait
(Transaction) The amount of exceptional wait time for the jobs in the job set in seconds.

Excp Wait /Tns
(Transaction) The average exceptional wait time, in seconds, per transaction. This value is the sum of those waits listed under the Exceptional Wait Breakdown by Job Type part.

Excp Wait Sec
(Transaction) The total amount of exceptional wait time in seconds for the job.

Excs ACTM /Tns
(Transaction) The average time, in seconds, of the excess activity level time per transaction (for example, time spent in the active state but not using the processing unit). If enough activity levels are available and there is plenty of interactive work of higher priority to do, a job waits longer for processing unit cycles. If the value is greater than .3, look at jobs that correspond to particular applications for more information. By looking at these jobs, you might be able to determine which application's jobs are contributing most to this value. Use the Transaction and Transition Reports for these jobs for additional information. The formula for excessive activity-level time is shown below:

```
Active Time - [
(multiplier X CPU X Beginning Activity Level) +
(Number of synchronous disk I/O operations X .010)]
```

Note: If the beginning activity level is greater than 1, the multiplier equals 0.5. If the beginning activity level is any other value, the multiplier equals 1.

EXIT
(Job Trace) The instruction number in the program where the program gave up control.

Expert Cache
(System, Component) Directs the system to determine which objects or portions of objects should remain in a shared main storage pool based on the reference patterns of data within the object. Expert cache uses a storage management tuner, which runs independently of the system dynamic tuner, to examine overall paging characteristics and history of the pool. Some values that you might see in this column are associated with the Work with Shared Pools (WRKSHRPOOL) command:

- 0=*FIXED, which indicates the system does not dynamically adjust the paging characteristics of the storage pool. The system uses default values.
- 3=*CALC, which indicates the system dynamically adjusts the paging characteristics of the storage pool for optimum performance.

Exposed AP System Journaled
(Component) The number of exposed access paths currently being journaled by the system.

Exposed AP System Not Journaled
(Component) The number of exposed access paths currently not being journaled by the system.

/F
(System, Resource Interval) The line speed of the protocol reported as full duplex. This indicator applies to the line speeds for an Ethernet (ELAN) token-ring (TRLAN) line, or an asynchronous transfer mode line.

Far End Code Violation
(Resource Interval) The number of unintended code violations detected by the network termination 1 (NT1) end point for frames transmitted to the NT1 end point on the interface for the T reference point. The NT1 end point reports a violation to the termination equipment (TE) through the maintenance channel S1.

Faults
(System) A value that represents the total page faults that occurred for each job type or job priority during the collection. This is the same value as shown in the JBTFLT field of the QAPMJOBS or QAPMJOBL file.

File
(Transaction) The file that contains the object.

Flp
(Transaction) The number of floating point overflow exceptions.

Flp Overflow
(Component) Number of floating point overflows per second.

Frame Retry
(Resource Interval) The number of attempts to retransmit a frame to a remote controller.

Frames Received Pct Err
(Resource Interval) The percentage of frames received in error. Errors can occur when the host system has an error or cannot process received data fast enough.

Frames Received Total
(Resource Interval) The total number of frames received including frames with errors and frames that are not valid.

Frames Transmitted Pct Err
(Resource Interval) The percentage of frames retransmitted due to error.

Frames Transmitted Total
(Resource Interval) The total number of frames transmitted.

FULL CLS
(Job Trace) The number of full closes for all types of files.

FULL OPN
(Job Trace) The number of full opens for all types of files.

FUNCTION
(Job Trace) This causes the trace entry to be recorded. The possible trace entries are as follows:

Table 7.

Function ID	Description
DATA	Data trace record
CALL	Call external

Table 7. (continued)	
Function ID	**Description**
XCTL	Transfer control
EVENT	Event handler invocation
EXTXHINV	External exception handler invocation
INTXHINV	Internal exception handler invocation
INTXHRET	Return from internal exception handler
INVEXIT	Invocation exit
RETURN	Return external
ITRMXRSG	Invocation ended due to resignaling exception
EXTXHRET	Return external or from a procedure instruction
PTRMTPP	Termination phase end
PTRMUNX	End process due to an unhandled exception
NOTUSED	This type is a non-valid trace type
ITERM	Invocation ended
CANCLINV	Cancel invocation instruction

Functional Areas
(System, Component, Transaction, Job Interval, Pool Interval) For Report Selection Criteria, the list of functional areas selected to be included (SLTFCNARA parameter) or excluded (OMTFCNARA parameter).

/H
(System, Resource Interval) The line speed of the protocol reported as half duplex. This indicator applies to the line speeds for an Ethernet (ELAN) token-ring (TRLAN) line, or an asynchronous transfer mode line.

HDW
(Transaction) Listed in the Wait Code column, Hold Wait (job suspended or system request). The job released a lock it had on the object named on the next detail line of the report (OBJECT --). The job that was waiting for the object is named on this line (WAITER --) along with the amount of time the job spent waiting for the lock to be released.

High Srv Time
(Resource Interval) The highest average service time in seconds for a disk arm in the system.

High Srv Unit
The disk arm with the highest service time.

High Util
(Resource Interval) The percentage of use for the disk arm that has the highest utilization.

High Util Unit
(Component, Resource Interval) The disk arm with the highest utilization.

High Utilization Disk
(Component) Percent of utilization of the most utilized disk arm during this interval.

High Utilization Unit
(Component) Disk arm that had the most utilization during this interval.

Holder Job Name
(Transaction) The name of the job that held the object.

Holder Number
(Transaction) The number of the job that held the object.

Holder Pool
(Transaction) The pool that held the job while it was running.

Holder Pty
(Transaction) The priority of the holder's job.

Holder Type
(Transaction) The type and subtype of the holder's job.

Holder User Name
(Transaction) The name of the user that held the object.

Holder's Job Name
(Lock) The name of the job holding the lock.

I Frames Recd per Sec
(Resource Interval) The number of information frames received per second.

I Frames Trnsmitd per Sec
(Resource Interval) The number of information frames transmitted per second.

I/O Wait
(Resource Interval) The amount of time in which a given I/O request is ready to be processed, but the disk arm is not yet available to perform the request.

ICMP Messages Error
(Component) This is the number of Internet Control Message Protocol (ICMP) messages that the entity received but determined that the messages had errors or are messages that the entity did not send due to problems.

ICMP Messages Received
(Component) This is the total number of Internet Control Message Protocol (ICMP) messages that the entity received.

ICMP Messages Sent
(Component) This is the total number of Internet Control Message Protocol (ICMP) messages that the entity attempted to send.

Incoming Calls Pct Retry
(Resource Interval) The percentage of incoming calls that were rejected by the network.

Incoming Calls Total
(Resource Interval) The total number of incoming call attempts.

Inel Time A-I/W-I
(Transaction) The amount of time the job spent in the ineligible state, either coming from time slice end (active-to-ineligible) or from the wait state (wait-to-ineligible).

Inel Wait
(Transaction) Listed in the Elapsed Time--Seconds column, the time the job spent in the ineligible wait state waiting for an activity level.

Int Feat Util
(Component) The percentage of Interactive Feature that is used by all jobs.

Inter CPU Utilization
(Component) Percentage of available processing unit time used by the jobs that the system considers to be interactive.

Note: For a multiple-processor system, this is the average use across all processors.

INV
(Job Trace) The call level of the program.

IOP
(Component) Input/output processor (IOP) Resource name and model number for each communications IOP, DASD IOP, local workstation IOP, and multifunction IOP. Communications IOP is

the percent of CPU used in the IOP. The percent does not necessarily mean that the IOP is doing any data transfers. Some of the percent can be attributed to overhead of an active line.

IOP Name/Line
(System, Resource Interval) Input/output (IOP) processor resource name and model number line.

IOP Name(Model)
(Resource Interval) The input/output processor (IOP) identification and the model number in parentheses.

IOP Name
(System, Component) Input/Output processor (IOP) resource name.

IOP Name Network Interface
(Resource Interval) The IOP name of the network interface.

IOP Processor Util Comm
(Component, Resource) Utilization of IOP due to communications activity.

IOP Processor Util LWSC
(Component, Resource) Utilization of IOP due to local workstation activity.

IOP Processor Util DASD
(Component, Resource) Utilization of IOP due to DASD activity.

IOP Processor Util Total
(Component, Resource Interval) The total percent of utilization for each local workstation, disk, and communications IOP.

IOP Util
(System) For the Disk Utilization section of the System Report, it is the percentage of utilization for each input/output processor (IOP).

Note: For the multifunction I/O processors, this is utilization due to disk activity only, not communications activity. For the System Model Parameter section it is the fraction of the time interval the disk IOP was performing I/O operations.

Itv End
(Component, Transaction, Job Interval, Pool Interval, Resource Interval) The time (hour and minute) when the data was collected. For the Exception Occurrence Summary and Interval Counts of the Component Report, it is the ending time for the sample interval in which Collection Services recorded the exception.

Job Maximum A-I
(Pool Interval) The highest number of active-state to ineligible-state transitions by a selected job in the pool or subsystem.

Job Maximum A-W
(Pool) The highest number of active-to-wait state transitions by a selected job in the pool or subsystem.

Job Maximum CPU Util
(Pool Interval) The highest percentage of available processing unit time used by a selected job in the pool or subsystem.

Job Maximum Phy I/O
(Pool Interval) The highest number of physical disk input and output operations by a selected job in the pool or subsystem.

Job Maximum Rsp
(Pool Interval) The highest response time in seconds per transaction by a selected job in the pool or subsystem. The response time is the amount of time spent waiting for and using the resources divided by the number of transactions.

Job Maximum Tns
(Pool Interval) The highest number of transactions by a selected job in the pool or subsystem.

Job Maximum W-I
(Pool Interval) The highest number of wait-state to ineligible-state transitions by a selected job in the pool or subsystem.

Job Name
(Component, Transaction, Job Interval, Batch Job Trace) Name of the job. In the Job Summary Report of the Transaction Report, a job (identical job name, user name, and job number) appears multiple times in this list if the job uses the system Reroute Job (RRTJOB) command.

Job Number
(Component, Transaction, Job Interval, Batch Job Trace) The number of the job which the summary line describes. In the Transaction Report, an asterisk (*) before the job number indicates the job signed on during the measurement period. An asterisk (*) after the job number indicates the job signed off during the measurement period.

Job Pty
(Batch Job Trace) Priority of the job.

Job Set
(Transaction) The number of job sets is the number of batch jobs that could be active at any time during the trace period. If two jobs run sequentially, they show up as two jobs in the same job set. If two jobs run concurrently, they show up in two different job sets.

Job Type
(All Reports except where noted for the Transaction Report) Job type and subtype. Possible job type values include the following:

A
Autostart

B
Batch

BD
Batch immediate (Transaction only)

Note: The batch immediate values are shown as BCI on the Work with Active Job display and as BATCHI on the Work with Subsystem Job display.

BE
Batch evoke (Transaction only)

BJ
Batch pre-start job (Transaction only)

C
Programmable workstation application server, which includes 5250 emulation over APPC and IBM i Access host servers running either APPC or TCP/IP. A job is reported as a IBM i Access server if any of the following items are true:

- Incoming APPC evoke requests one of the server program names. This also applies to the pre-started jobs for the QSERVER, QCMN, and QSYSWRK subsystems that are already waiting for the named program.
- Incoming IP port number corresponds to one of the service name-description-port-numbers. This also applies to the pre-started jobs for the QSERVER, QCMN, and QSYSWRK subsystems that are already waiting for the assigned IP port number.
- Incoming IPX socket number corresponds to one of the service name-description-port-numbers. This also applies to the pre-started jobs for the QSERVER, QCMN, and QSYSWRK subsystems that are already waiting for the assigned IPX port number.
- Incoming 5250 display emulation jobs that come from APPC data streams sent by 5250 emulation under OS/2 Communications Manager or WARP equivalent.

D
Target distributed data management (DDM) server

I
Interactive. Interactive includes twinaxial data link control (TDLC), 5250 remote workstation, and 3270 remote workstation. For the Transaction Report, this includes twinaxial data link control (TDLC), 5250 remote workstation, 3270 remote workstation, SNA pass-through, and 5250 Telnet.

L
Licensed Internal Code task

M
Subsystem monitor

P
SNA pass-through and 5250 Telnet pass-through. On the Transaction Report, these jobs appear as I (interactive).

R
Spool reader

S
System

W
Spool writer, which includes the spool write job, and if Advanced Function Printing (AFP) is specified, the print driver job.

WP
Spool print driver (Transaction only)

X
Start system job

Possible job subtype values include the following:

D
Batch immediate job

E
Evoke (communications batch)

J
Pre-start job

P
Print driver job

T
Multiple requester terminal (MRT) (System/36 environment only)

3
System/36

Noninteractive job types include:

- Autostart
- Batch
- Evoke
- IBM i Access-Bch
- Server
- Spool
- Distributed data management (DDM) server

Special interactive job categories include:

- Interactive
- Multiple requester terminal (MRT)
- Pass-through

- System/36

Jobs
(System, Component, Transaction, Pool Interval, Job Interval) The jobs you specify. The format of the entries is `jobnumber/username/jobname`. For the Report Selection Criteria report, it is the list of jobs selected to be included (SLTJOB parameter) or excluded (OMTJOB parameter). This does not include jobs selected by using the STLFCNARA or OMTFCNARA parameter.

K per I/O
(System, Resource Interval) The average number of kilobytes (1024 bytes) read or written for each disk I/O operation.

K/T /Tns Sec
(Transaction) The average delay time, or time spent keying and thinking between transactions for the job, in seconds. The value represents the time interval between active-to-wait and wait-to-active or wait-to-ineligible job state transitions.

KB per I/O Read
(Resource Interval) The average number of kilobytes (1 KB equals 1024 bytes) transferred per read operation.

KB per I/O Write
(Resource Interval) The average number of kilobytes (1024 bytes) transferred per write operation.

KB Received/Second
(System, Component) The total number of kilobytes (1024) received per second on the specified interface when it was active on the selected intervals, which includes framing characters.

KB Transmitted/Second
(System, Component) The total number of kilobytes (1024) transmitted per second from the specified interface when it was active on the selected intervals, which includes framing characters.

KBytes Transmitted IOP
(Component, Resource Interval) Total kilobytes transmitted from an IOP to the system across the bus.

KBytes Transmitted System
(Component, Resource Interval) Total kilobytes transmitted to the IOP from the system across the bus.

Key/Think
(Transaction) The amount of time spent waiting for the work station user by the program.

Key/Think /Tns
(Transaction) The average think time and keying time (or the delay time between transaction boundaries), in seconds, for the interactive jobs.

L
(Lock) Whether this is a lock or seize conflict. The column contains an L if lock, blank if seize.

LAPD Pct Frames Recd in Error
(Resource Interval) The percentage of frames received in error (applies to D-channel only). Errors can occur when the host system has an error or cannot process received data fast enough.

LAPD Pct Frames Trnsmitd Again
(Resource Interval) The percentage of frames retransmitted due to error (applies to D-channel only).

LAPD Total Frames Recd
(Resource Interval) The total number of frames received including frames with errors and frames that are not valid (applies to D-channel only).

LAPD Total Frames Trnsmitd
(Resource Interval) The total number of frames transmitted (applies to D-channel only).

Last 4 Programs in Invocation Stack
(Transaction) The last four programs in the program stack. For example, at the start of a transaction (such as when the work station operator presses the Enter key), you see the program names QT3REQIO, QWSGET, and the program that issued a read operation. At the end of the transaction (such as when the program writes to the display), you see QT3REQIO, QWSPUT, and the program that wrote the display. Usually, the third or fourth program in the stack is the program shown in the

transaction summary PGMNAME data. However, if the *Wait Code* column has a value, the program in the column labeled *Last* is the one that caused the trace record. If there is no program name in a column, the program name was the same as the previous one in the column, and the name is omitted.

Length of Wait
(Lock) The number of milliseconds the requester waited for the locked object.

Lgl I/O /Sec
(Job Interval) The average number of logical disk I/O operations performed per second by the job during the interval. This is calculated from the logical disk I/O count divided by the elapsed time.

Library
(System, Transaction) The library that contains the object.

LIBRARY
(Job Trace) The library name that contains the program associated with the trace entry.

Line Count
(Job Interval) The number of lines printed by the selected noninteractive jobs during the interval.

Line Descriptn
(Resource Interval) Line description name.

Line Errors
(Resource Interval) The total of all detected errors. Check the condition of the line if this value increases greatly over time.

Line Speed
(System, Resource Interval) The line speed in kilobits (1 kilobit = 1000 bits) per second.

Line Type/Line Name
(Component, System) The type and name of the line description that is used by the interface. For interfaces that do not use a line descriptions, the Line Name field will be shown as *LOOPBACK, *OPC, or *VIRTUALIP with no Line Type specified.

Line Util
(Resource Interval) The percent of available line capacity used by transmit and receive operations.

Line Util Trans/Recd
(Resource Interval) The percent used of the data transmission capacity of the communications line. The number of bits transmitted and the number of bits received, during the interval, divided by the line speed.

LKRL
(Transaction) Lock Released. The job released a lock it had on the object named on the next detail line of the report (OBJECT --). The job that was waiting for the object is named on this line (WAITER --) along with the amount of time the job spent waiting for the lock to be released.

LKW
(Transaction) Listed in the Wait Code column, Lock Wait. If there are a number of these, or you see entries with a significant length of time in the ACTIVE/RSP* column, additional analysis is necessary. The LKWT report lines that precede this LKW report line show you what object is being waited on, and who has the object.

LKWT
(Transaction) Listed in the Wait Code column, Lock Conflict Wait. The job is waiting on a lock conflict. The time (*/ time /*) is the duration of the lock conflict and, though not equal to the LKW time, should be very close to it. The holder of the lock is named at the right of the report line (HOLDER --). The object being locked is named on the next report line (OBJECT --).

Local End Code Violation
(Resource Interval) The number of times an unintended code violation was detected by the terminal equipment (TE) for frames received at the interface for the ISDN S/T reference point.

Local Not Ready
(Resource Interval) The percent of all receive-not-ready frames that were transmitted by the host system. A large percentage often means the host cannot process data fast enough (congestion).

Local work station IOP utilization
The fraction of the time interval the work station I/O processors are busy.

Local work station IOPs
(System) The resource name and model number for each local workstation IOP.

Lock Conflict
(Component) Number of lock exceptions per second. Database record contention is reflected in this count. For more information, issue the Start Performance Trace (STRPFRTRC) command and use the Print Transaction Report (PRTTNSRPT) and Print Lock Report (PRTLCKRPT) commands. This count could be very high, even under normal system operation. Use the count as a monitor. If there are large variations or changes, explore these variations in more detail.

Lock Wait /Tns
(Transaction) The average time, in seconds, of the lock-wait time per transaction. If the value is high, investigate with the transaction detail calculation and the Print Lock Report (PRTLCKRPT) command.

Logical
(Job Interval) The number of logical disk I/O operations performed by the selected interactive jobs during the interval.

Logical Database I/O Other
(System) Other logical database operations per transaction. This includes operations such as update and delete.

Logical Database I/O Read
(System) Logical database read operations per transaction.

Logical Database I/O Write
(System) Logical database write operations per transaction.

Logical DB I/O
(System) Average number of logical I/O operations per transaction.

Logical DB I/O Count
(System) Number of times an internal database I/O read, write, or miscellaneous function was called. This does not include I/O operations to readers, writers, or I/O operations caused by the Copy Spooled File (CPYSPLF) command or the Display Spooled File (DSPSPLF) command. If you specify SEQONLY(*YES), you see numbers that show each block of records read or written, not the number of individual records read or written. Miscellaneous functions include the following: updates, deletes, force-end-of-data, and releases.

Logical Disk I/O
(Component) Number of logical disk operations (Get, Put, Update, Other).

Logical I/O /Second
(System) Average number of logical disk I/O operations per second.

Logical I/O Per Second
(Job Interval) The average number of logical disk I/O operations performed per second by the selected noninteractive jobs during the interval.

Long Wait
(Transaction) The time the job spent waiting for a system resource. An example of a long wait would be a record-lock conflict. Also listed in the Elapsed Time--Seconds column, it is the elapsed time in the state (such as waiting for the next transaction or lock-wait time).

Long Wait Lck/Oth
(Transaction) The amount of time the job spent waiting for a system resource. An example of a long wait would be a record-lock conflict.

Loss of Frame Alignment
(Resource Interval) The number of times a time period equivalent to two 48-bit frames elapsed without detecting valid pairs of line code violations.

MAC Errors
(Resource Interval) The number of medium access control (MAC) errors.

Main storage (MB)
(System) The total main storage size, as measured in megabytes. These codes are in the wait code column, but they are not wait codes. They indicate transaction boundary trace records.

Max Util
(System) Consistent use at or above the threshold value given will affect system performance and cause longer response times or less throughput.

Maximum
(Transaction) The maximum value of the item that occurred in the column.

Member
(System, Transaction) For the System Report, this is the name of the performance data member that was specified on the TOMBR parameter of the Create Performance Data (CRTPFRDTA) command. For the Transaction Report, the member that was involved in the conflict.

Minimum
(Transaction) The minimum value of the item that occurred in the column.

MRT Max Time
(System) The time spent waiting, after MRTMAX is reached, by jobs routed to a multiple requester terminal.

Note: No value appears in this column if job type is not MRT.

MSGS
(Job Trace) The number of messages sent to the job during each transaction.

MTU size (bytes)
(System) The size of the largest datagram that can be sent or received on the interface. The size is specified in octets (bytes). For interfaces that are used for transmitting network datagrams, this is the size of the largest network datagram that can be sent on the interface.

Nbr A-I
(Transaction) The number of active-to-ineligible state transitions by the job. This column shows the number of times that the job exceeded the time-slice value assigned to the job, and had to wait for an activity-level slot before the system could begin processing the transaction. If a value appears in this column, check the work that the job was doing, and determine if changes to the time-slice value are necessary.

Nbr Disk Units
(System) The number of disk units assigned to the reported partition.

Nbr Evt
(Transaction) The number of event waits that occurred during the job processing.

Nbr Jobs
(Transaction) The number of jobs.

Nbr Sign offs
(Transaction) The number of jobs that signed off during the interval.

Nbr Sign ons
(Transaction) The number of jobs that signed on during the interval.

Nbr Tns
(Transaction) The number of transactions in a given category.

Note: The values for transaction counts and other transaction-related information shown on the reports you produce using the Print Transaction Report (PRTTNSRPT) command may vary from the values shown on the reports you produce using the Print System Report (PRTSYSRPT) and Print Component Report (PRTCPTRPT) commands. These differences are caused because the PRTTNSRPT command uses trace data as input, while the PRTSYSRPT and PRTCPTRPT commands use sample data as input.

If there are significant differences in the values for transaction-related information shown on these reports, do not use the data until you investigate why these differences exist.

Nbr W-I
(Transaction) The number of wait-to-ineligible state transitions by the job. This column shows how many times the job had to wait for a transaction.

NDB Read
(Transaction) Listed in Physical I/O Counts column, it is the number of nondatabase read requests while the job was in that state. Listed in the Sync Disk I/O Rqs/Tns column, it is the average number of synchronous nondatabase read requests per transaction.

NDB Write
(Transaction) Listed in the Sync Disk I/O Rqs/Tns column, it is the average number of synchronous nondatabase write requests per transaction.

NDB Wrt
(Transaction) Listed in Physical I/O Counts column, the number of nondatabase write requests while the job was in that state. Listed under Synchronous Disk I/O Counts column, it is the number of synchronous nondatabase write requests per transaction.

NON-DB
(Job Trace) The number of physical nondatabase reads that occurred for the entry.

Non-DB Fault
(System, Component) Average number of nondatabase faults per second.

Non-DB Pages
(System, Component) Average number of nondatabase pages read per second.

NON-DB RDS
(Job Trace) The number of physical nondatabase reads that occurred.

Non SMAPP
(Component) Journal deposits not directly related to SMAPP (System Managed Access Path Protection).

Non-SSL Inbound Connect
(System) The number of non-SSL inbound connections accepted by the server.

Non-Unicast Packets Received
(System) The total number of non-unicast packets delivered to a higher-layer protocol for packets received on the specified interface.

Non-Unicast Packets Sent
(System) The total number of packets that higher-level protocols requested to be transmitted to a non-unicast address; therefore, this number includes those packets that were discarded or were not sent as well as those packets that were sent.

Number
(Transaction) The number of the job with which the transaction is associated.

Number I/Os per Second
(System) The number of I/Os per second for this particular IOP.

Number Jobs
(Transaction) The number of batch jobs in the job set.

Number Lck Cft
(Transaction) The number of lock-wait (including database record lock) state conflicts that occurred during the job processing. If this number is high, look at the Transaction and Transition Reports for the job to see how long the lock-wait state conflicts were lasting. In addition, you can do further investigation using the reports produced when you use the Print Lock Report (PRTLCKRPT) command.

Number Lck Conflict
(Transaction) The number of times the job had a lock conflict.

Number Locks
(Transaction) The number of locks attributed to interactive or noninteractive waiters.

Number of batch jobs
(System) The average number of active batch jobs. A batch job is considered active if it averages at least one I/O per 5 minutes.

Number of Jobs
(System) Number of jobs.

Number of Packets Received with Errors
(System) The total number of packets that were received with errors or discarded for other reasons. For example, a packet could be discarded to free up buffer space.

Number Seizes
(Transaction) The number of seizes attributed to interactive or noninteractive waiters.

Number Sze Cft
(Transaction) The number of seize/lock conflicts that occurred during the job processing. If this number is high, look at the Transaction and Transition Reports for the job to see how long the conflicts lasted, the qualified name of the job that held the object, the name and type of object being held, and what the job was waiting for.

Number Sze Conflict
(Transaction) The number of times the job had a seize conflict.

Number Tns
(System, Transaction) Total number of transactions processed. For example, in the System Report it is the total number of transactions processed by jobs in this pool. In the Transaction Report it is the number of transactions associated with the program.

Number Traces
(Batch Job Trace) Number of traces.

Number Transactions
(System) Total number of transactions processed.

Object File
(Transaction) The file that contains the object.

Object Library
(Transaction) The library that contains the object.

Object Member
(Transaction) The member that was involved in the conflict.

Object Name
(Lock) The name of the locked object.

Object RRN
(Transaction) The relative record number of the record involved in the conflict.

Object Type
(Transaction, Lock) The type of the locked object. The following are possible object types:

AG
Access group

CB
Commit block

CBLK
Commit block

CD
Controller description

CLS
Class

CMD
Command

CTLD
Controller description

CTX
Context

CUD
Control unit description

CUR
Cursor

DEVD
Device description

DS
Data space

DSI
Data space index

DTAARA
Data area

EDTD
Edit description

FILE
File

JOBD
Job description

JOBQ
Job queue

JP
Journal port

JRN
Journal

JRNRCV
Journal receiver

JS
Journal space

LIB
Library

LIND
Line description

LUD
Logical unit description

MBR
Member

MEM
Database file member

MSGF
Message file

MSGQ
Message queue

ND
Network description

OCUR
Database operational cursor

OUTQ
Output queue

PGM
Program

PROG
Program

PRTIMG
Print image

QDAG
Composite piece - access group

QDDS
Composite piece - data space

QDDSI
Composite piece - data space index

QTAG
Temporary - access group

QTDS
Temporary - data space

QTDSI
Temporary - data space index

SBSD
Subsystem description

TBL
Table

Omit Parameters
(System, Component, Transaction, Job Interval, Pool Interval) The criteria used to choose the data records to be excluded from the report. The criteria are generally specified using an OMTxxx parameter of the command. Only nondefault values (something other than *NONE) are printed. If a parameter was not specified, it does not appear on the report.

Op per Second
(System) Average number of disk operations per second.

Other Wait /Tns
(Transaction) The average time, in seconds, spent waiting that was not in any of the previous categories per transaction. For example, the time spent waiting during a save/restore operation when the system requested new media (tape or diskette).

Outgoing Calls Pct Retry
(Resource Interval) The percentage of outgoing calls that were rejected by the network.

Outgoing Calls Total
(Resource Interval) The total number of outgoing call attempts.

Over commitment ratio
(System) The main storage over commitment ratio (OCR).

PAG
(Transaction) The number of process access group faults.

PAG Fault
(Component, Job Interval) In the Exception Occurrence Summary of the Component Report, it is the total number of times the program access group (PAG) was referred to, but was not in main storage. The Licensed Internal Code no longer uses process access groups for caching data. Because of this implementation, the value will always be 0 for more current releases. In the Exception Occurrence Summary of the Component Report, it is the number of faults involving the process access group per second.

Page Count
(Job Interval) The number of pages printed by the selected noninteractive jobs during the interval.

Pct CPU By Categories
(Transaction) The percentage of available processing unit time used by the transactions that fell into the various categories. See the ANALYSIS by Interactive Transaction Categories part of the System Summary Data Section for an explanation of the categories.

Pct Data Characters Received in Error
(Resource Interval) The percent of data characters received with error.

Pct Data Characters Transmitted in Error
(Resource Interval) The percent of data characters transmitted with error.

Pct Datagrams Error
(Component) The percentage of datagrams that were discarded due to these errors:

- The IP address in the destination field of the IP header was not a valid address to be received at this entity.
- The protocol was unknown or unsupported.
- Not enough buffer space.

Pct Error Responses
(Component) Percentage of responses in error.

Pct Ex-Wt /Rsp
(Transaction) The percentage of the response time that is due to exceptional wait.

Pct ICMP Messages Error
(Component) This is the number of Internet Control Message Protocol (ICMP) messages that the entity received but determined that the messages had errors or are messages that the entity did not send due to problems.

Pct Of Tns Categories
(Transaction) The percentage of all transactions that fell into the various categories. See the Analysis by Interactive Transaction Categories part of the System Summary Data Section for an explanation of the categories.

Pct Packets Received Error
(System) The percentage of packets that were received with errors or discarded for other reasons. For example, a packet could be discarded to free up buffer space.

Pct Packets Sent Error
(System) The percentage of packets that were not sent because of errors or discarded for other reasons. For example, a packet could be discarded to free up buffer space.

Pct PDUs Received in Error
(Resource Interval) The percent of protocol data units (PDUs) received in error during the time interval. These errors can occur if the host system has errors or cannot receive data fast enough (congestion).

Note: A protocol data unit (PDU) for asynchronous communications is a variable-length unit of data that is ended by a protocol control character or by the size of the buffer.

Pct Poll Retry Time
(Resource Interval) The percent of the time interval the line was unavailable while the IOP waited for a response from a work station controller (or remote system) that was in disconnect mode.

Note: To minimize this lost time:

- Vary on only the controllers that are turned on.
- Turn on all controllers.
- Use the Change Line Description (SDLC) (CHGLINSDLC) command to set the connect poll timer to a small value (reduces wait time).
- Use the Change Controller Description (CHGCTLxxxx) command (where xxxx is APPC, FNC, RWS, or RTL, as appropriate) to set the NDMPOLLTMR value to a large value (increases time between polls).

Pct Tns
(Transaction) The percentage of the total transactions. For the System Summary section of the Job Summary Report, the transactions are within the given trace period with the given purge attribute. For the Interactive Program Transaction Statistics section of the Job Summary Report, the percentage of transactions that were associated with a program. For the Job Statistics section, it is the percentage of total transactions that were due to this job. For the Interactive Program Statistics section, it is all transactions that were associated to a program.

Pct UDP Datagrams Error
(Component) The percentage of User Datagram Protocol (UDP) datagrams for which there was no application at the destination port or that could not be delivered for other reasons.

Percent Errored Seconds
(Resource Interval) The percentage of seconds in which at least one Detected Access Transmission (DTSE) in or out error occurred.

Percent Frames Received in Error
(Resource Interval) The percent of all received frames that were received in error. Errors can occur when the host system has an error or cannot process received data fast enough (congestion).

Percent Full
(System) Percentage of disk space capacity in use.

Percent I Frames Trnsmitd in Error
(Resource Interval) The percent of transmitted information frames that required retransmission. Retransmissions can occur when a remote device has an error or cannot process received data fast enough (congestion).

Percent Severely Errored Seconds
(Resource Interval) The percent of seconds in which at least three Detected Access Transmission (DTSE) in or out errors occurred.

Percent transactions (dynamic no)
(System) A measure of system main storage utilization. The percent of all interactive transactions that were done with the purge attribute of dynamic NO.

Percent transactions (purge no)
(System) A measure of system main storage utilization. The percent of all interactive transactions that were done with the purge attribute of NO.

Percent transactions (purge yes)
(System) A measure of system main storage utilization. The percent of all interactive transactions that were done with the purge attribute of YES.

Percent Util
(System) Average disk arm utilization (busy). Consistent use at or above the threshold value provided for disk arm utilization affects system performance, which causes longer response times or less throughput.

Note: The percent busy value is calculated from data measured in the I/O processor. When comparing this value with percent busy reported by the Work with Disk Status (WRKDSKSTS) command, some differences may exist. The WRKDSKSTS command estimates percent busy based on the number of I/O requests, amount of data transferred, and type of disk unit.

The system-wide average utilization does not include data for mirrored arms in measurement intervals for which such intervals are either in resuming or suspended status.

Perm Size
(Component) Kilobytes placed within the permanent area; these are traditional journal entries which can be retrieved and displayed.

Perm Write
(Component, Job Interval) The number of permanent write operations performed for the selected jobs during the interval.

Permanent writes per transaction
(System) The average number of permanent write operations per interactive transaction.

Physical I/O Count
(Transaction, Batch Job Trace) For the Job Summary section of the Batch Job Trace Report, the number of synchronous and asynchronous disk operations (reads and writes). For the Transition Report, the next five columns provide information about the number of synchronous and asynchronous disk I/O requests while the job was in the given state. The first line is the synchronous disk I/O requests, and the second line is the asynchronous disk I/O requests.

DB Read
The number of database read requests while the job was in that state.

DB Wrt
The number of database write requests while the job was in that state.

NDB Read
The number of nondatabase read requests while the job was in that state.

NDB Wrt
The number of nondatabase write requests while the job was in that state.

Tot
The total number of DB Read, DB Wrt, NDB Read, and NDB Wrt requests.

Physical Writes
(Component) Physical journal write operations to disk.

Pl
(Component, Transaction, Job Interval, Pool Interval) The number of the pool in which the subsystem or job ran.

Pool
(Transaction, Job Interval, Batch Job Trace) The number of the pool containing the transaction (for example, in which the job ran.)

Pool ID
(System) Pool identifier.

Pool ID Faults
(Component) User pool that had the highest page fault rate.

Pool Mch Faults/Sec
(Component) Average number of machine pool page faults per second.

Pool size (MB)
(Component) For the Storage Pool Activity section of the Component Report it is the initial pool size in megabytes.

Pool User Faults/Sec
(Component) Average number of user pool page faults per second, for the user pool with highest fault rate during this interval.

Pools
(System, Component, Transaction, Job Interval, Pool Interval) In the Report-Selection Criteria section, the list of pools selected to be included (SLTPOOLS parameter) or excluded (OMTPOOLS parameter). Otherwise, the pools you specify. The values can be from 1 through 64.

Prg
(Transaction) The purge attribute of the jobs.

Printer Lines
(System, Job Interval) The number of lines printed by the job during the interval.

Printer Pages
(System, Job Interval) The number of pages printed by the job during the interval.

Priority
(System, Transaction) The priority of the job.

Program
(Transaction) The name of the program with which the transaction is associated.

PROGRAM
(Job Trace) The name of the program for the entry.

PROGRAM CALL
(Job Trace) The number of non-QSYS library programs called during the step. This is not the number of times that the program named in the PROGRAM NAME field was called.

PROGRAM DATABASE I/O
(Job Trace) The number of times the IBM-supplied database modules were used during the transaction. The database module names have had the QDB prefix removed (PUT instead of QDBPUT). The type of logical I/O operation performed by each is as follows:

GETDR
Get direct

GETSQ
Get sequential

GETKY
Get by key

GETM
Get multiple

PUT, PUTM
Add a record

UDR
Update, delete, or release a record

PROGRAM INIT
(Job Trace) The number of times that the IBM-supplied initialization program was called during the transaction. For RPG programs this is QRGXINIT, for COBOL it is QCRMAIN. Each time the user program ends with LR (RPG) or END (COBOL), the IBM-supplied program is also called. This is not the number of times the program named in the PROGRAM NAME field was initialized. QCRMAIN is used for functions other than program initialization (for example, blocked record I/O, some data conversions).

Program Name
(Transaction) For the Job Summary section of the Transaction Report, the name of the program in control at the start of the transaction. Other programs may be used during the transaction. For the Transaction Report section, the name of the program active at the start of the transaction. If ADR=UNKNWN (address unknown) is shown under the column, the program was deleted before the trace data was dumped to the database file. If ADR=000000 is shown under the column, there was not enough trace data to determine the program name, or there was no program active at that level in the job when the trace record was created.

PROGRAM NAME
(Job Trace) The name of the last program called that was not in the library QSYS before the end of a transaction.

Protocol
(System) Line protocol.

- SDLC
- ASYNC
- BSC
- X25
- TRLAN
- ELAN (Ethernet)
- IDLC
- DDI
- FRLY

- PPP

Pty
(Component, Transaction, Job Interval) Priority of the job. For the Concurrent Batch Job Statistics section of the Transaction Report, it is the priority of the jobs in the job set.

PU
(System) Partition capacity. The number of processor units assigned to the reported partition.

Purge
(Transaction) The purge attribute of the jobs.

PWrt
(Transaction) The number of permanent write I/O operations.

Queue Length
(Resource Interval) The average number of I/O requests that had to wait in the queue for this unit.

Rank
(Transaction) The order. For the Job Summary section, it is the order of the program according to the number of transactions. For the Job Statistics section, it is the order of the job. For the Interactive Program Statistics section, it is the order of the program. For the Individual Transaction Statistics section, it is the order of the transaction according to the data being put in order by importance. For the Largest Seize/Lock Conflicts section, it is the order of the seize or lock conflict.

Ratio of write disk I/O to total disk I/O
(System) The fraction of the total disk activity that is due to writing data to the disks.

Reads per Second
(Resource Interval) The average number of disk read operations performed per second by the disk arm.

Receive CRC Errors
(Resource Interval) The number of received frames that contained a cycle redundancy check (CRC) error. This indicates that the data was not received error free.

Record Number
(Lock) For database file members, the relative record number of the record within the database file member.

Remote LAN Pct Frames Recd
(Resource Interval) The number of frames received from a local area network (LAN) connected to the locally attached LAN.

Remote LAN Pct Frames Trnsmitd
(Resource Interval) The number of frames transmitted to a local area network (LAN) connected to the locally attached LAN.

Remote Not Ready
(Resource Interval) The percentage of all receive-not-ready frames that were received by the host system. A large percentage often means the remote device cannot process data fast enough (congestion).

Remote Seq Error
(Resource Interval) The percent of frames received out of order by a remote device or system. This can occur when the remote device or system cannot process data fast enough.

Req type
(Component) The type of request being reported.

Requests received
(System, Component) The number of requests of all types received by the server.

Requestor's Job Name
(Lock) The name of the job requesting the locked object (the same as in the detail listing).

Reset Packets Recd
(Resource Interval) The number of reset packets received by the network. **Reset packets** are packets retransmitted because an error occurred.

Reset Packets Trnsmitd
(Resource Interval) The number of reset packets transmitted by the network.

Response
(System) Average system response (service) time.

Response Sec Avg and Max
(Transaction) The average (AVG) and maximum (MAX) transaction response time, in seconds, for the job. The average response time is calculated as the sum of the time between each pair of wait-to-active and active-to-wait transitions divided by the number of pairs that were encountered for the job. The MAX response time is the largest response time in the job.

Response Seconds
(System) Average response time in seconds per transaction.

Responses sent
(System, Component) The number of responses of all types sent by the server.

Rsp
(Component) Average interactive transaction response time in seconds.

Rsp Time
(Component, Resource Interval) The average external response time (in seconds). For the Local Work Station IOP Utilizations section of the Resource Interval Report, it is the response time for work stations on this controller. For the Remote Work Stations section of the Component Report, it is the response time for this work station.

Rsp Timer Ended
(Resource Interval) The number of times the response timer ended waiting for a response from a remote device.

Rsp/Tns
(Component, Transaction, Job Interval) The average response time (seconds) per transaction. For the Job Summary section of the Job Interval Report, it is the response time per transaction for the selected interactive jobs during the interval (the amount of time spent waiting for or using the system resources divided by the number of transactions processed). This number will not be accurate unless at least several seconds were spent processing transactions.

S/L
(Transaction) Whether the conflict was a seize (S) or lock (L) conflict.

SECONDS
(Job Trace) The approximate time the job was waiting or active.

Segments Pct Rtrns
(Component) The percentage of segments retransmitted. This number is the TCP segments that were transmitted and that contain one or more previously transmitted octets (bytes).

Segments Rcvd per Second
(Component) The number of segments received per second. This number includes those received in error and those received on currently established connections.

Segments Sent per Second
(Component) The number of segments sent per second. This number includes those sent on currently established connections and excludes those that contain only retransmitted octets (bytes).

Seize and Lock Conflicts
(Batch Job Trace) Number of seize conflicts and lock waits.

Seize Conflict
(Component) Number of seize exceptions per second. For more detailed information, issue the Start Performance Trace (STRPFRTRC) command, and use the PRTTNSRPT or PRTLCKRPT commands. This count could be very high, even under normal system operation. Use the count as a monitor. If there are large variations or changes, explore these variations in more detail.

Seize Hold Time
(Transaction) The amount of time that the transaction held up other jobs in the system by a seize or lock on an object.

Seize Wait /Tns
(Transaction) The average time, in seconds, for all seize-lock conflicts that occur during an average transaction. More than one seize-lock conflict can occur during a single transaction for the same job. If this number is high, investigate those jobs with seize conflicts. The Transaction Report lists each conflict that occurs, the name of the holder, and the name of the object held. For the Transaction by 5-Minute Intervals section of the Job Summary Report, it is the average seize wait time per transaction in seconds. This is the average amount of time that the transactions spent in a seize/lock conflict. If this number is high, look at the Transaction and Transition Reports for the jobs that are causing the excessive wait time.

Select Parameters
(System, Component, Transaction, Job Interval, Pool Interval) The criteria used to choose the data records to be included in the report. The criteria are generally specified using an SLTxxx parameter of the command. Only nondefault values (something other than *ALL) are printed. If a parameter is not specified, it does not appear on the report.

SEQNBR
(Job Trace) The number of the trace entry.

SEQNCE or SEQUENCE
(Job Trace) The job trace sequence number in the detail report that this summary line refers to.

Sequence Error
(Resource Interval) The number of frames received that contained sequence numbers indicating that frames were lost.

Server job name
(System) The server job number. Identifies the child job for the server.

Server job user
(System) The server job user. Identifies the child job for the server.

Server name
(System) The server job name. Identifies the child job for the server.

Server start date/time
(System) The most recent start or restart time in format mm/dd/yy hh:mm:ss

Short Frame Errors
(Resource Interval) The number of short frames received. A short frame is a frame that has fewer octets between its start flag and end flag than are permitted.

Short Wait /Tns
(Transaction) The average time, in seconds, of short (active) wait time per transaction. For the Interactive Program Statistics section, if the value is high, it may be due to the use of data queues or to the use of DFRWRT(*NO) or RSTDSP(*YES) in the program display files.

Short WaitX /Tns (Short wait extended)
(Transaction) The average time, in seconds, of wait time per transaction that resulted due to a short (active) wait that exceeded 2 seconds, and caused a long wait transition to occur. The activity level has been released but this time is still counted against your total response time. Waits on data queues or the use of DFRWRT(*NO) and/or RSTDSP(*YES) in the display files could be reasons for this value to be high.

Size
(Component) Decimal data overflow and underflow exceptions per second. An indication of improper field size on numeric calculations.

Size (MB)
(System) The size of the pool in megabytes.

Size (GB)
(Pool Interval) The size of the pool in gigabytes.

Size (M)
(System) Disk space capacity in millions of bytes.

SHARE CLS
(Job Trace) The number of shared closes for all types of files.

SHARE OPN
(Job Trace) The number of shared opens for all types of files.

SMAPP ReTune
(Component) System-managed access path protection tuning adjustments.

SMAPP System
(Component) SMAPP-induced journal entries deposited in system-provided (default) journals.

SMAPP User
(Component) SMAPP-induced journal entries deposited in user-provided journals.

SOTn
(Transaction) Listed in the Wait Code column, Start of transaction n. These codes are in the wait code column, but they are not wait codes. They indicate transaction boundary trace records.

Spool CPU seconds per I/O
(System) The average number of system processing unit seconds used by all spool jobs for each I/O performed by a spool job.

Spool database reads per second
(System) The average number of read operations to database files per second of spool processing.

Spool I/O per second
(System) The average number of physical disk I/O operations per second of spool processing.

SQL CPU Util
Percentage of available CPU time used to perform work done on behalf of SQL operations. This field applies to all systems running IBM i 7.2 or later.

Srv Time
(Component) Average disk service time per request in seconds not including the disk wait time.

SSL Inbound Connections
(System) AThe number of SSL inbound connections accepted by the server.

Start
(Transaction) The time the job started.

Started
(Transaction) The time of the first record in the trace data, in the form HH.MM.SS (hours, minutes, seconds).

State
(Transaction) The three possible job states are:

- **W**--(Wait state) not holding an activity level.
- **A**--(Active or wait state) holding an activity level.
- **I**--(Ineligible state) waiting for an activity level.

The table below shows the possible job state transitions. For example, from **W** to **A** is **yes**, which means it is possible for a job to change from the *wait* state to the *active* state.

Table 8.

	To state			
		A	**W**	**I**
From state	**A**	yes	yes	yes
	W	yes		yes
	I	yes		

State Transitions A-A
(Batch Job Trace) Number of active-to-active transitions.

State Transitions A-I
(Batch Job Trace) Number of active-to-ineligible transitions.

Stop
(Transaction) The time the job ended.

Stopped
(Transaction) The time of the last record in the trace data, in the form HH.MM.SS (hours, minutes, seconds).

SUBFILE READS
(Job Trace) The number of subfile reads.

SUBFILE WRITES
(Job Trace) The number of subfile writes.

Subsystem Name
(Pool Interval) The name of the subsystem.

Subsystems
(System, Component, Pool Interval) For the System Report, the subsystem names you specify. Each name is a 10-character name. For the Component Report, the list of subsystems selected to be included (SLTSBS parameter) or excluded (OMTSBS parameter).

Sum
(Transaction) Listed in the Sync Disk I/O Rqs/Tns column, the sum of the averages of the synchronous DB READ, DB WRITE, NDB READ, and NDB WRITE requests (the average number of synchronous I/O requests per transaction for the job).

SWX
(Transaction) Listed in the Wait Code column, Short Wait Extended. The short wait has exceeded a 2-second limit and the system has put the transaction into a long wait. This long wait must be charged to the transaction response time. In most cases, this active-to-wait transaction does not reflect a transaction boundary.

Sync
(Job Interval) The number of synchronous disk I/O operations performed by the selected interactive jobs during the interval.

Sync DIO /Tns
(Transaction) The average number of synchronous I/O requests per transaction during the interval.

Sync Disk I/O
(System, Component, Transaction) Synchronous disk I/O operations.

Sync Disk I/O per Second
(Component) Average synchronous disk I/O operations per second.

Sync Disk I/O Requests
(Transaction) The total number of synchronous disk I/O requests for the given combination of priority, job type, and pool.

Sync Disk I/O Rqs/Tns
(Transaction) The next five columns provide information about the number of synchronous disk I/O requests per transaction:

DB Read
The average number of synchronous database read requests per transaction.

DB Write
The average number of synchronous database write requests per transaction.

NDB Read
The average number of synchronous nondatabase read requests per transaction.

NDB Write
The average number of synchronous nondatabase write requests per transaction.

Sum
The sum of the averages of the synchronous DB READ, DB WRITE, NDB READ, and NDB WRITE requests (the average number of synchronous I/O requests per transaction for the job).

Sync I/O /Elp Sec
(Transaction) The average number of synchronous disk I/O requests for all jobs, per second of elapsed time used by the jobs.

Sync I/O /Sec
(Job Interval) The average number of synchronous disk I/O operations performed per second by the job during the interval. This is calculated from the synchronous disk I/O count divided by the elapsed time.

Sync I/O Per Second
(Job Interval) The average number of synchronous disk I/O operations performed per second by the selected noninteractive jobs during the interval.

Synchronous DBR
(System, Transaction, Job Interval, Pool Interval) The average number of synchronous database read operations. It is the total synchronous database reads divided by the total transactions. For the Pool Interval and Job Interval Reports, it is calculated per transaction for the job during the intervals. For the System Report, it is calculated per second. For the Transaction (Job Summary) it is calculated per transaction. Listed under Average DIO/Transaction, the average number of synchronous database read requests per transaction. This field is not printed if the jobs in the system did not process any transactions.

Synchronous DBW
(System, Transaction, Job Interval, Pool Interval) The average number of synchronous database write operations. It is the total synchronous database writes divided by the total transactions. For the Pool Interval and Job Interval Reports, it is calculated per transaction for the job during the intervals. For the System Report, it is calculated per second. For the Transaction (Job Summary) it is calculated per transaction. Listed under Average DIO/Transaction, the average number of synchronous database read requests per transaction. This field is not printed if the jobs in the system did not process any transactions.

Synchronous DIO / Act Sec
(System, Transaction) The number of synchronous disk I/O operations per active second. The active time is the elapsed time minus the wait times.

Synchronous DIO / Ded Sec
(Transaction) The estimated number of synchronous disk I/O operations per second as if the job were running in dedicated mode. Dedicated mode means that no other job would be active or in contention for resources in the system.

Synchronous DIO / Elp Sec
(Transaction) The number of synchronous disk I/O operations per elapsed second.

Synchronous Disk I/O Counts
(Transaction) The next five columns provide information about the number of synchronous disk I/O requests per transaction:

DB Read
The number of synchronous database read requests per transaction.

DB Wrt
The number of synchronous database write requests per transaction.

NDB Read
The number of synchronous nondatabase read requests per transaction.

NDB Wrt
The number of synchronous nondatabase write requests per transaction.

Sum
The sum of the synchronous DB Read, DB Wrt, NDB Read, and NDB Wrt requests (the number of synchronous I/O requests per transaction).

Synchronous disk I/O per transaction
(System, Transaction) The average number of synchronous physical disk I/O operations per interactive transaction.

Synchronous Max
(Transaction) The maximum number of synchronous DBR, NDBR, and WRT I/O requests encountered for any single transaction by that job. If the job is not an interactive or autostart job type, the total disk I/O for the job is listed here.

Synchronous NDBR
(System, Transaction, Job Interval, Pool Interval) The average number of synchronous nondatabase read operations per transaction for the jobs in the system during the interval. For the Transaction Report, the operations on the disk per transaction for the selected jobs in the pool. This is calculated from the synchronous nondatabase read count divided by the transactions processed. This field is not printed if the jobs in the system did not process any transactions.

Synchronous NDBW
(System, Job Interval, Pool Interval) The average number of synchronous nondatabase write operations on the disk per transaction for the selected jobs in the pool. For the System Report, it is the operations per transaction for the jobs in the system during the interval. This is calculated from the synchronous nondatabase write count divided by the transactions processed. This field is not printed if the jobs in the system did not process any transactions.

Synchronous Sum
(Transaction) The sum of the averages of the synchronous DBR, NDBR, and WRT requests (the average number of synchronous I/O requests per transaction for the job).

Synchronous wrt
(Transaction) The average number of synchronous database and nondatabase write requests per transaction.

System CPU per transaction (seconds)
(System) The average number of system processing unit seconds per interactive transaction.

System disk I/O per transaction
(System) The total number of physical disk I/O operations attributed to the system per interactive transaction.

System Starts
(Component) The number of start journal operations initiated by the system.

System Stops
(Component) The number of stop journal operations initiated by the system.

System Total
(Component) The total number of journal deposits resulting from system-journaled objects. These are the deposits performed by system-managed access path protection (SMAPP).

System ToUser
(Component) The number of journal deposits resulting from system-journaled objects to user-created journals.

SZWG
(Transaction) Listed in the Wait Code column, Seize Wait Granted. The job was waiting on a seize conflict. The original holder released the lock that it had on the object, and the lock was then granted to the waiting job. The job that was waiting for the object is named on this line (WAITER --) along with the amount of time the job spent waiting for the seize conflict to be released. The object that is held is named on the next line of the report (OBJECT --).

SZWT
(Transaction) Listed in the Wait Code column, Seize/Lock Conflict Wait. The job is waiting on a seize/lock conflict. The time (*/ time /*) is the duration of the seize/lock conflict, and is included in the active time that follows it on the report. The holder of the lock is named at the right of the report line (HOLDER --). The object being held is named on the next report line (OBJECT --).

Teraspace EAO
(Component) Listed in the Exception Occurrence summary and Interval Counts. A teraspace effective address overflow (EAO) occurs when computing a teraspace address that crosses a 16-boundary. A quick estimate indicates that a 1% performance degradation would occur if there were 2,300 EAOs per second.

Thread
(Job Summary, Transaction, Transition) A thread is a unique flow of control within a process. Every job has an initial thread associated with it. Each job can start one or more secondary threads. The system assigns the thread number to a job as follows:

- The system assigns thread IDs sequentially. When a job is started that uses a job structure that was previously active, the thread ID that is assigned to the initial thread is the next number in the sequence.
- The first thread of a job is assigned a number.
- Any additional threads from the same job are assigned a number that is incremented by 1. For example:

```
Job Name   User Name/  Job Number
            Thread
QJVACMDSRV   SMITH       023416
QJVACMDSRV    00000006   023416
QJVACMDSRV    00000007   023416
QJVACMDSRV    00000008   023416
```

A thread value greater than 1 does not necessarily mean the job has had that many threads active at the same time. To determine how many threads are currently active for the same job, use the WRKACTJOB, WRKSBSJOB, or WRKUSRJOB commands to find the multiple three-part identifiers with the same job name.

Threads active
(System) The number of threads doing work when the data was sampled.

Threads idle
(System) The number of idle threads when the data was sampled.

Time
(Transaction) The time when the transaction completed, or when a seize or lock conflict occurred. Also, a column heading that shows the time the transition from one state to another occurred, in the HH.MM.SS.mmm arrangement.

TIME
(Job Trace) The time of day for the trace entry. The time is sequentially given in hours, minutes, seconds, and microseconds.

Tns
(Component, Pool Interval) The total number of transactions processed by the selected jobs in the pool or subsystem.

Tns Count
(Component, Job Interval) The number of transactions performed by the selected interactive jobs during the interval.

Tns/Hour
(Component, Transaction, Job Interval) The average number of transactions per hour processed by the selected interactive jobs during the interval.

Tns/Hour Rate
(System) Average number of transactions per hour.

TOD of Wait
(Lock) The time of day of the start of the conflict.

Tot
(Transaction) Listed in Physical I/O Counts column, the total number of DB Read, DB Wrt, NDB Read, and NDB Wrt requests.

Tot Nbr Tns
(Transaction) The total number of transactions the PRTTNSRPT program determined from the input data that were accomplished for the job.

Total
(Component) Total exception counts for the reporting period.

TOTAL
(Job Trace) Totals for the fields.

Total /Job
(Transaction) The total (sum) of the items in the column for the job.

Total characters per transaction
(System) The average number of characters either read from or written to display station screens per interactive transaction.

Total CPU Sec /Sync DIO
(Transaction) The ratio of total CPU seconds divided by the total synchronous disk I/O requests.

Total CPU Utilization
(System, Component) Percentage of available processing unit time used by the partition. For a multiple-processor system, this is the average use across all processors. For dedicated partitions, *Total CPU Utilization* is replaced by a utilization value for each processor in the partition. Here is an example of this part of the display for a dedicated partition with two processors:

```
Average CPU utilization  . . . . . :    41.9
CPU 1 utilization  . . . . . . . . :    41.7
CPU 2 utilization  . . . . . . . . :    42.2
```

In shared processor partitions, individual CPU utilization rows are not printed.

Note: This value is taken from a system counter. Other processing unit uses are taken from the individual job work control blocks (WCBs). These totals may differ slightly. For uncapped partitions, *Total CPU utilization* might exceed 100 percent.

Total CPU Utilization (Interactive Feature)
(System) The CPU Utilization (Interactive Feature) shows the CPU utilization for all jobs doing 5250 workstation I/O operations relative to the capacity of the system for interactive work. Depending on the system and associated features purchased, the interactive capacity is equal to or less than the total capacity of the system.

Total CPU Utilization (SQL)
(System) Shows you the SQL activity on your systems. This field applies to all systems running IBM i 7.2 or later.

Total Data Characters Received
(Resource Interval) The number of data characters received successfully.

Total Data Characters Transmitted
(Resource Interval) The number of data characters transmitted successfully.

Total Datagrams Requested for Transmission
(Component) The percentage of IP datagrams that are discarded because of the following reasons:

- No route was found to transmit the datagrams to their destination.
- Lack of buffer space.

Total fields per transaction
(System) The average number of display station fields either read from or written to per interactive transaction.

Total Frames Recd
(Resource Interval) The number of frames received, including frames with errors and frames that are not valid.

Total I Frames Trnsmitd
(Resource Interval) The total number of information frames transmitted.

Total I/O
(System) Sum of the read and write operations.

Total PDUs Received
(Resource Interval) The number of protocol data units (PDUs) received during the time interval.

Note: A protocol data unit (PDU) for asynchronous communications is a variable-length unit of data that is ended by a protocol control character or by the size of the buffer.

Total Physical I/O per Second
(Resource Interval) The average number of physical disk I/O operations performed per second by the disk arm.

Total Responses
(Component, Resource Interval) The total number of transactions counted along with the average response time for all active work stations or devices on this controller for the report period.

Total Seize/Wait Time
(Component) The response time in milliseconds for each job.

Total Tns
(Component) Number of transactions processed in this pool.

Transaction Response Time (Sec/Tns)
(Transaction) The response time in seconds for each transaction. This value includes no communications line time. Response times measured at the work station exceed this time by the data transmission time (the time required to transmit data from the work station to the processing unit and to transmit the response data back to the work station from the processing unit).

Transactions per hour (local)
(System) The interactive transactions per hour attributed to local display stations.

Transactions per hour (remote)
(System) The interactive transactions per hour attributed to remote display stations.

Transient Size
(Component) Kilobytes placed within the journal transient area; these are hidden journal entries produced by the system.

Transmit/Receive/Average Line Util
(Resource Interval) In duplex mode, the percentage of transmit line capacity used, the percentage of receive line capacity used, and the average of the transmit and receive capacities.

TSE
(Transaction) Listed in the Wait Code column, Time Slice End. The program shown in the stack entry labeled LAST is the program that went to time slice end.

Typ
(Component, Transaction) The system job type and subtype. The Component Report allows only one character in this column. The Transaction Report allows two characters. The Transaction Report reports the job type and job subtype directly from the QAPMJOBS fields. The Component Report takes the job type and job subtype values and converts it to a character that may or may not be the value from the QAPMJOBS field. The possible job types are:

A
Autostart

B
Batch

BD
Batch immediate (Transaction only)

Note: The batch immediate values are shown as BCI on the Work with Active Job display and as BATCHI on the Work with Subsystem Job display.

BE
Batch evoke (Transaction only)

BJ
Batch pre-start job (Transaction only)

C
Programmable work station application server, which includes 5250 emulation over APPC and IBM i Access host servers running either APPC or TCP/IP. See the Host server administration topic for more information. A job is reported as a IBM i Access server if any of the following items are true:

- Incoming APPC evoke requests one of the server program names. This also applies to the pre-started jobs for the QSERVER, QCMN, and QSYSWRK subsystems that are already waiting for the named program.
- Incoming IP port number corresponds to one of the service name-description-port-numbers. This also applies to the pre-started jobs for the QSERVER, QCMN, and QSYSWRK subsystems that are already waiting for the assigned IP port number.
- Incoming IPX socket number corresponds to one of the service name-description-port-numbers. This also applies to the pre-started jobs for the QSERVER, QCMN, and QSYSWRK subsystems that are already waiting for the assigned IPX port number.
- Incoming 5250 display emulation jobs that come from APPC data streams sent by 5250 emulation under OS/2 Communications Manager or WARP equivalent.

D
Target distributed data management (DDM) server

I
Interactive. For the Component Report, this includes twinaxial data link control (TDLC), 5250 remote workstation, and 3270 remote workstation. For the Transaction Report, this includes twinaxial data link control (TDLC), 5250 remote workstation, 3270 remote workstation, SNA pass-through, and 5250 Telnet.

L
Licensed Internal Code Task

M
Subsystem monitor

P
SNA pass-through and 5250 Telnet pass-through. On the Transaction Report, these jobs appear as I (interactive).

R
Spool reader

S
System

W
Spool writer, which includes the spool write job, and if Advanced Function Printing (AFP) is specified, the print driver job.

WP
Spool print driver (Transaction only)

X
Start the system

The possible job subtypes are:

D
Batch immediate job

E
Evoke (communications batch)

J
Pre-start job

P
Print driver job

T
Multiple requester terminal (MRT) (System/36 environment only)

3
System/36

Notes:

1. Job subtypes do not appear on the Component Report.
2. If the job type is blank or you want to reassign it, use the Change Job Type (CHGJOBTYP) command to assign an appropriate job type.

Type
(System, Transaction, Job Interval) One of the transaction types listed in the description of the DTNTY field.

(System)
The disk type.

(Transaction)
The type and subtype of the job.

(Transaction)
For the Seize/Lock Conflicts by Object section, the type of seize/lock conflict.

UDP Datagrams Received
(Component) The total number of User Datagram Protocol (UDP) datagrams delivered to UDP users.

UDP Datagrams Sent
(Component) The total number of User Datagram Protocol (UDP) datagrams sent from this entity.

Uncap CPU Avail
(Component)Percentage of CPU time available to a partition in the shared processors pool during the interval in addition to its configured CPU. This value is relative to the configured CPU available for the particular partition.

Unicast Packets Received
(System) The total number of subnetwork-unicast packets delivered to a higher-layer protocol. The number includes only packets received on the specified interface.

Unicast Packets Sent
(System) The total number of packets that higher-level protocols requested to be transmitted to a subnetwork-unicast address. This number includes those packets that were discarded or were not sent.

Unit
(System, Component, Resource Interval) The number assigned by the system to identify a specific disk unit or arm. An 'A' or 'B' following the unit number indicates that the disk unit is mirrored. (For example, 0001A and 0001B are a mirrored pair.)

Unit Name
The resource name of the disk arm.

User ID
(System, Component, Transaction, Job Interval, Pool) The list of users selected to be included (SLTUSRID parameter) or excluded (OMTUSRID parameter).

User Name
(Component, Transaction, Job Interval, Batch Job Trace) Name of the user involved (submitted the job, had a conflict, and so on.)

User Name/Thread
(Component, Transaction) If the job information contains a secondary thread, then this column shows the thread identifier. If the job information does not contain a secondary thread, then the column shows the user name. The system assigns the thread number to a job as follows:

- The system assigns thread IDs sequentially. When a job is started that uses a job structure that was previously active, the thread ID that is assigned to the initial thread is the next number in the sequence.
- The first thread of a job is assigned a number.
- Any additional threads from the same job are assigned a number that is incremented by 1. For example:

```
Job Name   User Name/  Job Number
            Thread
QJVACMDSRV   SMITH       023416
QJVACMDSRV    00000006   023416
QJVACMDSRV    00000007   023416
QJVACMDSRV    00000008   023416
```

A thread value greater than 1 does not necessarily mean the job has had that many threads active at the same time. To determine how many threads are currently active for the same job, use the WRKACTJOB, WRKSBSJOB, or WRKUSRJOB commands to find the multiple three-part identifiers with the same job name.

User Starts
(Component) The number of start journal operations initiated by the user.

User Stops
(Component) The number of stop journal operations initiated by the user.

User Total
(Component) The total number of journal deposits resulting from system-journaled objects.

Util
(Component, Resource Interval) The percent of utilization for each local work station, disk, or communications IOP, controller, or drive.

Note: The system-wide average utilization does not include data for mirrored arms in measurement intervals for which such intervals are either in resuming or suspended status.

Util 2
(Component, Resource) Utilization of co-processor.

Value
(Transaction) For the Individual Transaction Statistics section of the Job Summary report, it is the value of the data being compared for the transaction. For the Longest Seize/Lock Conflicts section, it is the number of seconds in which the seize or lock conflict occurred.

Verify
(Component) Number of verify exceptions per second. Verify exceptions occur when a pointer needs to be resolved, when blocked MI instructions are used at security levels 10, 20, or 30, and when an unresolved symbolic name is called. This count could be very high, even under normal system operation. Use the count as a monitor. If there are large variations or changes, explore these variations in more detail.

VP
(System) The number of virtual processors active in the reported partition.

Vrt Shr Proc Pool ID
(System) Virtual Shared Processor Pool ID. This column is only printed for the IBM i partition.

W-I Wait/Tns
(Transaction) The average time, in seconds, of wait-to-ineligible time per transaction. This value is an indication of what effect the activity level has on response time. If this value is low, the number of wait-to-ineligible transitions probably has little effect on response time. If the value is high, adding additional interactive pool storage and increasing the interactive pool activity level should improve response time. If you are unable to increase the interactive pool storage (due to limited available storage), increasing the activity level may also improve response time. However, increasing the activity level might result in excessive faulting within the storage pool.

Wait Code

(Transaction) The job state transition that causes the trace record to be produced. The values can be as follows:

EVT

Event Wait. A long wait that occurs when waiting on a message queue.

EOTn

End of transaction for transaction for type n. These codes are in the wait code column, but they are not wait codes. They indicate transaction boundary trace records.

EORn

End of response time for transaction n. These codes are in the wait code column, but they are not wait codes. They indicate transaction boundary trace records.

Error Responses

(Component> The number of responses in error.

HDW

Hold Wait (job suspended or system request).

LKRL

Lock Released. The job released a lock it had on the object named on the next detail line of the report (OBJECT --). The job that was waiting for the object is named on this line (WAITER --) along with the amount of time the job spent waiting for the lock to be released.

LKW

Lock Wait. If there are a number of these, or you see entries with a significant length of time in the ACTIVE/RSP* column, additional analysis is necessary. The LKWT report lines that precede this LKW report line show you what object is being waited on, and who has the object.

LKWT

Lock Conflict Wait. The job is waiting on a lock conflict. The time (*/ time /*) is the duration of the lock conflict and, though not equal to the LKW time, should be very close to it. The holder of the lock is named at the right of the report line (HOLDER --). The object being locked is named on the next report line (OBJECT --).

SOTn

Start of transaction n. These codes are in the wait code column, but they are not wait codes. They indicate transaction boundary trace records.

SWX

Short Wait Extended. The short wait has exceeded a 2-second limit and the system has put the transaction into a long wait. This long wait must be charged to the transaction response time. In other words, this active-to-wait transaction does not reflect a transaction boundary.

SZWG

(Transaction) Listed in the Wait Code column, Seize Wait Granted. The job was waiting on a seize conflict. The original holder released the lock that it had on the object, and the lock was then granted to the waiting job. The job that was waiting for the object is named on this line (WAITER --) along with the amount of time the job spent waiting for the seize conflict to be released. The object that is held is named on the next line of the report (OBJECT --).

SZWT

Seize/Lock Conflict Wait. The job is waiting on a seize/lock conflict. The time (*/ time /*) is the duration of the seize/lock conflict, and is included in the active time that follows it on the report. The holder of the lock is named at the right of the report line (HOLDER --). The object being held is named on the next report line (OBJECT --).

TSE

Time Slice End. The program shown in the stack entry labeled LAST is the program that went to time slice end. Every time a job uses 0.5 seconds of CPU time (0.2 seconds on the faster processors) between long waits, the system checks if there are jobs of equal priority on the CPU queue. If there are, then the next job with equal priority is granted the CPU and the interrupted job is moved to the queue as the last of equal priority. The job, however, retains its activity level. This is an internal time slice end. When a job reaches the external time slice value, there can

be a job state transition from active to ineligible if another job is waiting for an activity level. When a job is forced out of its activity level, its pages are liable to be stolen by other jobs, and cause additional I/O when the job regains an activity level. The IBM-supplied default values of 2 seconds for interactive jobs and 5 seconds for batch jobs may often be too high, especially for the high-end processors. As an initial value, set the time slice at 3 times the average CPU seconds per transaction.

WTO
Wait Timed Out. The job has exceeded the wait time-out limit defined for a wait (such as a wait on a lock, a message queue, or a record).

WAITS
(Job Trace) The number of waits that occurred.

WAIT-ACT
(Job Trace) In the Job Trace Analysis Summary, this is the time between the ENDTNS and STRTNS programs is labeled WAIT-ACT. If you were tracing an interactive job and used the default STRTNS and ENDTNS parameters, this value is the time taken to process the transaction.

In the Job Trace Analysis I/O Summary, this is the time that the job was inactive, probably due to typing or think time by the user.

Wait-Inel
(System, Component) Average number of wait-to-ineligible job state transitions per minute.

Work Station Controller
(Resource Interval) The name of the remote work station controller.

WRITES
(Job Trace) The number of physical writes that occurred.

Writes per Second
(Resource Interval) The average number of disk write operations performed per second by the disk arm.

WRITTEN
(Job Trace) The number of physical writes that occurred for the entry.

WTO
(Transaction) Listed in the Wait Code column, Wait Timed Out. The job has exceeded the wait time-out limit defined for a wait (such as a wait on a lock, a message queue, or a record).

0.0-1.0
(Component, Resource Interval) The number of times the response time was between 0 and 1 second.

1.0-2.0
(Component, Resource Interval) The number of times the response time was between 1 and 2 seconds.

2.0-4.0
(Component, Resource Interval) The number of times the response time was between 2 and 4 seconds.

4.0-8.0
(Component, Resource Interval) The number of times the response time was between 4 and 8 seconds.

Performance Report header
Each report, regardless of the type or section, contains information in the header of the report that identifies characteristics of the data. Look here for descriptions of the header information.

Report title
Identifies the type of performance report on the first line. The second line identifies the section of the report.

Current date and time
Indicates the date and time the report was printed.

Report page number
Identifies the page of the report.

Perf data from *time* to *time* at *interval*
Indicates the time period over which the data was collected and at what interval.

User-selected report title
Indicates the name assigned to the report by a user.

Member
Indicates the performance data member used in the report. This name corresponds to the name used on the MBR parameter of the Create Performance data (CRTPFRDTA) command.

Library
Identifies the library where the performance data used for a particular report is located.

Model/Serial
Indicates the model and serial number of the server on which the performance data for the report was collected. The serial number can be 10 characters.

Main storage size
Indicates the size of the main storage on the server on which the performance data was collected.

Started
Indicates the date and time Collection Services started collecting performance data for the report. Depending on whether or not you select specific intervals or a specific starting time, you could see the following:

- If you specify no intervals at which to run the report, the start date and time is the date and time at which the data was collected.
- If you specify specific intervals at which to run the report, the start date and time is the date and time at which the data was collected.

  Note: For the System Report only, you should consult the Report Selection Criteria section to find out which intervals were selected.

Stopped
The date and time Collection Services stopped collecting performance data for this report. Depending on whether or not you select specific intervals or a specific ending time, you could see the following:

- If you specify no intervals at which to run the report, the stop date and time is the date and time at which the data was collected.
- If you specify specific intervals at which to run the report, the stop date and time is the date and time at which the data was collected.

  Note: For the System Report only, you should consult the Report Selection Criteria section to find out which intervals were selected.

System name
Indicates the name of the server on which the performance data was collected for the report.

Version/Release level
`x/ x.0` indicates which version and release level of the operating system the server was running at the time the performance data was collected.

Partition ID
Identifies the ID of the partition on which the collection was run. This change accommodates the logical partition implementation. Here are some of the values that you might see:

- If your system is not partitioned (which is the default) or you used Collection Services to collect and print the performance data for the primary partition of a logical partition system, this value is 00.
- If you collected data with the Start Performance Monitor (STRPFRMON) command in a previous release, the value for the partition ID is 00.
- If you used Collection Services to collect and print the performance data in any secondary partition of a logical partition system, this value is the same as the partition ID that is shown on the Work with System Partitions display under the Start Service Tools (STRSST) command.

Feature Code
Identifies the Interactive feature code value for the server.

Int Threshold
Indicates the percent of the total system CPU for interactive work that was used during the collection period. The value is obtained from the QAPMCONF file (GKEY IT) and reflects the configuration metric obtained when the collection started. You should be aware that this value may change for each interval within a collection period due to dynamic changes in logical partition configuration.

Virtual Processors
The number of virtual processors configured for the partition. The value is obtained from the QAPMCONF file (GKEY 13)and reflects the configuration metric obtained when the collection started. You should be aware that this value may change for each interval within a collection period due to dynamic changes in logical partition configuration.

Processor Units
The number of processor units allocated to the partition. The value is obtained from the QAPMCONF file (GKEY PU) and reflects the configuration metric obtained when the collection started. You should be aware that this value may change for each interval within a collection period due to dynamic changes in logical partition configuration.

Processing units are a unit of measure for shared processing power across one or more virtual processors. One shared processing unit on one virtual processor accomplishes approximately the same work as one dedicated processor. One shared processing unit on two virtual processors accomplishes approximately half the work of two dedicated processors.

Column headings
Each report also has several columns that make up the information of the report. Some are specific to a particular report and others are consistent between reports. For short descriptions of these columns, see the Performance Report columns page.

Scenarios: Performance

One of the best ways to learn about performance management is to see examples that illustrate how you can use these applications or tools in your business environment.

Scenario: Improving system performance after an upgrade or migration

In this scenario, you have just upgraded or migrated your system and it now appears to be running slower than before. This scenario will help you identify and fix your performance problem.

Situation

You recently upgraded your system to the newest release. After completing the upgrade and resuming normal operations, your system performance has decreased significantly. You would like to identify the cause of the performance problem and restore your system to normal performance levels.

Details

Several problems may result in decreased performance after upgrading the operating system. You can use the performance management tools included in IBM i and IBM Performance Tools for i licensed program (5770-PT1) to get more information about the performance problem and to narrow down suspected problems to a likely cause.

1. Check CPU utilization. Occasionally, a job will be unable to access some of its required resources after an upgrade. This may result in a single job consuming an unacceptable amount of the CPU resources.
   - Use WRKSYSACT, WRKSYSSTS, WRKACTJOB, or IBM Navigator for i monitors to find the total CPU utilization.
   - If CPU utilization is high, for example, greater than 90%, check the amount of CPU utilized by active jobs. If a single job is consuming more than 30% of the CPU resources, it may be missing file calls

or objects. You can then refer to the vendor, for vendor-supplied programs, or the job's owner or programmer for additional support.

2. Start a performance trace with the STRPFRTRC command, and then use the system and component reports to identify and correct the following possible problems:
   - If the page fault rate for the machine pool is higher than 10 faults/second, give the machine pool more memory until the fault rate falls below this level.
   - If the disk utilization is greater than 40%, look at the waiting and service time. If these values are acceptable, you may need to reduce workload to manage priorities.
   - If the page faults in the user pool are unacceptably high, you might want to automatically tune performance.
3. Run the job summary report and refer to the Seize lock conflict report. If the number of seize or lock conflicts is high, ensure that the access path size is set to 1TB. If the seize or lock conflicts are on a user profile, and if the referenced user profile owns many objects, reduce the number of objects owned by that profile.

Scenario: System monitor

See an example system monitor that alerts you if the CPU utilization gets too high and temporarily holds any lower priority jobs until more resources become available.

Situation

As a system administrator, you need to ensure that the system has enough resources to meet the current demands of your users and business requirements. For your system, CPU utilization is an important concern. You would like the system to alert you if the CPU utilization gets too high and to temporarily hold any lower priority jobs until more resources become available.

To accomplish this, you can set up a system monitor that sends you a message if CPU utilization exceeds 80%. Moreover, it can also hold all the jobs in the QBATCH job queue until CPU utilization drops to 60%, at which point the jobs are released, and normal operations resume.

Configuration example

To set up a system monitor, you need to define what metrics you want to track and what you want the monitor to do when the metrics reach specified levels. To define a system monitor that accomplishes this goal, complete the following steps:

1. In IBM Navigator for i, select **Monitors** > **System Monitors**. From the **Actions** menu, select **Create New System Monitor...**
2. On the **General** page, enter a name and description for this monitor. Click **Next**.
3. Add and edit the **CPU Utilization (Average)** metric properties by performing the following steps:
   a. To add the metric, select **CPU Utilization (Average)** from the list of Available Metrics, and click **Add**. CPU Utilization (Average) is now listed under Metrics to monitor.
   b. To edit the metric properties, click the **CPU Utilization (Average)** metric in the **Metrics to monitor** list. This action opens the **Configure Metric** page where you can edit the properties of the metric.
   c. For **Collection interval**, specify how often you would like to collect the data. This action overrides the Collection Services setting. For this example, specify **30 seconds**.
   d. For **Threshold 1**, enter the following values to send an inquiry message if the CPU Utilization is greater than or equal to 80%:
      i) Select **Enable threshold**.
      ii) For the threshold trigger value, specify **>= 80** (greater than or equal to 80 percent busy).
      iii) For **Duration**, specify **1** interval.

iv) For the **IBM i command**, specify the following:

```
SNDMSG MSG('Warning,CPU...') TOUSR(*SYSOPR) MSGTYPE(*INQ)
```

v) For the threshold reset value, specify **< 60** (less than 60 percent busy). This action resets the monitor when CPU utilization falls below 60%.

e. For **Threshold 2**, enter the following values to hold all the jobs in the QBATCH job queue when CPU utilization stays above 80% for five collection intervals:

i) Select **Enable threshold**.

ii) For the threshold trigger value, specify **>= 80** (greater than or equal to 80 percent busy).

iii) For **Duration**, specify **5** intervals.

iv) For the **IBM i command**, specify the following:

```
HLDJOBQ JOBQ(QBATCH)
```

v) For the threshold reset value, specify **< 60** (less than 60 percent busy). This action resets the monitor when CPU utilization falls below 60%.

vi) For **Duration**, specify **5** intervals.

vii) For the **IBM i command**, specify the following:

```
RLSJOBQ JOBQ(QBATCH)
```

This command releases the QBATCH job queue when CPU utilization stays below 60% for five collection intervals.

4. Click **OK** to save the metric properties.
5. Click **Next** to view the monitor summary page.
6. Click **Finish** to save the monitor.
7. From the list of system monitors, right-click the new monitor and select **Start**.

Results

The new monitor collects the CPU utilization, with new data points added every 30 seconds, according to the specified collection interval. The monitor automatically carries out the specified threshold actions whenever CPU utilization reaches 80%. The monitor will continue to run and perform threshold actions until you stop the monitor.

Note: This monitor tracks only CPU utilization. However, you can include any number of the available metrics in the same monitor, and each metric can have its own threshold values and actions. You can also have several system monitors that run at the same time.

Scenario: Message monitor

This example describes a monitor that displays any inquiry messages in your message queue that occur on your system.

Situation

As a system administrator, you need to be aware of inquiry messages as they occur across your system. You can set up a message monitor to display any inquiry messages in your message queue that occur on your system.

Configuration example

To set up a message monitor, you need to define the types of messages you would like to watch for and what you would like the monitor to do when these messages occur. To set up a message monitor that accomplishes this goal, complete the following steps:

1. In IBM Navigator for i, select **Monitors** > **Message Monitors**. From the **Actions** menu, select **Create New Message Monitor**.
2. On the **General** page, enter a name and description for this monitor. Click **Next**.
3. On the **Message Queue** page, enter the following values:
   a. For **Message Queue to Monitor**, specify **QSYSOPR**.
   b. For **Library**, specify **QSYS**.
   c. Click **Next**.
4. On the **Message Set** page, perform the following steps:
   a. On the **Message Set 1** tab, click **Add**.
   b. On the **Add A Message Set** page, enter the following values:
      i) Select **Add a user defined set of messages**.
      ii) For **Message Type**, select **Inquiry**.
      iii) Click **OK**.
   c. Select **Set the message trigger and reset**.
   d. For **Trigger at the following message count**, specify **1**.
   e. Click **Next**.
5. Click **Next** to view the monitor summary page.
6. Click **Finish** to save the monitor.
7. From the list of message monitors, right-click the new monitor and select **Start**.

Results

The new message monitor displays any inquiry messages sent to QSYSOPR.

Note: This monitor responds to only inquiry messages sent to QSYSOPR. However, you can include two different sets of messages in a single monitor, and you can have several message monitors that run at the same time. Message monitors can also carry out IBM i commands when specified messages are received.

Related information for Performance

The following product manuals and IBM Redbooks (in PDF format), websites, and IBM Knowledge Center topics relate to the Performance topic. You can view or print any of the PDFs.

Manuals

- Performance Tools for iSeries

  This book provides the programmer with the information needed to collect data about the system, job, or program performance. It also includes tips for printing and analyzing performance data to identify and correct inefficiencies that might exist and information about the Manager and Agent features.

- IBM Power® Systems Performance Capabilities Reference

  This reference provides guidance in terms of IBM i operating system performance, capacity planning information, and tips to obtain optimal performance on the IBM i operating system.

- IBM i on Power - Performance FAQ

  This reference provides a basic understanding of IBM i on Power Systems performance concepts, workloads and benchmarks on Power Systems, capacity planning, performance monitoring and analysis, frequently asked questions and guidelines addressing common performance issues.

- IBM Power Systems Performance Report

This reference contains performance and benchmark results for IBM Power Systems servers.

IBM Redbooks

- End to End Performance Management on IBM i

  The topics in this IBM Redbooks publication help you better understand the cycle of Performance Management and provide you with tips and best practices. Information is provided on the following data collectors: Collection Services, Job Watcher, Disk Watcher, and Performance Explorer. It also provides information about how to maximize performance analysis by using the web-based graphical user interface in IBM Navigator for i.

- IBM Systems Director Navigator for i

  Learn how to use the IBM Navigator for i web-based console to manage your IBM i. The information included is intended to help you start using this console and provide tips for working with various parts of the console. It provides details on many of the individual tasks (functions) that are included such as Network, Database, Performance, File Systems, Advanced Job Scheduler for IBM i, Security, and Integrated Server Administration.

- IBM eServer™ iSeries Performance Management Tools

  Learn about the complete array of IBM i performance management tools. This IBM Redpaper is designed to help you understand the different performance management tools that are available to you and when to use them.

- AS/400 HTTP Server Performance and Capacity Planning

  The Internet and web browser-based applications have had a profound effect on how organizations distribute information, perform business processes, service customers, and reach new markets. This book is intended for IBM i programmers, network and system management professionals, and other information technologists who are responsible for designing, developing, and deploying web-based applications and information systems.

- DB2 UDB/WebSphere Performance Tuning Guide

  This document provides an overview of WebSphere Application Server architecture and its main components and introduces some of its key application tuning parameters and system tuning parameters.

- IBM eserver iSeries Universal Connection for Electronic Support and Services

  This document introduces Universal Connection. It also explains how to use the variety of support tools that report inventories of software and hardware on your machine to IBM so you can get personalized electronic support, based on your system data.

- Java and WebSphere Performance on IBM eserver iSeries Servers

  This document provides tips, techniques, and methodologies for working with Java and WebSphere Application Server performance-related issues.

- Lotus® Domino for AS/400: Performance, Tuning, and Capacity Planning

  This document describes a methodology for performance management. It includes setting up performance objectives, collecting and reviewing performance data, tuning of resources, and capacity planning. Performance guidelines and application design tips are also provided.

Websites

- IBM i Technology Updates - Performance Tools (IBM Support)

See the Performance Tools topic of the IBM i Technology Updates website on IBM Support to read about the most recent enhancements to various IBM i performance tools. The topic includes updates for the performance data collectors (Collection Services, Disk Watcher, Job Watcher, and Performance Explorer), the performance components of IBM Navigator for i, and the IBM Performance Tools for i (5770-PT1) licensed program. This website also includes a resources page with links to a wide variety of performance reference materials: forums, blogs, Redbooks, white papers, presentations, articles, and websites.

Saving PDF files

To save a PDF on your workstation for viewing or printing:

1. Right-click the PDF link in your browser.
2. Click the option that saves the PDF locally.
3. Navigate to the directory in which you would like to save the PDF.
4. Click **Save**.

Downloading Adobe Acrobat Reader

You need Adobe Reader to be installed on your system to view or print these PDFs. You can download a free copy from the Adobe Web site (https://get.adobe.com/reader/).

Code license and disclaimer information

IBM grants you a nonexclusive copyright license to use all programming code examples from which you can generate similar function tailored to your own specific needs.

SUBJECT TO ANY STATUTORY WARRANTIES WHICH CANNOT BE EXCLUDED, IBM, ITS PROGRAM DEVELOPERS AND SUPPLIERS MAKE NO WARRANTIES OR CONDITIONS EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT, REGARDING THE PROGRAM OR TECHNICAL SUPPORT, IF ANY.

UNDER NO CIRCUMSTANCES IS IBM, ITS PROGRAM DEVELOPERS OR SUPPLIERS LIABLE FOR ANY OF THE FOLLOWING, EVEN IF INFORMED OF THEIR POSSIBILITY:

1. LOSS OF, OR DAMAGE TO, DATA;
2. DIRECT, SPECIAL, INCIDENTAL, OR INDIRECT DAMAGES, OR FOR ANY ECONOMIC CONSEQUENTIAL DAMAGES; OR
3. LOST PROFITS, BUSINESS, REVENUE, GOODWILL, OR ANTICIPATED SAVINGS.

SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF DIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, SO SOME OR ALL OF THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

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:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 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 YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

© Copyright IBM Corp. 2002, 2015

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 document 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.

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. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample 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_.

Programming interface information

This Performance publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

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 of Oracle, Inc. in the United States, other countries, or both.

Other product and service names might be trademarks of IBM or other companies.

Terms and conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

Personal Use: You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use: You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Product Number: 5770-SS1